Université de Neuchâtel
Faculté des sciences économiques

Design, Development and Evaluation of Gossip Learning Algorithms in Dynamic Setups

---

PhD thesis submitted to the Faculty of Economics and Business
Institute of Information Management
University of Neuchâtel

For the PhD degree in Computer Science
by

Mina Aghaei Dinani

Approved by the dissertation committee:

Prof. Adrian HOLZER, University of Neuchâtel, co-supervisor of the thesis
Prof. Paul COTOFREI, University of Neuchâtel
Prof. Yiyu Lydia CHEN, University of Neuchâtel
Prof. Giovanna DI MARZO SERUGENDO, University of Geneva
Prof. Benoît GARBINATO, University of Lausanne

Defended on 1 September 2025

unine
Université de Neuchâtel
Faculté des sciences économiques

IMPRIMATUR POUR LA THÈSE

Design, Development and Evaluation of Gossip Learning Algorithms in Dynamic Setups

Mina AGHAEI DINANI

UNIVERSITÉ DE NEUCHÂTEL
FACULTÉ DES SCIENCES ÉCONOMIQUES

La Faculté des sciences économiques,
sur le rapport des membres du jury

Prof. Adrian HOLZER, Université de Neuchâtel, co-directeur de thèse
Prof. Paul COTOFREI, Université de Neuchâtel
Prof. Yiyu Lydia CHEN, Université de Neuchâtel
Prof. Giovanna DI MARZO SERUGENDO, Université de Genève
Prof. Benoît GARBINATO, Université de Lausanne

autorise l'impression de la présente thèse.

Neuchâtel, le 14 octobre 2025

Le doyen
Adrian Holzer

Faculté des sciences économiques
Avenue du 1er-Mars 26
CH-2000 Neuchâtel
Email: secretariat.seco@unine.ch
Tel: +41 32 718 15 00
www.unine.ch/seco

Abstract

Mina Aghaei Dinani

Design, Development and Evaluation of Gossip Learning Algorithms in Dynamic Setups

The rise of edge computing and the growing demand for privacy-preserving Machine Learning (ML) have accelerated the need for decentralized learning methods. Among these, Gossip Learning (GL) offers a fu lly distributed, serverless approach in which nodes exchange and merge models to collaboratively learn without sharing raw data. While promising for scalability and privacy, existing GL frameworks typically assume static or fully connected topologies, which do not hold in real-world environments such as vehicular and Internet of Things (IoT) networks, characterized by mobility, node churn, and resource constraints.

This thesis addresses these limitations by designing, developing, and evaluating a suite of GL algorithms tailored for dynamic network environments. The work is organized into three main parts.

Part I proposes personalized GL frameworks for highly dynamic networks such as VANETs, where nodes maintain individualized models and improve them via peer-to-peer exchange. We develop novel merging strategies, including marginal utility-based weighting and location-based averaging, that enhance learning performance under non-Independent and Identically Distributed (non-IID) data distributions and transient connectivity. Applied to trajectory nowcasting, our methods demonstrate fast convergence and robust accuracy across a diverse range of urban mobility scenarios.

Part II investigates how temporal and structural changes in network topology impact GL performance. Existing studies often rely on static graphs or application-specific mobility traces, providing limited generalizable insights into dynamic envi-

ronments. To address this gap, we develop a principled evaluation framework based on Time-Varying Graphs (TVG)s, capturing key real-world phenomena such as node churn, edge memory, and evolving connectivity. We propose new classes of TVGs which integrate edge persistence and node churn while preserving the structural properties of their static counterparts. We show that the balance between exploration (via rewiring) and exploitation (via stable links) is critical for performance, and that scale-free networks are more resilient than random graphs under high churn conditions. These findings provide crucial insights into designing adaptive, robust GL strategies for volatile, real-world networks.

Part III focuses on energy-efficient learning in resource-constrained environments. We present an optimized GL framework, OGL, that dynamically adjusts local learning parameters, such as training epochs and model exchange frequency, based on contextual information and device constraints. A data-driven DNN-based orchestrator provides tuning guidance, enabling nodes to operate efficiently without centralized control. Evaluation on synthetic graphs and real-world vehicular traces confirms that OGL significantly reduces energy consumption while achieving competitive learning accuracy.

Overall, this thesis advances GL as a practical solution for decentralized learning in volatile, bandwidth-limited, and energy-constrained settings. The proposed methods contribute to both theoretical understanding and real-world applicability of scalable, privacy-preserving learning at the network edge.

Keywords: Gossip Learning, Decentralized Machine Learning, Opportunistic Communication, Time-Varying Graphs, Personalized Learning, Energy-Efficient Edge AI

Résumé

Mina Aghaei Dinani

Conception, Développement et Évaluation d'Algorithmes de Gossip Learning dans des Environnements Dynamiques

La montée en puissance de l'edge computing et la demande croissante pour des méthodes de *Machine Learning* respectueuses de la vie privée ont accéléré le besoin de paradigmes d'apprentissage décentralisé. Parmi eux, le *Gossip Learning* propose une approche totalement distribuée et sans serveur, où les nœuds échangent et fusionnent des modèles pour apprendre de manière collaborative sans partager de données brutes. Bien que prometteur pour la scalabilité et la protection de la vie privée, les cadres existants de *Gossip Learning* supposent souvent des topologies statiques ou entièrement connectées — des hypothèses peu réalistes dans des environnements réels tels que les réseaux véhiculaires (VANETs) et l'Internet des Objets (IoT), caractérisés par la mobilité, le renouvellement des nœuds et des contraintes de ressources.

Cette thèse s'attaque à ces limitations en concevant, développant et évaluant une série d'algorithmes de *Gossip Learning* spécifiquement adaptés aux environnements dynamiques. Le travail est structuré en trois parties principales.

Première partie : Nous proposons des cadres personnalisés de *Gossip Learning* pour les réseaux hautement dynamiques tels que les VANETs, où chaque nœud maintient son propre modèle personnalisé et l'améliore par des échanges pair-à-pair opportunistes. Nous développons de nouvelles stratégies de fusion de modèles, notamment une pondération basée sur l'utilité marginale et une approche de moyenne géolocalisée. Appliquées à la prédiction de trajectoires à court terme, ces méthodes démontrent une convergence rapide et une précision robuste dans des scénarios de mobilité urbaine réalistes.

Deuxième partie : Nous analysons l'impact des structures d'interactions dynamiques sur *Gossip Learning* en introduisant une famille de *Time-Varying Graphs (TVGs)* qui simulent des comportements réalistes de réseau, notamment la persistance des liens et le churn des nœuds. Ces modèles permettent une évaluation contrôlée des facteurs structurels et temporels affectant la convergence. Nos résultats montrent que l'équilibre entre l'exploration (par reconfiguration des liens) et l'exploitation (par stabilisation des connexions) est essentiel pour maintenir de bonnes performances, et que les topologies à attachement préférentiel (scale-free) résistent mieux au churn que les graphes aléatoires.

Troisième partie : Nous nous concentrons sur l'optimisation énergétique dans les environnements contraints. Nous présentons *Optimized Gossip Learning (OGL)*, un cadre qui ajuste dynamiquement les paramètres d'apprentissage locaux — tels que le nombre d'époques d'entraînement et la fréquence d'échanges de modèles — en fonction du contexte et des ressources disponibles à chaque nœud. Un orchestrateur basé sur un *Deep Neural Network (DNN)* fournit des recommandations d'ajustement en temps réel, permettant aux nœuds de fonctionner efficacement sans coordination centralisée. Nos évaluations sur des *TVGs* synthétiques et des traces de mobilité urbaine réelles confirment que notre approche réduit significativement la consommation d'énergie tout en conservant une précision d'apprentissage compétitive.

Dans l'ensemble, cette thèse fait progresser le *Gossip Learning* en tant que solution pratique pour l'apprentissage décentralisé dans des environnements volatils, limités en bande passante et en énergie. Les méthodes proposées apportent à la fois des contributions théoriques et des solutions pratiques pour rendre possible l'intelligence collaborative à la périphérie du réseau.

Mots-clés : Gossip Learning, Machine Learning Décentralisé, Communication Opportuniste, Time-Varying Graphs, Apprentissage Personnalisé, Intelligence Artificielle Économe en Énergie

Acknowledgement

I would like to express my gratitude to my advisors, Professor Adrien Holzer and Professor Gianluca Rizzo, whose guidance and support were instrumental in the completion of this journey. Their wisdom and mentorship have been invaluable.

I am also immensely grateful to Professor Marco Ajmone Marsan for his insightful comments and encouragement, which motivated me to broaden my research from various perspectives.

My heartfelt thanks go to my parents. Although they are not physically present, their memories continue to inspire and strengthen me. Their love and blessings have been my constant source of energy.

I would also like to express my gratitude to my family. To my beloved grandmother, sisters, and nephews, your unwavering faith in me has strengthened me.

This journey toward a PhD was not merely about writing papers; it was a roller-coaster ride filled with highs and lows. Throughout this journey, my partner stood by me, providing support and being my rock in times of need. I am profoundly grateful for his love, patience, and unwavering belief in me.

Lastly, I would like to extend my sincere gratitude to everyone who has touched my life, however briefly, during this journey. No matter how small, each interaction has left an indelible mark on me and shaped the person I am today. Their contributions, seen and unseen, acknowledged and unacknowledged, have played a significant role in the completion of this chapter of my life. Thank you.

Contents

CHAPTER 1

Introduction

1.1 Background

Affordable edge devices and embedded sensors, capable of sensing and interacting with their environments, have revolutionized data generation across numerous domains by producing vast amounts of heterogeneous, real-time data [38, 8]. This data plays a critical role in enabling intelligent decision-making and improving the behavior of complex systems.

One of the most effective approaches for extracting actionable insights from such data is Machine Learning (ML) [5], which enables the discovery of patterns, supports forecasting, and facilitates intelligent automation across various application domains. Advances in ML are driving remarkable progress in Artificial Intelligence (AI), transforming industries such as logistics [11], healthcare [28], education [7], vehicular networks [10], and a wide range of industrial applications of the IoT.

Despite the potential, designing an efficient strategy for training a ML model for specific scenarios and applications presents several challenges. Traditionally, ML models are trained using centralized learning architectures [29], where all data is collected and processed on a central server or in the cloud. In the next part, we will delve into centralized learning, exploring its benefits and limitations.

1.1.1 Centralized Learning

In centralized learning, data is collected from multiple edge devices and sent to a central server where a ML model is trained [29]. This process involves cleaning and pre-processing to ensure the data is ready for training. The prepared data is then

fed into an ML algorithm that identifies patterns and relationships. After training and evaluation, the model is used to make predictions for new data. In practical scenarios, however, data sources are often distributed across different geographical locations, leading to high data traffic and costly communication. Transferring large amounts of data from edge devices to a centralized server can overload the network's bandwidth, causing congestion and performance degradation [19]. Additionally, data privacy and ownership concerns are becoming increasingly important. As many users are reluctant to transfer their data to centralized servers due to privacy risks, further complicated by regulations such as the General Data Protection Regulation (GDPR) [33].

To mitigate these issues, privacy-friendly ML techniques and distributed processing methods, such as Federated Learning (FL), are being developed. With these approaches, data can be processed closer to the source, reducing the burden on the network infrastructure and ensuring data privacy [40]. In the following section, we provide a detailed overview of FL, highlighting its key concepts and showing how it addresses the challenges associated with centralized learning.

1.1.2 Federated Learning (FL)

Several distributed ML approaches have been proposed to overcome the limitations of centralized learning [37, 18]. Among these, FL, first introduced by Google [26], has emerged as one of the most widely adopted paradigms.

FL operates under a synchronous server-client architecture, in which a central server coordinates with a subset of clients during each training round to collaboratively train a shared global model. The server initializes the model parameters, selects a group of clients, and sends them the model. Each selected client then trains the model locally on its private dataset and returns the updated models to the server. The server aggregates the received models (commonly using Federated Averaging [26]) to refine the global model. This iterative process of client selection, local training, and aggregation continues until a convergence criterion is met.

In this approach, instead of transmitting raw data, only models are shared, which enhances privacy and reduces communication overhead [17]. However, FL's reliance on a central coordinating server introduces several challenges. It creates a single point of failure and limits scalability, especially when dealing with a large number of clients or highly heterogeneous devices and data distributions [6]. Such

architecture may be unsuitable for applications requiring high availability or fault tolerance.

To overcome the limitations associated with centralized coordination, fully decentralized learning paradigms have been proposed. The next section explores these approaches in detail.

1.1.3 Gossip Learning (GL)

Decentralized ML algorithms have emerged as a powerful alternative to centralized learning architectures, particularly in large-scale, resource-constrained, or privacy-sensitive environments [22, 20]. By enabling collaborative model training across a network of peers without relying on a central server, these approaches improve scalability, fault tolerance, and data privacy, making them attractive for applications such as mobile edge computing, vehicular networks, and the Internet of Things (IoT).

Among decentralized learning paradigms, GL [30] stands out as a fully serverless alternative to FL. In GL, each node trains a local model and periodically exchanges it with its neighbors through peer-to-peer interactions, merging incoming models with its own. This decentralized protocol eliminates single points of failure, enhances resilience to network disruptions, and significantly reduces communication overhead compared to server-based architectures [15, 12]. Furthermore, by avoiding centralized coordination, GL inherently supports asynchronous updates, making it well-suited for heterogeneous and dynamic environments.

Recent years have witnessed a growing body of work advancing GL schemes across diverse architectures and applications [45, 14, 21, 3]. Contributions span from adaptations of GL for federated settings [14, 45], to applications in IoT [21], healthcare [34], and edge intelligence systems [2]. Moreover, variants such as personalized GL [3] and secure GL protocols have expanded their applicability.

Despite these advances, most existing works assume static or quasi-static network topologies—such as rings, meshes, or expander graphs [22, 34, 3]—and overlook the profound challenges posed by dynamic environments characterized by node churn, intermittent connectivity, and evolving communication patterns. While convergence guarantees and performance analyses in static or mildly dynamic settings are relatively well-established [14], the impact of highly volatile topologies, typical of vehicular networks, robotic swarms, and large-scale sensor systems, remains underexplored.

This critical gap motivates the core problem addressed in this thesis: understanding and quantifying the feasibility, robustness, and trade-offs of GL under realistic, time-evolving network conditions. In the following sections, we formulate this problem precisely and introduce new modeling and evaluation frameworks designed to overcome the limitations of prior work.

1.2 Problem Statement

As discussed in the previous section, existing GL methods are inadequate in environments characterized by frequent topology changes and node churn. This limitation leads to the central issue explored in this thesis, which can be summarized by the following question:

> *How can Gossip Learning algorithms be designed, developed, and evaluated to manage node mobility and churn in dynamic networks effectively?*

To systematically address the overarching research question, the problem is decomposed into three tightly related but distinct sub-problems. Each sub-problem corresponds to a specific class of challenges observed in real-world deployments of GL, including: (i) high mobility and peer churn in vehicular networks, (ii) the broader impact of network-level dynamics on learning performance, and (iii) the need for an adaptable and energy-efficient GL in resource-constrained settings.

These sub-problems are addressed individually in the following three subsections:

- **GL for Dynamic Networks:** How can GL algorithms be designed and evaluated to address the challenges posed by node mobility and churn, particularly in application domains such as Vehicular Ad-hoc Networks (VANETs)?
- **Modeling and Analysis of Network Dynamics Impact:** How can dynamic network properties (e.g., churn rate, topology evolution) be effectively modeled and analyzed to understand their impact on GL performance?
- **Energy-Efficient GL for Resource-Constrained Devices:** How can energy-efficient GL algorithms be designed to enable effective learning on devices with limited communication and computational resources, such as those found in IoT systems?

1.2.1 GL for Dynamic Networks

Real-world dynamic networks, such as VANETs, are characterized by frequent mobility, transient connections, and highly fluctuating node densities [39]. In these settings, data generated at individual nodes is inherently non-IID (non-independent and identically distributed) and exhibits strong heterogeneity shaped by localized contexts, such as geolocation and environmental conditions.

Achieving accurate, real-time predictive capabilities (e.g., trajectory nowcasting, proactive handover, or anomaly detection) in such environments is essential for a variety of applications, including anticipatory radio access network (RAN) management, autonomous driving, and mobile edge computing [44, 23, 24]. However, the deployment of robust, privacy-preserving ML models in these settings is profoundly challenged by intermittent connectivity, stringent communication constraints, and the imperative to safeguard sensitive local information [32].

Traditional centralized ML models and even FL approaches struggle in these scenarios due to their dependence on persistent infrastructure, centralized aggregation, and synchronous training paradigms. These assumptions fail under the constraints of vehicular environments, which are inherently decentralized, privacy-sensitive, and bandwidth-constrained [32].

In this context, GL emerges as a promising alternative, offering a fully decentralized, peer-to-peer learning framework inherently suited to highly dynamic environments [30]. Unlike FL, GL removes the reliance on a central aggregator and leverages opportunistic, local model exchanges to enable scalable and resilient model propagation. However, existing GL methods predominantly assume static or fully connected topologies [4, 14, 13] and fail to address critical challenges such as node churn, intermittent peer availability, and the highly non-IID nature of decentralized data distributions.

These limitations reveal a significant gap: without mechanisms explicitly designed to accommodate the temporal volatility and topological fragmentation of dynamic environments such as VANETs, GL frameworks risk degraded convergence, limited personalization, and heightened privacy vulnerabilities.

To address these challenges, this thesis proposes the design and evaluation of GL algorithms that can sustain learning performance under frequent topology changes, support personalized model updates over non-IID localized data, and operate asyn-

chronously in the absence of persistent connectivity. While the proposed approaches are broadly applicable to dynamic networks, a particular focus is placed on trajectory nowcasting for anticipatory RAN management in vehicular settings.

It is important to note that while strong real-time consistency may be required in certain safety-critical vehicular applications (e.g., collision avoidance), the scenarios addressed here tolerate eventual consistency. In this context, rapid convergence and robust local predictions are more relevant than instantaneous global agreement across all vehicles. Accordingly, strong consistency guarantees fall outside the scope of this work.

To guide this part of the thesis, we investigate the following research questions:

1. How can GL algorithms be adapted to function reliably in highly dynamic environments such as VANETs?
2. How can these algorithms support personalized, online learning under non-IID, context-dependent data distributions?
3. What mechanisms can ensure robust performance despite high node churn and transient peer availability?
4. Can GL achieve effective performance without centralized orchestration?

In the next section, we broaden the focus beyond specific application scenarios and systematically investigate how fundamental network dynamics shape the performance of GL algorithms.

1.2.2 Modeling and Analysis of Network Dynamics Impact:

The previous section outlined the key challenges of applying GL in real-world applications, such as VANETs. While those insights were valuable, they were inherently tied to specific application contexts and limited in their ability to generalize across broader, heterogeneous network conditions. To design GL algorithms that are robust and adaptable in diverse real-world deployments, it is essential to understand how intrinsic network dynamics, independent of any domain-specific trace, affect the core learning process.

Decentralized learning systems such as GL [30, 14] are increasingly deployed in open, dynamic networks, characterized by high node churn, fluctuating link stability,

and unpredictable connectivity patterns. These features can critically impact learning convergence, stability, and fairness. However, existing research remains focused on static networks, trace-driven networks, or narrowly defined dynamic settings, providing limited insight into the performance boundaries of GL under realistic and time-evolving network conditions [30, 14, 27, 9, 31, 34]. While these approaches offer valuable insights, they fail to capture key real-world properties such as node churn, link memory, and long-term topological evolution. Additionally, they offer limited insight into the performance trade-offs under various structural and temporal configurations.

To move beyond these limitations, this part of the thesis abstracts from domain-specific network traces and investigates more fundamental questions: How does GL behave when nodes and links evolve stochastically over time? What structural and temporal properties of the network facilitate or obstruct learning progress? How do churn, link memory, and long-term evolution affect learning convergence and stability?

Addressing these questions requires two foundational components: (i) a principled modeling framework for volatile environments that captures realistic temporal behaviors while avoiding dependence on specific mobility traces, and (ii) a controlled evaluation setup that isolates the influence of individual dynamic properties on learning outcomes.

To address these challenges, we propose and utilize Time-Varying Graphs (TVG)s to emulate a wide spectrum of dynamic behaviors, such as edge rewiring, node churn, and persistence, while preserving key graph-theoretic properties of static network counterparts (e.g., degree distribution, clustering).

The guiding research questions are:

1. How can dynamic graphs be constructed to realistically simulate node churn and link persistence while preserving static graph characteristics?

2. How do different underlying topologies (scale-free, random, community-based) affect GL convergence, stability, and learning efficiency?

3. What is the impact of network connectivity and dynamicity levels on learning progress? Is higher connectivity or higher dynamicity always beneficial?

4. How do node arrivals and departures (churn) disrupt or reinforce decentralized learning? How severe is the cold-start problem?

5. What is the role of edge memory in model propagation, learning fairness, and information diffusion?

6. How can we characterize and optimize the exploration–exploitation trade-off in dynamic, decentralized learning environments?

Answering these questions is key to unlocking the full potential of GL in open, volatile environments such as mobile edge networks, robotic swarms, and IoT ecosystems. It also provides the conceptual basis for the subsequent part of this thesis, where we address another major obstacle to practical deployment: the energy efficiency of GL under resource-constrained conditions.

The following section builds directly on these insights, formulating the problem of designing adaptive and energy-aware GL strategies that can thrive in dynamic, resource-limited networks.

1.2.3 Energy-Efficient GL for Resource-Constrained Devices

Building upon our prior investigation into the influence of network dynamics on the performance of GL, we now turn our focus to another critical axis of real-world feasibility: the resource constraints inherent in edge and IoT environments. While previous sections have addressed how dynamic topologies affect convergence and stability in decentralized learning, they presuppose idealized or unconstrained resource availability. In contrast, many target deployments—such as mobile edge computing, sensor networks, and connected vehicles—operate under stringent energy, bandwidth, and computational limitations [36, 35]. These constraints introduce new challenges that compound the effects of topology evolution, requiring a careful co-design of learning procedures and system-level optimizations.

GL offers a compelling alternative to centralized or FL paradigms due to its robustness and inherent scalability in fully decentralized contexts [30]. However, its high reliance on iterative local training and frequent peer-to-peer (P2P) model exchanges makes it especially sensitive to energy availability, particularly on battery-powered devices. Communication and computation overheads can rapidly deplete power budgets, leading to premature dropout of participants and degraded system performance [41].

While energy-efficient strategies have been extensively studied in the context of FL [46, 1, 25], they are often not directly applicable to GL due to its lack of

centralized coordination and its peer-driven communication model. Moreover, most existing solutions presume static network conditions and uniform device capabilities, failing to address the compounded complexity of fluctuating node participation, varying connectivity, and heterogeneity in resource profiles [43, 42, 16].

To address this gap, this thesis presents a significant contribution: a novel adaptive, GL framework that maintains target model performance while minimizing energy expenditure under dynamic network and resource conditions.

This part of the thesis seeks to answer the following key research questions:

1. How can each node dynamically and autonomously adjust its learning parameters (e.g., number of training epochs, peer selection) to adapt to local context, such as neighbor availability, model quality, and resource constraints?

2. How can a centralized orchestration function assist in pre-training a reusable decision model that allows fully decentralized, real-time tuning of learning processes on the edge?

3. How can we quantify and model the energy consumption of GL across computation and communication layers in resource-constrained dynamic networks?

These questions form the core focus of this thesis. Answering these questions is essential for advancing GL from a proof-of-concept distributed learning method to a practical tool deployable in real-world, constrained environments.

In the next section, we outline the key contributions of this thesis, detailing how our work bridges the gaps identified above and advances the state of the art in both the theory and application of GL in dynamic settings.

1.3 Thesis Contributions

This thesis presents a set of substantial and original contributions to the field of GL, with a focus on making GL algorithms practical, robust, and scalable for deployment in dynamic, real-world network environments. Motivated by the limitations identified in the problem formulation, the work undertaken in this thesis advances the state-of-the-art in decentralized learning by addressing critical challenges related to node mobility, network churn, communication overhead, model personalization, and energy efficiency in resource-constrained settings.

The contributions of this thesis are structured to address the sub-problems identified in the problem formulation directly. A summary of the principal advancements is provided below.

- **Personalized GL in Dynamic Networks:** Unlike prior works that assume a global or cluster-level model [30, 14], we design and develop new GL algorithms in which each node maintains and updates its own personalized model throughout the learning process. Two novel model merging strategies are proposed: one based on marginal utility, the other leveraging geographic proximity. These enable learning from non-IID, localized data under highly dynamic conditions (Chapters 2 and 3).

- **Time-Varying Graph (TVG) Models for Realistic Network Dynamics:** This thesis introduces a principled framework based on TVGs to model the temporal and structural evolution of dynamic networks. This includes a randomized rewiring mechanism to simulate evolving topologies (Chapter 4). In addition, we design three new TVG classes that preserve the statistical properties of classical topologies while modeling temporal behaviors by incorporating node churn and link persistence. To our knowledge, this is the first work to systematically evaluate GL under such broad and controllable dynamic conditions (Chapter 5).

- **Fundamental Analysis of Network Dynamics on GL Performance:** Using both random and structured TVGs, we reveal how factors such as node churn, edge persistence, network topology, and data distribution influence the performance of GL algorithms. Notably, we identify a fundamental trade-off between exploration and exploitation, and show how structural properties shape accuracy and convergence in decentralized learning. These findings generalize beyond specific mobility traces to a wide class of network environments (Chapters 4 and 5).

- **Energy-Aware GL for Resource-Constrained Systems:** This thesis contributes to energy-efficient decentralized learning through two efforts. First, Chapter 6 extends prior work on hybrid content delivery by incorporating a learning-assisted mechanism that combines opportunistic V2V communication with cloud fallback under delay constraints. While not focused on GL, it supports the broader vision of decentralized, adaptive operation in constrained

environments. Second, Chapter 7 presents Optimized Gossip Learning (OGL), where nodes adapt their learning behavior and client selection based on resource context using a pre-trained decision model, significantly improving energy efficiency while maintaining accuracy in dynamic networks.

- **Comprehensive Evaluation across Synthetic and Real-World Environments:** We conduct extensive evaluations of all proposed GL algorithms using both controlled simulations on our proposed TVGs (Chapters 4, 5, and 7) and real-world vehicular mobility traces (Chapters 2,3, 6and7). Application domains include trajectory prediction in vehicular networks (Chapters 2, and 3) and image recognition tasks (Chapters 4, 5, and 7). Our experiments demonstrate the robustness, scalability, and adaptability of the proposed methods across diverse network densities, topologies, and dynamic conditions, including node churn, intermittent connectivity, and heterogeneous data availability. Results confirm that GL can achieve near-centralized performance even in highly fragmented, resource-constrained environments.

Together, these contributions constitute a unified framework for the design, development, and evaluation of GL algorithms in dynamic network environments. Each component builds upon the others to address both theoretical challenges and practical implementation barriers. The work presented in this thesis provides a foundation for deploying GL in a broad range of real-world scenarios, from autonomous transportation systems to large-scale IoT networks.

The remainder of this chapter outlines the structure of the thesis, highlighting how each chapter addresses specific components of the research contributions introduced above.

1.4 Structure of the Thesis

This thesis is structured to progressively address the central research question posed in the problem formulation: how can GL algorithms be designed, analyzed, and optimized to operate effectively in dynamic, resource-constrained network environments? To this end, the thesis is organized into three parts, each corresponding to one of the main challenges identified: (i) enabling GL in highly dynamic networks, (ii) understanding and modeling the impact of network-level dynamics on GL performance, and (iii) ensuring energy-efficient operation in real-world edge and IoT

systems.

The ordering of the thesis parts reflects the chronology of our research. We first tackled the design and evaluation of GL algorithms in practical vehicular settings, and later abstracted these insights into more general models of dynamic networks. This ordering mirrors the research trajectory and highlights how application-driven findings motivated the subsequent development of principled modeling frameworks.

Unlike a traditional monograph, this thesis follows a collection-of-articles format. Each chapter is based on peer-reviewed research published or submitted to international conferences and journals. Where a journal version extends a previous conference paper, only the journal version is included to avoid duplication.

Part I: GL for Highly Dynamic Networks

This part investigates the design and evaluation of GL algorithms for highly dynamic network environments such as VANETs. The focus is on addressing challenges related to node mobility, churn, intermittent connectivity, and non-IID data distributions, as outlined in Subsection 1.2.1. We develop two personalized GL algorithms in which each node maintains and updates its own model through asynchronous, peer-to-peer interactions. These approaches differ in their communication and model integration strategies.

- Chapter 2: This chapter introduces a model exchange mechanism based on training delegation, where nodes send their models to peers for remote updates. The proposed merging strategy, called Decentralized Powerloss (DP), incorporates performance-aware integration by leveraging marginal utility to weigh received models. The proposed method is evaluated using real-world vehicular mobility traces and applied to a classification task that predicts the future location of vehicles within urban regions, in an online manner. Results demonstrate accurate predictions in dense settings, though performance can degrade under sparse connectivity.
  - **Mina Aghaei Dinani**, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. "Gossip Learning of Personalized Models for Vehicle Trajectory Prediction," IEEE WCNC, 2021.
- Chapter 3: To address the limitations identified in the previous chapter, this

chapter uses an opportunistic model exchange scheme in which nodes transmit their trained models upon direct contact. A novel merging strategy, Location-Based Averaging (LBA), is introduced to incorporate spatial context into the integration process. The algorithm is evaluated on the task of online vehicle position nowcasting using real urban mobility traces, demonstrating improved adaptability and accuracy across diverse traffic and mobility patterns when compared to baseline methods. This chapter consolidates and extends results previously published in a conference paper and its journal version.

- **Mina Aghaei Dinani**, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. "Vehicle Position Nowcasting with Gossip Learning," IEEE WCNC, 2022.
- **Mina Aghaei Dinani**, Adrian Holzer, Marco Ajmone Marsan, Gianluca Rizzo. "A Gossip Learning Approach to Urban Trajectory Nowcasting for Anticipatory RAN Management," IEEE TMC, 2023.

Through these works, we demonstrate that it is possible to deploy GL effectively in real-world systems characterized by node mobility, short-lived connectivity, and strict privacy requirements. Although our design and evaluation are centered on vehicular networks, the underlying principles generalize to a broader class of dynamic network environments, including mobile edge computing platforms, ad-hoc wireless sensor networks, and IoT deployments.

These studies reveal broader challenges related to the behavior of GL under dynamic network conditions, such as node churn rate, topology evolution, and intermittent peer availability. To better understand these effects across a broader range of dynamic environments, the next part of this thesis systematically investigates how varying network dynamics influence the performance of GL algorithms beyond the VANETs and in more generalized dynamic settings.

Part II: Modeling and Analysis of Network Dynamics Impact

While Part I focuses on real-world deployment scenarios, this part abstracts from application-specific traces and systematically explores how fundamental network dynamics influence GL performance. Using TVG models, it aims to generalize findings across a wide range of dynamic conditions, as discussed in Subsection 1.2.2.

- Chapter 4: This chapter proposes a random TVG model to establish a baseline for GL performance in dynamic environments. It isolates the effect of connection rewiring and data distribution, revealing that certain levels of network turbulence can benefit learning. However, the model does not include node churn or edge memory.
  - Antonio Di Maio, **Mina Aghaei Dinani**, and Gianluca Rizzo. The Upsides of Turbulence: Baselining Gossip Learning in Dynamic Settings. ACM Mobihoc, 2023.
- Chapter 5: To overcome the limitations of the previous model, this chapter introduces three new TVG classes that incorporate node churn and edge persistence while preserving the statistical properties of their static counterparts. Systematic evaluations show how structural and temporal features—such as connectivity, memory, and churn rate—affect convergence, fairness, and learning efficiency. The results reveal a trade-off between exploration and exploitation, showing that balanced dynamics can achieve near-centralized performance across various network conditions.
  - **Mina Aghaei Dinani**, Antonio Di Maio, and Gianluca Rizzo. Gossip Learning in Edge-Retentive Time-Varying Random Graphs with Node Churn. IEEE AIOT, 2024.
  - **Mina Aghaei Dinani**, Gianluca Rizzo. Exploring Gossip Learning in Dynamic Networks through Time-Varying Graphs. Journal article in progress.

Part III: Energy-Efficient GL for Resource-Constrained Devices

Building on insights from Part II, this final part addresses the challenge of operating GL in energy-constrained environments, such as IoT networks and vehicular systems. As discussed in Subsection 1.2.3, maintaining learning effectiveness under dynamic conditions requires explicit strategies for minimizing communication and computation overhead.

- Chapter 6: Although this chapter is not centered on GL, it addresses a complementary challenge of decentralized, learning-assisted decision-making under

resource and delay constraints in vehicular networks. This chapter broadens the scope of the thesis by demonstrating how the principles of adaptive, context-aware operation—central to GL—can also benefit related tasks such as QoS-aware content delivery. Its inclusion highlights the versatility of decentralized learning-inspired methods in dynamic, resource-constrained environments.

  - **Mina Aghaei Dinani**, Hung Nguyen, Gaetano Manzo, Torsten Braun, Gianluca Rizzo. Learning Assisted QoS-Aware Content Delivery in Fragmented Vehicular Named Data Networks. Submitted to IEEE TMC.

- Chapter 7: This chapter introduces OGL, an energy-efficient GL framework in which each node autonomously adjusts its learning parameters, such as the number of training epochs and model merges, based on resource context and peer availability. A pre-trained DNN orchestrator guides these local decisions. Evaluations on synthetic TVGs and real vehicular traces show that OGL achieves substantial energy savings while preserving learning accuracy, even under mobility and churn.

  - **Mina Aghaei Dinani**, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Context-Aware Orchestration of Energy-Efficient Gossip Learning Schemes. IEEE AIIOT, 2024.

The three parts of this thesis address the core challenges outlined in the problem statement—ensuring that GL remains effective, adaptable, and resource-aware in the face of real-world network dynamics.

Finally, Chapter 8 concludes the thesis and outlines directions for future research.

Bibliography

[1] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. REFL: Resource-efficient federated learning. In *Proceedings of the Eighteenth European Conference on Computer Systems*. ACM, may 2023.

[2] Abdul Aziz Alkathiri, Lodovico Giaretta, Sarunas Girdzijauskas, and Magnus Sahlgren. Decentralized word2vec using gossip learning. In *23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*, 2021.

[3] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 473–481. PMLR, 2018.

[4] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.

[5] Giuseppe Bonaccorso. *Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning*. Packt Publishing, 2017.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[7] Maud Chassignol, Aleksandr Khoroshavin, Alexandra Klimova, and Anna Bilyatdinova. Artificial intelligence trends in education: a narrative overview. *Procedia Computer Science*, 136:16–24, 2018. 7th International Young Scientists Conference on Computational Science, YSC2018, 02-06 July2018, Heraklion, Greece.

[8] Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer, and Ernoe Kovacs. Building a big data platform for smart cities: Experience and lessons from

santander. In *2015 IEEE International Congress on Big Data*, pages 592–599. IEEE, 2015.

[9] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Vehicle position nowcasting with gossip learning. In *IEEE WCNC*, pages 728–733. IEEE, 2022.

[10] Yolanda Blanco Fernández. Intelligent technologies for vehicular networks. *Electronics*, 13(20):3990, 2024.

[11] Michel Gendreau, Gianpaolo Ghiani, and Emanuela Guerriero. Time-dependent routing problems: A review. *Computers & Operations Research*, 64:189–197, 2015.

[12] Peyman Gholami and Hulya Seferoglu. Digest: Fast and communication efficient decentralized learning with local updates. *arXiv preprint arXiv:2307.07652*, 2023.

[13] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings 19*, pages 74–90. Springer, 2019.

[14] István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

[15] István Hegedűs, András Vidács, and László Gyimóthy. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 27(4):870–886, 2015.

[16] Ahmed Imteaj and M. Hadi Amini. Fedar: Activity and resource-aware federated learning model for distributed mobile robots, 2021.

[17] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

[18] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. MLbase: A Distributed Machine-learning System. In *Cidr*, volume 1, pages 2–1, 2013.

[19] Maxime Labonne, Charalampos Chatzinakis, and Alexis Olivereau. Predicting bandwidth utilization on network links using machine learning. *arXiv preprint arXiv:2112.02417*, 2021.

[20] Ameet Lalitha, Aylin Ozgur, and H Vincent Poor. Peer-to-peer federated learning on graphs. In *International Conference on Machine Learning Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[21] Chengxi Li, Gang Li, and Pramod K. Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147, 2022.

[22] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[23] Lei Lin, Siyuan Gong, Tao Li, and Srinivas Peeta. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles. In *The Autonomous Vehicles Symposium*, volume 2018, 2018.

[24] Qian Liu, Gang Chuai, Jingrong Wang, and Jianping Pan. Proactive mobility management with trajectory prediction based on virtual cells in ultra-dense networks. *IEEE Transactions on Vehicular Technology*, 69(8):8832–8842, 2020.

[25] Othmane Marfoq, Giovanni Neglia, Laetitia Kameni, and Richard Vidal. Personalized federated learning through local memorization, 2022.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[27] Othon Michail. An introduction to temporal graphs: An algorithmic perspective*. *Internet Mathematics*, 12(4):239–280, 2016.

[28] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017.

[29] Dishita Naik and Nitin Naik. The changing landscape of machine learning: A comparative analysis of centralized machine learning, distributed machine learning and federated machine learning. In *Advances in Computational Intelligence Systems*. Springer, 2024.

[30] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[31] Luigi Palmieri, Chiara Boldrini, Lorenzo Valerio, Andrea Passarella, and Marco Conti. Impact of network topology on the performance of decentralized federated learning. *Computer Networks*, 253:110681, 2024.

[32] Deval Parikh, Sarangkumar Radadia, and Raghavendra Kamarthi Eranna. Privacy-preserving machine learning techniques, challenges and research directions. *International Research Journal of Engineering and Technology*, 11(03):499, 2024.

[33] Regulation, P. Regulation (eu) 2016/679 of the european parliament and of the council. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, 2016. Accessed: 2024-08-29.

[34] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[35] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[36] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[37] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30:4424–4434, 2017.

[38] Hatem Ben Sta. Quality and the efficiency of data in "Smart-Cities". *Future Generation Computer Systems*, 74:409–416, 2017.

[39] Chai K Toh. *Ad hoc mobile wireless networks: protocols and systems*. Pearson Education, 2001.

[40] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), mar 2020.

[41] Shiqiang Wang, Tianyi Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Optimizing federated learning on non-iid data with reinforcement learning. *IEEE Transactions on Mobile Computing*, 20(4):1392–1406, 2020.

[42] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems, 2019.

[43] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting, 2021.

[44] Z. Xiao, P. Li, V. Havyarimana, G. M. Hassana, D. Wang, and K. Li. GOI: A Novel Design for Vehicle Positioning and Trajectory Prediction Under Urban Environments. *IEEE Sensors Journal*, 18(13):5586–5594, 2018.

[45] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully decentralized joint learning of personalized models and collaboration graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 864–874. PMLR, 2020.

[46] Yuhao Zhou, Ye Qing, and Jiancheng Lv. Communication-efficient federated learning with compensated overlap-fedavg, 2021.

GL for Dynamic Networks

CHAPTER 2

Gossip Learning of Personalized Models for Vehicle Trajectory Prediction

Published in IEEE Wireless Communications and Networking Conference Workshops (WCNC Workshops), Pages: 1–7.
Date: March 2021.
Publisher: IEEE.
ISBN: 978-1-7281-9507-0.

Abstract– Gossip Learning (GL) is a peer-to-peer machine learning protocol based on direct, opportunistic exchange of models among nodes via wireless D2D communications, and on collaborative model training, which has recently proven to scale efficiently to large numbers of nodes, and to offer better privacy guarantees than traditional centralized learning architectures. Existing approaches to GL are, however, limited to scenarios in which nodes are static or in which the node connectivity graph is fully connected, and they are fragile to node churn as well as to any change in network configuration. To overcome this limitation, we present a new decentralized architecture for GL suitable for setups with dynamic nodes, which benefits from node mobility instead of being hampered by it. In our approach, nodes improve their personalized model instance by sharing it with neighbors, and by weighting neighbors' contributions according to an estimate of their marginal utility. We propose a new strategy for the estimation of the neighbors' instances marginal utility, which yields satisfactory trajectory estimation accuracy for nodes with long enough sojourn times.

2.1 Introduction

The amount of data produced by affordable edge devices and sensors has been increasing exponentially over the recent past, thanks to the increasing pervasiveness of the Internet of Things (IoT) paradigm, the growing interest in Smart City applications, and the gradual deployment of 5G networks [25, 7, 14]. The amount of data traffic is poised to grow faster than the number of connections because of the increased use of data-hungry applications, such as telemedicine and smart driving systems [9]. Much of this data is key for enabling decision-making so as to improve the behavior of complex systems. For example, vehicle trajectory prediction creates new opportunities for improving transport services and reducing traffic congestion [28]. An increasing number of services and applications in the vehicular domain rely on trajectory data and AI/ML (artificial intelligence, machine learning) techniques to improve their effectiveness and efficiency.

Standard ML algorithms rely on training over datasets built from a large number of sources, and typically stored centrally, on one machine or in a data center. Storing such data in a central place has become more and more problematic because of data protection rules, and customer privacy concerns. In addition, collecting data from different devices can be inefficient and their transfer can quickly clog the available bandwidth. Several distributed ML approaches have been proposed to overcome these limitations [24, 17]. Among these, Federated Learning (FL), first introduced by Google [20], is based on a synchronous coordinator-client (server-client) architecture. FL collaboratively learns a single consensus model for all clients from decentralized data without the need to store data centrally. Data remains where it was generated, which guarantees privacy and reduces communication costs. However, learning heavily depends on the coordinating server, which causes scalability issues with large numbers of nodes.

Decentralized ML algorithms have been proposed to tackle scalability issues. In these algorithms, learning is implemented collaboratively amongst all nodes with no central server, aggregator or coordinator. One of the state-of-the-art approaches in this field is Gossip Learning (GL) [22], which implements a decentralized version of FL. Each node in this distributed algorithm acts as a client for other nodes. At the same time, it can act as a coordinating server that merges received models. GL has been applied to different ML problems. In [5], a gossip protocol is used in which local models are distributed over a logically fully connected peer-to-peer

network serving an application of distributed learning for medical data centres. However, the solution has scalability and connectivity issues of its own. In [13], a segmented gossip aggregation is introduced. The global model is divided into non-overlapping subsets. Local learners aggregate the segmentation sets from other learners. The proposed approach is application-dependent and not suitable for more general machine learning contexts. Savazzi et al.[23] proposed a fully serverless FL approach, in which nodes receive a combined model from their neighbours, and each one independently performs training steps on its local dataset. Then, similarly to [5], nodes forward updated models to their one-hop neighbourhood for a new consensus step. Their goal is exploiting a serverless consensus paradigm for FL and enhancing the speed of convergence. Individual model instances are not personalized to increase local performance. Moreover, the majority of existing GL approaches consider scenarios in which either each node communicates with all other nodes, or the connectivity graph is static. By not accounting for churn and mobility, these approaches are not applicable in dynamic setups such as vehicular ad hoc networks (VANETs).

In this work, we consider a scenario in which learning agents are vehicles moving in an urban setting, and exchanging information directly in an ad-hoc mode, e.g., via cellular D2D, WiFi direct, or Dedicated Short-Range Communications (DSRC) [3, 1]. We consider the case in which each vehicle has to train a Long Short-Term Memory (LSTM) model in order to predict in an online manner its own trajectory, for purposes of vehicular traffic management, or the implementation of coordinated driving, or for enabling proactive resource allocation algorithms, e.g. in Mobile Edge Computing (MEC) schemes [26].

We propose a decentralized GL scheme which is online, peer-to-peer and based on asynchronous communications. Each node in this network uses its local dataset to improve the model instances of nodes it meets opportunistically. At the same time, each node acts as a coordinating server that merges received models in order to improve the performance of its own personalized model instance.

We present three practical algorithms, called DFed Avg, DFed Pow and DFed MinLoss, to personalize the model instance of each node, based on iterative model averaging. We perform our numerical experiments over measurement-based mobility traces in an urban setting. Results over a clean-slate scenario suggest that these approaches already perform well when vehicles spend at least about 20 minutes in the considered area, even with dynamic, unbalanced and non-IID data distributions.

The rest of this paper is organized as follows: Section 2.2 describes the system model. In Section 2.3, GL algorithms are described in details. Section 2.4 is dedicated to numerical evaluation and results, and finally future work is discussed in Section 2.5.

2.2 System model

We consider a set of wireless nodes, moving on a finite region of the plane according to an arbitrary *stationary* mobility model. We assume nodes know exactly their position at any point in time. Nodes communicate among them using a wireless technology (e.g. WiFi, DSLR, Cellular D2D, among others). We say that two nodes are *in contact* when they are able to exchange information directly.

We assume that the given region of the plane is partitioned into *cells*. Location, size and shape of the region, as well as of each cell are typically determined by the specific application scenario requiring trajectory prediction. For instance, in a setup where vehicles offload part of their computing tasks to a MEC service, a cell of our system may correspond to the coverage area of the roadside unit(s) to which a specific MEC server is associated. The choice of the region instead depends on the spatial range of the specific service requiring trajectory predictions.

In addition to these cells, we define an *outside cell*, consisting of the area of the plane lying out of the given region. Without loss of generality, let us assume time to be divided into slots, and let Δ be the slot duration. Let v be the unique identifier of a vehicle in the given scenario. Starting from the slot at which the $v-th$ vehicle enters the given region (which we denote as slot 1), each vehicle samples its position in space at each time slot. If t is the label of the $t-th$ slot since ingress time, with (x_t, y_t) we denote the vehicle position at the beginning of that slot. The resulting time series $\{(x_t, y_t)\}_{t=1,\dots,t_0}$ constitutes the *local dataset* of the vehicle up to the $t_0 - th$ slot from node ingress.

As explained, we consider a scenario in which, at any slot, each user (vehicle) tries to predict its own location h slots ahead in the future. The outside cell is used to predict whether a node will get out of the given region in h slots. In order to avoid trivial prediction tasks (due to e.g. regular mobility patterns of vehicles) in what follows we adopt a *clean-slate model*, by which we assume that nodes entering the given region possess an empty local dataset. This models the worst-case con-

dition in which vehicles in the scenario are new to the area considered, and thus they cannot rely on data collected before entering the given region for elaborating a prediction of their trajectory. Though far from ordinary operating conditions in a realistic setting, the clean-slate assumption allows a first conservative assessment of our approach, in a way which is independent from any context-dependent assumption on the composition or the size of the initial dataset. As in machine learning techniques more data imply (at least the opportunity of) better performance for the model trained on that data, results obtained in a clean slate scenario may be seen as lower bounds on the actual performance achievable with our approach in a realistic scenario. Of course, note that our approach can be easily extended and adapted to the case in which such assumption does not hold, though in that case the performance (as well as the necessary adaptations) would be a function of the specific assumptions made on the composition of the initial dataset of each node.

2.3 A distributed architecture for Gossip Learning

In this section, we outline the architecture of our new GL protocol, whose goal is to endow each node in the scenario with a machine learning based model capable of accurately predicting its trajectory.

2.3.1 A LSTM architecture for trajectory prediction

In order to implement our learning architecture, we assume that each nodes employs a Long Short Term Memory (LSTM) network, a special kind of Recurrent Neural Network (RNN) already proposed in the literature to predict vehicle trajectories [2]. For the task of time series forecasting, when a sequence of input and output multi-variant data is available [15][21], encoder-decoder LSTM has been shown to outperform other approaches for motion prediction, such as Kalman filtering [6] or Support Vector Machines [18], as their lack of depth does not allow satisfactory prediction performance.

Thus, we assume that each node that traverses the considered region uses its local dataset to train an encoder-decoder LSTM model. Once the model is trained, at every time slot each node feeds the LSTM model with the last α samples of the time serie representing the vehicle's own trajectory in the last α time slots, and it outputs a prediction on where the node will be in h time slots in the future.

The detailed architecture of the encoder-decoder LSTM model is as follows. This model is the concatenation of two LSTM layers (encoder and decoder), each with 50 neurons. The LSTM encoder accepts as input a time series $\{(x_t, y_t)\}_{t=t_1,...,t_1+\alpha}$ of size α by 2 , representing vehicle trajectories over α consecutive time slots. The output of the first LSTM layer (encoder) is a fixed-length vector which captures the temporal structure of the past trajectory. The second LSTM layer (decoder) maps the vector representation back to a variable-length target sequence. The target sequence is the *cell trajectory* $c_1, ..., c_h$ of length h, i.e. a sequence of h cell labels describing the cell in which the vehicle is predicted to be, from time slot $t_1 + \alpha + 1$ up to time slot $t_1 + \alpha + h$. The resulting LSTM network is trained over a sequence of epochs, i.e. of learning iterations performed over the entire dataset. In each epoch, a node trains once the local model instance using its local dataset, and it updates the model parameters. We adopt a mini-batch Gradient Descent training approach, in which the training during one epoch is partitioned in two or more batches of size 32. Finally, we have adopted the Adam optimizer [16] in our model because of its computational efficiency and simplicity, as it requires little memory, and is well suited for problems with many parameters. It is also appropriate to cope with non-stationary objectives. Please note that the number of neurons, the batch size, the number of input time steps as well as the other hyperparameters of our LSTM network and of the training process have been tuned through extensive simulations and testing.

2.3.2 A framework for collaborative training in dynamic settings

In this section we describe a decentralized, collaborative algorithm for training the LSTM network, in a way which maximizes the accuracy of the prediction for each user. An outline of our proposed strategy for decentralized GL is presented in Algorithm 1.

Our algorithm is structured as follows. We assume the time spent by each node in the region to be divided into two stages. **Initialization stage**. It starts from the time slot in which the node enters the considered area (or from the beginning of the scheme if the node is already present in the given region at that point in time), and it lasts V slots. During this stage, the node does not possess yet a sufficiently large dataset to train its local model. Thus the node does not perform any prediction, but it collects data and builds its local dataset. Finally, at the

end of the initialization stage, each node initializes the model by training it on its local dataset. **Exploitation stage**. This second stage starts at slot $V + 1$, and

Algorithm 1 Decentralized GL algorithm.
K_j^v is the set of nodes in their exploitation phase which come in contact with node v during the $j - th$ round.

for every node v **do**
  for slot 1 to V **do** ▷ **Initialization stage**
    Update local dataset
  Train the initial model instance w_0^v over local dataset.
  w_1^v =CLIENTUPDATE(w_0^v)
  for Each round j **do** ▷ **Exploitation stage**
    for Every node $k \in K_j^v$ **do**
      Send model instance w_j^v
      Receive model update $w_j^{v,k}$
      Receive model w_j^k
      ▷ Train model w_j^k on local dataset
      $w_j^{k,v}$ =CLIENTUPDATE(w_j^k)
      Send model update $w_j^{k,v}$ to node k
    ▷ Merge all received model updates
  $w_j^v \leftarrow$MERGEMODELS($\{w_j^{v,k}\}_{k \in K_j^v}$)
  Update local dataset
  Train w_j^v over local dataset.

function CLIENTUPDATE(w_j^v) ▷ Run on client k
  Split local dataset into B batches
  Let $w \leftarrow w_j^v$
  for batch $b \in 1, ..., B$ **do**
    $w \leftarrow w - \eta \bigtriangledown (w; b)$
    ▷ η is the learning rate
  return $w_j^{v,k} \leftarrow w$

terminates when the node exits the given region. Three are the main activities of each node in this stage. Firstly, the node keeps expanding its local dataset, by adding in real time data about its current trajectory. In the exploitation stage, the *validation set*, which is used to evaluate the performance of the training process, is constituted by the last V samples of the trajectory of each node. Moreover, in the exploitation phase each nodes uses its LSTM model to issue a prediction about the cell in which it will be in h slots. The third and foremost activity of the exploitation stage is the training of the local model of the node, through a collaborative training, supported by all the nodes with which the given node comes in contact. The goal is to improve the accuracy of the local model over that which can be achieved by relying exclusively on local training. To this end, from the beginning of this stage,

each node partitions time into *rounds*, of duration equal to one or more slots. From the beginning of a round, and for all its duration, the given node (which we denote as *server node*) sends the coefficients of its local instance of the LSTM model to all nodes which are within its transmission range (which we denote as *client nodes*). Every client node that receives the model instance trains it using a subset of its own dataset.

Once each client node has completed the training of the received instance, it sends it back to the server node if it is still in range and if the round has not passed, and discards it otherwise. At the end of the round, and similarly to traditional Federated Learning, the server node combines the model instances received from clients during the given round, to build a meta-model which is used for issuing trajectory predictions, and for initiating a new training round. Note that each node has control of its data, and never shares its dataset with others. Instead, nodes share their model instances, thus providing support for protection of data confidentiality. Note also that the frequency of the training of the meta-model over the local dataset is a parameter which can be tuned and adapted to the specific setup considered. In particular, it depends on the duration of a round (in terms of number of slots), and it can be performed at the end of every round, or every n rounds.

We observe that the scheme we just described bears a close resemblance to classical FL algorithms. However, differently than classical FL, every node in the exploitation stage is at the same time playing the role of the *server* (or coordinator) for his own personal learning task, for which it is building its own local model, and of client node for other nodes' learning tasks. That is, each node in the exploitation stage (and thus with a substantial dataset) is available for training the models of all other nodes which come in contact. In this, they play a similar role as client nodes in FL, receiving a model instance from another node, and sending back to it an updated version of the received model instance. Differently than FL however, all client nodes participating in the training process in a round are all those nodes which the server node meets during the round. Moreover, each client is not required to be in contact with the server node for the whole round, as in FL. Instead, it may initiate the exchange with the server node at any time within the round, provided that the contact duration is long enough to allow for model exchange and training.

Among the many hyperparameters of our scheme, a key role is played by the total number of epochs for each training step. Our scheme adopts two different values for this parameter. When acting as client, and training other nodes' models,

we chose a maximum number of epochs equal to one. Indeed, though our scheme differs in important ways from traditional, centralized FL, they both share the same client-coordinator structure, for which it has been shown [20] that choosing a single epoch maximizes accuracy while minimizing training time. When a node trains its local model on its own local dataset instead, the total number of epochs is chosen in such a way as to avoid overfitting while maximizing accuracy.

A key aspect of our GL approach is represented by the strategy used by each server node to combine the model instances received by client nodes in order to produce an updated version of its local model instance (also denoted as *meta-model*). In what follows, we propose three different approaches to merge model instances received from neighbours and create a local model instance by using collaborative learning techniques. The performance of the three approaches will be then discussed in the numerical evaluation section.

It is important to stress that, in the considered setting, the strategies for the merging of model instances are the key factor influencing the system performance, due to the continuously changing set of neighbour nodes. For this reason, in this paper we concentrate on such strategies and on the corresponding model instance weighting approaches.

2.3.3 Decentralized Averaging (DFed Avg)

The *DFed Avg* approach is based on the way of combining models which is most frequently used in the literature for FL algorithms. With this approach, w^v is

Algorithm 2 DFedAvg
n_k is the number of samples in the local database of client k which have been used for training

function MERGEMODELS($\{w^{v,k}\}_k$, n_k)
$N \leftarrow \sum_{k \in K_j^v} n_k$
$w^v \leftarrow \sum_{k \in K_j^v} \frac{n_k}{N} w^{v,k}$
Return w^v

computed through a weighted sum over all clients model instances, in which the contribution of each client node is weighted by the proportion of the number of its samples involved in the training phase (n_k) over the total number of samples in the training phase (N). This approach gives more weight to a model instance which has a larger number of samples. The underlying idea is that to a larger amount of data

points used for training it corresponds a more accurate model.

2.3.4 Decentralized Powerloss (DFed Pow)

The *DFed Pow* approach weights each neighbor contribution to the meta-model according to a notion of similarity (in terms of roads and regions of the city covered) among the datasets of the server and of the client nodes. Specifically, the server node uses its local validation set to evaluate the loss value l_k of each of the updates of the model received by client nodes. As a loss function, we considered the well known categorical cross-entropy [11], which compares the probability vector given as the output of the model with a one-hot encoded vector representing the true class, and through a logarithmic formula computes the amount of loss. The result is a loss value which increases as the predicted probability diverges from the actual label. In computing the meta-model, each client contribution is proportional to the inverse of a function which grows exponentially with loss. Specifically, to the model update of each user k we assigned a weight given by $\frac{10^{-l_k}}{\sum_{k'} 10^{-l_{k'}}}$, where the normalization term at the denominator is a sum over all clients which returned a model in the given round.

Algorithm 3 DFed Pow
l_k is the loss of model update from node k (i.e. of $w^{v,k}$) over validation set of node v.

function MERGEMODELS($\{w^{v,k}\}_k$)
  for every node $k \in K_j^v$ **do**
    Compute l_k
    $P \leftarrow \sum_{k \in K_j^v} 10^{-l_k}$
    $w^v \leftarrow \sum_{k \in K_j^v} \frac{10^{-l_k}}{P} w^{v,k}$
  Return w^v

2.3.5 Decentralized Minimum Loss (DFed MinLoss)

Finally, in the *DFed MinLoss* approach, the server selects among the received model updates the one with the lowest loss value (computed as in DFed Pow), and it takes it as its local model instance. This approach originates from the observation that in DFed Pow the meta-model does not always perform better than the single model updates which compose it.

Algorithm 4 DFedMinLoss

function MERGEMODELS($\{w^{v,k}\}_k$)
  for every node $k \in K_j^v$ **do**
    Compute l_k
  Compute $k' = \arg\min_{k \in K_j^v} l_k$
  Return $w^{v,k'}$

2.4 Numerical Evaluation

In this section, we present the results of the numerical assessment of our framework for decentralized Gossip Learning. In order to implement a realistic scenario in our simulation experiments, we adopted the dataset of the Luxembourg SUMO Traffic (LuST) Scenario [10], consisting in measurement-based vehicular traces over 24 hours from the city of Luxembourg, with a typical topology common in mid-size European cities, and with realistic traffic demand and mobility patterns. In particular, we considered a square region of side 1050 m (2.1) in the city center, within which we assumed that predictions of vehicle trajectories are required. We partitioned such region into 49 square cells of side 150 m, compatibly with the case in which each cell corresponds to the coverage area of a MEC-enabled small cell base station. As in many real scenarios, vehicular traffic flows exhibit non-stationary properties which might affect the accuracy of our GL trained models. In order to limit such effects of the variations of traffic flows over time, we observed the performance of our scheme over a time interval of one hour (specifically, from 6 : 30 AM to 7 : 30 AM, where vehicular traffic is sufficiently intense and stationary). Such time interval has been chosen to be, on one side, long enough to allow our training framework to progress substantially, but short enough for the average pattern of vehicular traffic flows not to vary significantly. In the given region, on average about 300 vehicles are present at every time instant in the time interval considered, and every vehicle is in the range of about 60 other vehicles, of which on average only half are in the exploitation phase and are thus able to act as clients. For implementing the simulations of opportunistic communications among vehicles, we adopted the Omnet++ [27] framework, while Keras [8] was used for implementing our GL algorithm. We assumed vehicles sample their position in space every 5 s, i.e. at rate comparable to that common in many present day car fleet management applications. Moreover, we assumed to require a prediction about a vehicle's position in 10 seconds in the future. Such forecast horizon is of the same order of magnitude of those required in, e.g., predictive collision avoidance systems, or in MEC resource

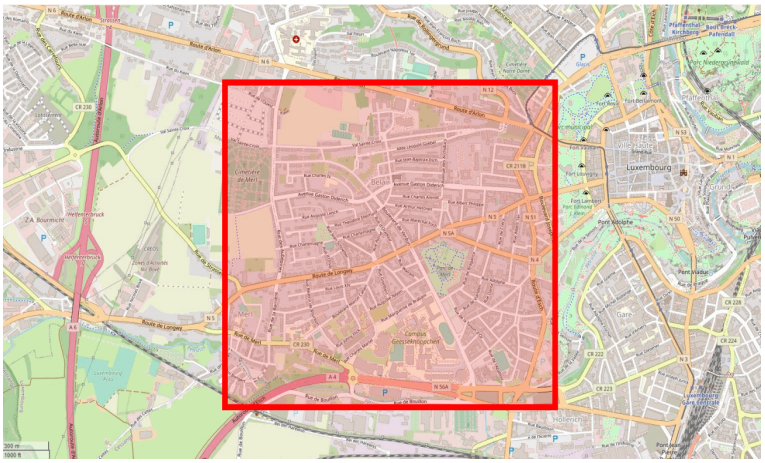


Figure 2.1: Map and road grid of Luxembourg City center. The area within the red square corresponds to the region considered in our simulations.

preallocation strategies. Our LSTM model input being composed by 24 steps, it requires at least the last two minutes of the trajectory of a car in order to issue a trajectory prediction. We considered each node performs model training over its entire local dataset, adopting a local batch size of 32 data points (coherently with the indications in [4] [19], and a 10^{-3} learning rate, as suggested in [12].

Given that in the scenario considered the average sojourn time of vehicles in the given region has been around 20 minutes, after several experiments for each node we have chosen for the initialization phase a duration of about half of this time, i.e. 9 m 30 s. Indeed in the given setting such a duration is, on one side, short enough to have a substantial amount of nodes in the exploitation stage at each point in time (about 50%), and thus contributing to our collaborative training. On the other, it is long enough to have a substantial training set at each node, and thus to allow the collaborative training to progress at a reasonable rate within the residual sojourn time of each vehicle. Given the relatively short duration of the exploitation phase, we have experimentally verified that the periodic training of the local model instance on the local database during the exploitation phase has no significant impact on model accuracy. Note however that these considerations are strictly related to the characteristics of the chosen scenario, and specifically, to the size and shape of the given region, as well as the mean node speed. In order to perform a first evaluation of our approach, we have assumed that the tasks of model training, of computing a meta-model, and of exchanging a model are instantaneous, thus modeling the case in which the limiting factor of our training framework is given by the way in which training and meta-model computing task are implemented, rather than by

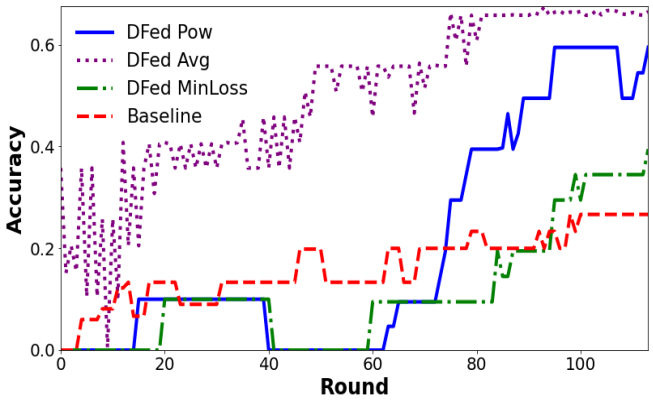


Figure 2.2: Mean accuracy versus iterations for our GL algorithm, for the three model merging schemes, and baseline model, with static test set, in the Luxembourg scenario.

their duration. In order to assess the performance of the training process, the test set consisted in the data points of the trajectory of each vehicle within the last 5 minutes of the vehicle's sojourn time within the given region. With such a choice for the test set (henceforth denoted as *static*), at each point in time during the exploitation stage the evaluation is done on data points about parts of the map which are generally far from those in which the node is located. This holds also, on average, for the dataset of the client nodes. Finally, in order to better appreciate in what measure our collaborative training improves over purely local training, we have considered this latter approach as as a baseline, and we have characterized its performance.

Fig. 2.2 and Fig. 2.3 show accuracy and loss of our GL schemes, averaged over the 30 cars in the area with longest sojourn time, for the static test set approach. As these figures show, one consequence of choosing a static test set is that on average the initial loss is high (and the accuracy low) for all of the three meta-model building approaches, and sometimes even lower than the baseline. The figures also show that these parameters keep on improving steadily across the iterations of our GL training scheme.

These results show also that despite every node enters the scenario with an empty dataset, a substantial improvement in model accuracy can be achieved within a relatively small amount of rounds. The two plots indicate that, when vehicles spend a sufficiently large amount of time in the scenario, GL schemes can collaboratively train models which achieve a substantially better performance with respect to their

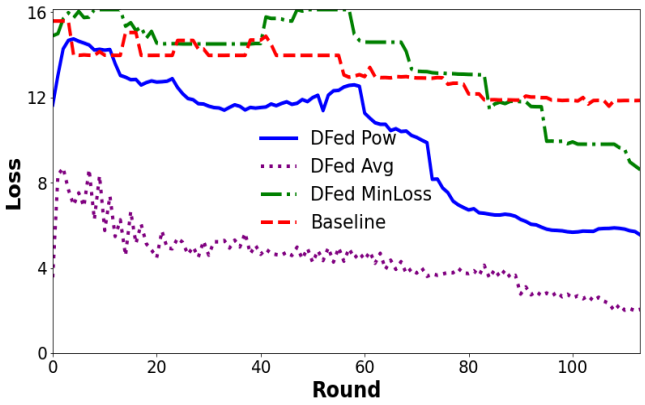


Figure 2.3: Mean loss versus iterations for our GL algorithm, for the three model merging schemes and baseline model with static test set, in the Luxembourg scenario.

initial, locally trained models, and over relatively short timespans. Moreover, the plots Fig. 2.2 and Fig. 2.3 seem to suggest that, as expected, the main potential factor contributing to such improvement is the progress in model training, i.e. the gradual inclusion over time in the trained model of an ever larger amount of data about trajectories in the given region.

As for the relative performance of the three approaches to meta-model elaboration, Fig. 2.2 and Fig. 2.3 show that weighting each contribution to the meta-model based on the relative size of the local training set, outperforms approaches based on loss estimation. This might be due to the fact that while the validation set (over which loss is evaluated in the *DFed Pow* and *DFed MinLoss* approaches) consists in the last V slots of a user's trajectory, and thus to a region of space very close to where the user will be in h slots, the test set is relative to a portion of the user trajectory which is (generally, except for the final part of the vehicle's trajectory) far enough from the current vehicle position (and from where it will be) to make the performance over the validation set not indicative of the actual prediction accuracy of the model. I.e., as the model evolves constantly to adapt its performance to the specific prediction task and to the context in which the prediction is formulated, it would make more sense to take as test set data points which are as much as possible related to that same spatio-temporal context.

In order to address this issue, in a new set of experiments, at each time slot t we have adopted as test set the data points relative to the time interval $(i - l, i + h)$ where l is the length of the LSTM input (24 slots), and h is the forecast horizon

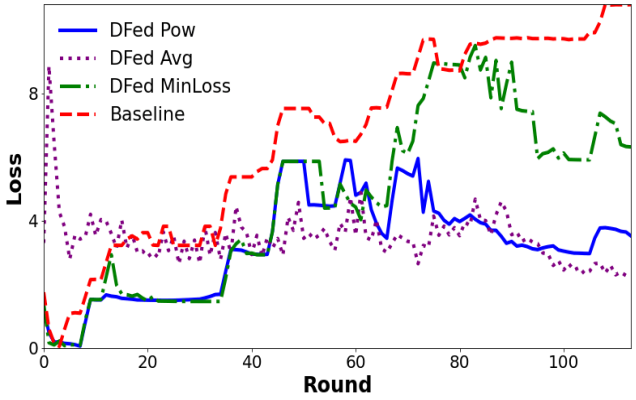


Figure 2.4: Mean loss versus iterations for our GL algorithm with a short size rolling test set, for the three model merging schemes, in the Luxembourg scenario.

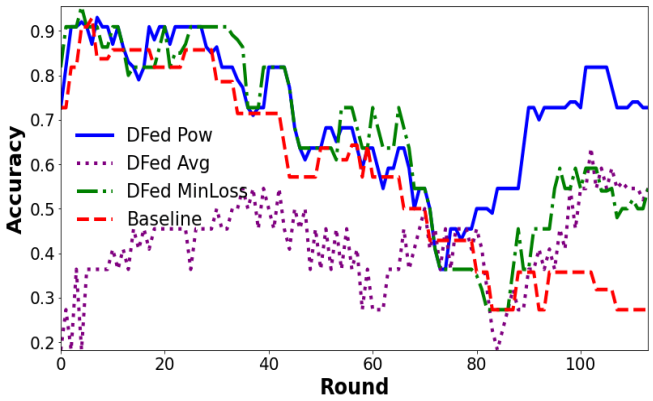


Figure 2.5: Mean accuracy versus iterations for our GL algorithm with a short size rolling test set, for the three model merging schemes, in the Luxembourg scenario.

(equal to 2 slots in our experiments). This should allow a more relevant assessment of model performance, as it is performed over the same segment of a vehicle trajectory over which the prediction is performed. As Fig. 2.4 and Fig. 2.5 show, when assessed with a rolling test set the performance (both in terms of accuracy and of loss) of the models trained with our GL approach is generally less pessimistic than with a static test set. In addition, it remains approximately constant as the number of rounds increases, if not for fluctuations which are due to spatial in homogeneities in node density, in the structure of the road grid, and to bursts of nodes arriving and leaving the considered region. Moreover, in this setting the loss-based meta model building approaches perform generally better than the strategy based on the relative size of the local dataset of client nodes, with the DFed Pow performing slightly better

than the DFed MinLoss. This is likely due to the fact that test set and validation set are now relative to contiguous regions of space and time interval, making loss measured over the validation set a more relevant indicator of the performance over the (rolling) test set. Note that improvement over the baseline becomes noticeable only after that a substantial amount of rounds has passed. This is due to the fact that the baseline algorithm has been obtained via training over a dataset which typically contains only a very limited number of cell transitions. Thus, it performs poorly whenever the node moves to another cell, an event whose likelihood grows with time, thus steadily worsening the performance of the purely locally trained model over time.

2.5 Conclusions and future work

The preliminary results presented in this paper suggest that the performance of nodes with too small/poor datasets, or short sojourn times might be improved by having nodes with better models spread them opportunistically, and let other nodes use such received models as starting point for the GL algorithm. We thus plan on expanding the approach discussed in this paper with other forms of model exchanges among vehicles, which can better enable vehicles with short sojourn times and/or small datasets to contribute significatively to the model sharing scheme. Moreover, we plan on performing a thorough assessment of our algorithms on a variety of other vehicular scenarios, and to characterize their convergence properties.

Bibliography

[1] Khadige Abboud, Hassan Aboubakr Omar, and Weihua Zhuang. Interworking of DSRC and cellular network technologies for V2X communications: A survey. *IEEE transactions on vehicular technology*, 65(12):9457–9470, 2016.

[2] Florent Altché and Arnaud de La Fortelle. An LSTM network for highway trajectory prediction. In *2017 ITSC*, pages 353–359. IEEE, 2017.

[3] Arash Asadi, Peter Jacko, and Vincenzo Mancuso. Modeling D2D communications with LTE and WiFi. *ACM SIGMETRICS Performance Evaluation Review*, 42(2):55–57, 2014.

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[5] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.

[6] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, pages 712–719, 2014.

[7] Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer, and Ernoe Kovacs. Building a big data platform for smart cities: Experience and lessons from santander. In *2015 IEEE International Congress on Big Data*, pages 592–599. IEEE, 2015.

[8] Francois Chollet. *Deep Learning with Python and Keras: The practical manual from the developer of the Keras library.* MITP-Verlags GmbH & Co., 2018.

[9] U Cisco. Cisco annual internet report (2018–2023) white paper, 2020.

[10] L. Codeca, R. Frank, and T. Engel. Luxembourg SUMO Traffic (LuST) Scenario. In *IEEE VNC*, pages 1–8, Dec 2015.

[11] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[12] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[13] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[14] N. Javaid, A. Sher, H. Nasir, and N. Guizani. Intelligence in IoT-Based 5G Networks: Opportunities and Challenges. *IEEE Communications Magazine*, 56(10):94–100, 2018.

[15] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE ITSC*, pages 399–404. IEEE, 2017.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. MLbase: A Distributed Machine-learning System. In *Cidr*, volume 1, pages 2–1, 2013.

[18] Puneet Kumar, Mathias Perrollaz, Stéphanie Lefevre, and Christian Laugier. Learning-based approach for online lane change intention prediction. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 797–802. IEEE, 2013.

[19] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[20] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[21] Peter Ondrúška and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proceedings of AAAI*, pages 3361–3367, 2016.

[22] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[23] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[24] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30:4424–4434, 2017.

[25] Hatem Ben Sta. Quality and the efficiency of data in "Smart-Cities". *Future Generation Computer Systems*, 74:409–416, 2017.

[26] Z. Sun and M. R. Nakhai. An Online Mirror-Prox Optimization Approach to Proactive Resource Allocation in MEC. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[27] Klaus Wehrle, Mesut Güneş, and James Gross. *Modeling and Tools for Network Simulation*. Springer Science & Business Media, 2010.

[28] Z. Xiao, P. Li, V. Havyarimana, G. M. Hassana, D. Wang, and K. Li. GOI: A Novel Design for Vehicle Positioning and Trajectory Prediction Under Urban Environments. *IEEE Sensors Journal*, 18(13):5586–5594, 2018.

CHAPTER 3

A Gossip Learning Approach to Urban Trajectory Nowcasting for Anticipatory RAN Management

Published in IEEE Transactions on Mobile Computing (TMC), Volume: 23, Issue: 5, Pages: 6288-6303.
Date: April 2023.
Publisher: IEEE.
Print ISSN: 1536-1233.

Abstract– In future radio access networks, machine learning (ML) based strategies for short-term forecasting of vehicular trajectories will be key for anticipatory resource allocation and management at the mobile edge. However, training ML models in a centralized fashion, over data collected from a massive heterogeneous and dynamic set of devices, poses significant scalability, reliability, and efficiency challenges, which are still open to date. In this paper, we look at the specific issue of scalable and resource-efficient training of ML models in a vehicular environment. To address such a challenge, we propose a new Gossip Learning scheme, i.e., a fully distributed, collaborative training approach based on direct, opportunistic model exchanges via wireless device-to-device (D2D) communications with no centralized support. Our approach is based on constantly improving each node's own model instance through knowledge transfer among nodes, and on different strategies for estimating the potential contribution of neighboring nodes to the training process at a node. Extensive numerical assessments on a variety of measurement-based dynamic urban scenarios suggest that our schemes are able to converge rapidly and provide sufficiently accurate forecasts of vehicle position for time horizons which are

typical of future 5G/6G dynamic resource allocation algorithms.

3.1 Introduction

Machine learning (ML) based techniques for vehicle trajectory nowcasting, i.e., short-term trajectory forecasting [44], play a central role in future 5G/6G vehicular networks as a key enabler of anticipatory radio access network (RAN) resource management strategies [28], as well as road safety algorithms for autonomous vehicles [27]. Several challenges, however, stand in the way of designing a scalable and efficient strategy for training an ML prediction model for vehicular trajectory nowcasting. The most prominent one is how to take advantage of the availability of large amounts of privacy-sensitive data distributed among users without exposing it to breaches. Indeed, training a prediction model in a centralized fashion would not only put a high privacy risk on the central server, but it would also be difficult to scale to large numbers of vehicles, and it would be inefficient from the viewpoint of RAN bandwidth utilization.

To overcome these shortcomings of centralized approaches, several distributed ML schemes have recently emerged [39, 22, 32]. Among these, Federated Learning (FL) [32] offers potentially better privacy protection than traditional approaches by substituting data sharing with model sharing. However, it relies on a central coordinating node that dispatches and aggregates models trained locally on each node. As such, it still bears all the main shortcomings of centralized architectures in terms of bandwidth utilization, scalability, and single point of failure. To address these drawbacks, several fully decentralized approaches have been proposed[35, 6, 17, 37]. Among these, a promising technique is Gossip Learning (GL) [35]. It is a fully decentralized version of FL, where individual nodes train, exchange, and merge models with neighbors without the need for a central coordinator. However, most of these existing solutions for decentralized model training mainly assume no node mobility and no churn, making them unsuitable for dynamical settings such as vehicular scenarios.

In this paper, we tackle the above-mentioned open issues for the specific problem of training an ML model for trajectory nowcasting in an urban setting. We propose a novel GL approach that allows new nodes in the area to build over, and benefit from, models elaborated by other nodes. Our approach does not require support from the infrastructure, nor a connected interconnection graph, and it allows contin-

uously improving the model and keeping it up to date with changes in the context. Moreover, it does not require maintaining a global state (such as in [31]), thus enabling better scalability. Our collaborative approach enables high-performing nodes to support those nodes whose model performs less well (e.g. because they do not encounter enough neighbors, or because they do not possess the data required to train a high-performing model by themselves).

Specifically, the main contributions of this paper are as follows:

- We propose a novel and fully distributed GL approach for highly dynamic settings, based on node cooperation and asynchronous communications, and on the combination of local model training with merging of models received via opportunistic exchanges.
- We present three practical model aggregation strategies, based on iterative weighted model averaging and on different estimators of the impact of each contribution on the accuracy of the model resulting from the aggregation.
- We characterize analytically its convergence properties, showing that our GL scheme converges to an optimal set of parameters under mild assumptions on the connectivity and regularization of the data.
- We assess the effectiveness of our GL strategies over measurement-based mobility traces against a set of relevant baselines, as functions of different urban settings and traffic configurations. Results suggest that our approach is effective and able to converge and adapt even in the presence of large variations in vehicle density and traffic patterns. They also suggest that the performance of our schemes is comparable (if not better, in some cases) to Federated Learning schemes.

The rest of this paper is organized as follows. Section 3.2 presents the system model, and in Section 3.3 our GL algorithms are described in detail. Section 3.4 presents the main results related to the convergence of our algorithms, and Section 3.5 is dedicated to their numerical assessment. Section 3.6 discusses the state of the art. Finally, conclusions and future directions of investigation are summarized in Section 3.7.

3.2 System model

We consider nodes moving on a road grid according to an arbitrary mobility model. We assume that each node knows its position at any time, and that nodes communicate in a peer-to-peer fashion using wireless technology (e.g., WiFi, BLE, or cellular D2D). When two nodes are in range of each other (and thus able to exchange information directly), we say that they are *in contact*.

Without loss of generality, we assume time to be divided into equal-sized slots, and let $t \in \mathbb{N}$ be the slots' label. We focus on a specific region of the road grid within which, at any slot t, each node tries to predict its own location h slots in the future. h is thus the *forecast horizon*, assumed to be equal for all nodes. Let $i \in \mathbb{N}$ denote the unique identifier of a node. We assume that each node entering the region possesses a *local dataset* $\mathcal{S}_i$, generally different in size and composition for each node, and composed by time series pertaining to its past trajectories in the region. Specifically, if with (x_t, y_t) we denote the node position at the beginning of the t-th slot, the s-th element of the local dataset is a time series $\{(x_t, y_t)\}_{t=t_s,\ldots,t_s+l_s}$ of size l_s by representing a trajectory, i.e. a sequence of positions over the plane over l_s consecutive time slots, starting at slot t_s.

3.3 A distributed Gossip Learning architecture

3.3.1 Model architecture

We assume that each node, in order to predict its own trajectory, employs a Long Short Term Memory (LSTM) neural network, a special kind of Recurrent Neural Network (RNN), widely used for vehicle trajectory prediction [3]. More generally, in the case of time series forecasting, when a sequence of input and output multivariant data is available, encoder-decoder LSTM models have shown to outperform other approaches for trajectory prediction, such as Kalman filtering [7] or support vector machines [20, 34, 23].

Specifically, the architecture of the LSTM model at each node is given by the concatenation of two LSTM layers (*encoder* and *decoder* respectively). At each time t, the encoder layer takes as input a trajectory over the last α time slots. The output of the encoder layer is a fixed-length vector that captures the trajectories' temporal structure. The decoder layer maps such a vector representation back to a

Symbol	Description
i	Node label
t	Time slot label
j	Round label
w_j^i	Array of coefficients of the local model instance of node i at the beginning of round j
ξ_j^i	Performance estimator of the model from node i at the beginning of round j
K_j^i	Set of nodes whose models have been received by node i during phase 1 of round j
β	Cutoff value
l_i	loss of the model received from node i
A	Total number of cells in the region
a	Cell label
$\mathcal{S}_i$	Local data set of node i
α	Number of input steps of the LSTM model
h	Forecast horizon (in n. of slots)

Table 3.1: Main notation used in the paper.

variable-length target sequence of h coordinate labels that describe the forecasted node position from time slot $t+1$ up to $t+h$.

The training process of such LSTM model takes place over a sequence of iterations (denoted as *epochs*). At each epoch, the parameters of the model are updated. We adopt a mini-batch gradient descent approach, in which the training data during one epoch is partitioned into two or more batches. This allows avoiding storing all training data in memory [5]. We employ the Adam optimizer [21] because of its computational efficiency and simplicity, as it requires little memory. It is well suited for problems with many parameters, and appropriate to cope with non-stationary objectives [21]. Nonetheless, our approach extends immediately to other optimizers and, more generally, to other ML architectures.

3.3.2 Gossip Learning algorithm

In what follows, we describe the training algorithm run by each node within the considered region of the road grid. Its goal is to generate more accurate models than those resulting from training exclusively on the local dataset of each node, without relying on any parameter server or centralized coordination function, and

exploiting only opportunistic communications during ephemeral contacts between nodes.

We assume that, in addition to a local dataset, each node entering the region is endowed with an instance of the encoder-decoder LSTM model. Such an instance is initialized randomly, and trained over the node's local dataset. In addition, it may incorporate a priori information on patterns of vehicular trajectories in the region. The composition of the local dataset at ingress time, and the coefficients of the LSTM model can be tuned to model settings with various degrees of availability of a priori knowledge about patterns of mobility in the region.

At every time slot, each node in the region feeds its local instance of the LSTM model with the last α samples of the time series representing the node's own trajectory over the last α slots, and it uses its local instance to issue a prediction about its position in the next h slots. In addition, it keeps expanding its local dataset by adding new data about its trajectory in the region as it unfolds over time.

Starting from ingress time, the training algorithm run by each node develops into a sequence of *rounds*. We assume that all rounds have the same duration, and are equal for all nodes, but they are not synchronized across nodes.

Each round is composed of three *phases*. During the first one, each node sends the coefficients of its local model instance to all nodes with which it comes in contact, and it receives the coefficients of their local model instance. As local instances change over time due to the training process, the version of the local instance received from each node is relative to the time at which the exchange occurs.

In the second phase, similarly to traditional FL, each node combines the model instances received during the first phase to produce an updated version of the local model instance (denoted as *meta-model*). Finally, in the third phase, each node trains the meta-model on its own local dataset to include data about the most recent part of the node trajectory, thus producing a new version of its local model instance. This third phase may occur every $m \geq 1$ rounds, where m depends on the rate at which the local dataset is enriched with new data points. Thus, the duration of each phase within a round and the frequency of the training of the meta-model over the local dataset can be tuned and adapted to the specific setup. The total number of epochs for each local training step is chosen in such a way as to balance between minimizing the risk of overfitting, and maximizing model accuracy.

As a consequence of mobility, nodes enter and exit the given region. Thus, the

population of nodes participating in a GL scheme is subject to *churn*. Indeed, as we assume that the trained model is of use only for nodes in the region, all nodes exiting the region discard the trained model and thus stop contributing to the GL scheme.

As evident in such a training strategy, each node never shares its data with others. Instead, nodes share their model instances, thus providing support for the protection of data confidentiality. No support from servers or from infrastructure-based communication technologies is required, and stable connectivity among nodes is not needed.

A key aspect of GL is the *model merging* strategy, which determines the coefficients of the meta-model as a combination of the corresponding coefficients of the models to be merged. The strategy for choosing the weights associated with every model to be merged is a key factor influencing the performance of the meta-models and thus of the whole training scheme. When GL is applied to dynamic settings, each node's context, and thus its neighbours (and the models received from them) constantly vary over time. In these conditions, many factors may impact the utility of the received models, in terms of their potential for generating a meta-model which improves over the existing local instance. Therefore, there are many possible ways in which these factors can be accounted for in choosing which model instances to combine, and thus in defining a specific merging algorithm.

In this work, we propose three different approaches to meta-model computation: *Decentralized Averaging* (DA), *Decentralized Powerloss* (DP), and *Location-Based Averaging* (LBA). Each corresponds to a different way to account, in the merging process, for the specificities of the environment in which the distributed model training occurs. They are all based on associating a *performance estimator* (which we denote with ξ) with each model instance which has to be merged, i.e., an estimate of the potential of the model to improve the performance of the merged model with respect to the existing local model instance. The performance estimator is used to compute the weight attributed to each model in the derivation of the merged model. These merging strategies differ in how the estimator is derived and updated.

Moreover, as such an estimate is absolute and not relative to the specific set of models to be merged, a cutoff value $0 \leq \beta \leq 1$ is introduced, so that only a fraction β of the largest contributions (in terms of weight used in the computation of the merged model) is used in the derivation of the merged model. By tuning

the cutoff value, it is thus possible to prevent poorly performing model instances from negatively affecting the accuracy of the merged model. An example is given by those settings in which nodes with high-performing models are relatively rare and sparsely distributed over the region, so that the vast majority of models to be merged are likely to perform poorly on the prediction task of the node. In settings where models not used in the merging process are not exchanged, tuning the cutoff value can also be beneficial to implement different tradeoff points between communication efficiency, rate of convergence of the training process, and model accuracy.

Decentralized Averaging (DA). The key idea behind this merging strategy is to attribute a weight to model instances based on the number of samples they incorporate, either directly (i.e. by training on the local dataset), or indirectly (i.e. by merging). Thus, in this strategy, weights are computed based only on the properties of the models to be merged, not on the data or the context of the node which performs the merge operation. That is, nodes in different parts of the scenario, with different local data sets, but with the same set of model instances to be merged produce the same merged model. Such a property, in an abstract scenario with full connectivity among all nodes, would bring the GL training process towards solving a single consensus problem, thus producing a single trained model with the same coefficients for all nodes. However, in realistic scenarios, spatial heterogeneity in the data being collected by nodes and fed to the training process, together with lack of full connectivity among nodes, typically result into a spatial heterogeneity in the models resulting from the GL training process, and thus into a form of local specialization of models. In the numerical section, we will assess the impact of these aspects on the performance of models trained via the DA strategy.

The details of the DA strategy are outlined in Algorithm 5. For every node i present in the region during round j, with w_j^i we denote the array of coefficients of the local model instance of node i at the beginning of round j, with estimator ξ_j^i. $w_j^{i,out}$ is instead the array of coefficients of the local model instance of node i at the end of the second phase of round j, with estimator $\xi_j^{i,out}$. Finally, K_j^i is the set of those nodes whose models have been received by node i during the first phase of round j, and $K_j^i(\beta) \subseteq K_j^i$ is the subset composed by those nodes in K_j^i whose models' estimator at round j, normalized, is above a cutoff value β. That is,

$$K_j^i(\beta) = \left\{ k | k \in K_j^i, \frac{\xi_j^k}{\sum\limits_{h \in K_j^i} \xi_j^h} \geq \beta \right\} \tag{3.1}$$

At the beginning of the first round, when the local model instance is trained on the local dataset, the performance estimator of the local model is set as the number of data points in the local dataset. After that, whenever a set of model instances is merged, the meta-model is computed as a weighted sum over all merged model instances. In this sum, the contribution of the $k-th$ merged instance is weighted by its associated performance estimator, normalized over the sum of all such parameters from all merged instances. As shown in Algorithm 5, if a cutoff β is applied, only those instances whose normalized weight is larger than β are included in the merging process. The performance estimator of the meta-model resulting from the merging operation is computed as a (normalized) weighted sum of the estimators of all the merged model instances, with the same weights as those used for the derivation of the meta-model itself. Whenever a model instance is trained again over the local dataset (i.e., during the third phase of a round), the performance estimator of the local model instance *after* the local training is the sum of its value *before* the local training, and the number of data points which have been added to the local dataset during round j.

Hence, in DA the performance estimator of a model instance is a loose estimate of the number of data points included in the model by training or merging, over the course of the training process. Such a choice is based on the intuition that models including larger amounts of data points are potentially more accurate. In reality model accuracy depends also in a complex manner on several other factors, such as the distribution of the data included in a model (which in our GL training algorithm is a function of the patterns of contacts and model exchanges among nodes, and thus of the features of mobility patterns themselves). In Section 3.5, we explore such interdependency by assessing the algorithm on various scenarios and mobility patterns.

Location-Based Averaging (LBA). The intuition behind the LBA strategy is that, in models used for nowcasting, affinity among models is related to affinity in prediction tasks, possibly because they are relative to the same area. Thus, the goal of GL with LBA is to train a set of models, each capturing the features of mobility

Algorithm 5 Decentralized Averaging at node i in round j

function MERGEMODELSDA($\{(w_j^k, \xi_j^k)\}_{k \in K_j^i}$, β)

$$w_j^{i,out} \leftarrow \sum_{k \in K_j^i(\beta)} \frac{\xi_j^k}{\sum_{h \in K_j^i(\beta)} \xi_j^h} w_j^k$$

$$\xi_j^{i,out} \leftarrow \sum_{k \in K_j^i(\beta)} \frac{(\xi_j^k)^2}{\sum_{h \in K_j^i(\beta)} \xi_j^h}$$

of a specific, predefined area within the region. Unlike DA (in which it is rather an emerging property), in LBA model specialization is thus explicitly accounted for in the training process. Specifically, in LBA the given region is partitioned into A cells. For each cell, each node (independently of whether it is located in that cell or not) trains a model which is meant to be used for trajectory predictions only when a node is in that cell. Thus, every node trains A models at the same time, maintaining for each a specific performance estimator. Similarly to DA, this estimator is related to how many data points of a specific cell have been included (by training and merging) in the model associated with that cell. In scenarios with enough heterogeneity in mobility patterns, this approach to distributed training of personalized (or, more specifically, location-specific) models should potentially enable higher accuracy levels, as it avoids combining models with little affinity among them.

The LBA merging strategy is outlined in Algorithm 6. With $a = 1, ..., A$, we denote the label of the a-th cell in the region. $\mathbf{w}_j^i = (w_{j,1}^i, ..., w_{j,a}^i, ..., w_{j,A}^i)$ denotes the array of the coefficients of the A local model instances of node i at the beginning of round j, one for each cell. Similarly, with $\boldsymbol{\xi}_j^i$ we denote the array of estimators associated with them. $K_j^i(\beta, a) \subseteq K_j^i$ is the subset composed by those nodes whose contribution to the merged model for cell a is above the cutoff β:

$$K_j^i(\beta, a) = \left\{ k | k \in K_j^i, \frac{P_{j,a}^k S_j^k}{\sum_{h \in K_j^i} P_{j,a}^h S_j^h} \geq \beta \right\} \tag{3.2}$$

where

$$P_{j,a}^k = \frac{\xi_{j,a}^k}{\sum_{a' \in 1,...,A} \xi_{j,a'}^k}$$

and

$$S_j^k = \frac{\sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^k}{\sum\limits_{h \in K_j^i} \sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^h}$$

For each cell, the performance estimator $\xi_{j,a}^{i,out}$ of node i's meta-model for cell a in round j is computed as the weighted sum of the estimators of the models to be merged. For any model to be merged from node k and cell a, the weight is the normalized product of two components. Similarly to DA, the first component $P_{j,a}^k$ is directly proportional to the estimator of node k's model, and thus to the relative number of data points incorporated in that model for cell a. The second component S_j^k is proportional to the total amount of data points included in all models of node k. The product $P_{j,a}^k S_j^k$ is thus an indicator of how well a model is estimated to perform in cell a with respect to the other cells. Indeed, the fact that, for a given node, a model for a given cell is not as good as that for the others might signal that the node has not been able to include enough information about trajectories in that cell. Note that at the beginning of the first round, for every cell a, the performance estimator equals the number of data points in the local dataset relative to that cell.

Algorithm 6 Location-Based Averaging at node i in round j

function MERGEMODELSLBA($\{(\mathbf{w}_j^k, \boldsymbol{\xi}_j^k)\}_{k \in K_j^i}, \beta$)

for every node $k \in K_j^i$ **do**

$$S_j^k \leftarrow \frac{\sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^k}{\sum\limits_{h \in K_j^i} \sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^h}$$

for every cell a **do**

for every node $k \in K_j^i$ **do**

$$P_{j,a}^k \leftarrow \frac{\xi_{j,a}^k}{\sum\limits_{a' \in 1,\ldots,A} \xi_{j,a'}^k}$$

compute $K_j^i(\beta, a)$ (eq. 3.2)

$$w_{j,a}^{i,out} \leftarrow \frac{\sum\limits_{k \in K_j^i(\beta,a)} P_{j,a}^k S_j^k w_{j,a}^k}{\sum\limits_{h \in K_j^i(\beta,a)} P_{j,a}^h S_j^h}$$

$$\xi_{j,a}^{i,out} \leftarrow \frac{\sum\limits_{k \in K_j^i(\beta,a)} P_{j,a}^k S_j^k \xi_{j,a}^k}{\sum\limits_{h \in K_j^i(\beta,a)} P_{j,a}^h S_j^h}$$

Decentralized Powerloss (DP). In this strategy, the weights associated with each model to be merged are derived from a measure of the model's performance

over the most recent trajectory of the merging node. The intuition behind this is that such a measure is likely to be correlated to the model's performance in the near future along the merging node's trajectory. As a consequence, in DP nodes exchange models but not their performance estimators, as they are computed by the node which receives those models, based on its own recent trajectory. Therefore, performance evaluators have a very narrow validity (i.e., limited to the merging node and to a specific time) and they are recomputed whenever they are needed, i.e. just before the merging operation.

The DP merging strategy is detailed in Algorithm 7. In round j at node i, the performance measure of a model to be merged received from node $k \in K_j^i$ consists of its loss, denoted as $l_{j,k}^i$. Such loss is computed over the merging node's validation set, composed by the last V samples of its trajectory. The weight associated with each model to be merged is then computed as a (normalized) logarithmic function of the loss, as this choice increases the weight of those models with small losses. Thus, the expression of $K_j^i(\beta) \subseteq K_j^i$ for the DP strategy, when a cutoff value is applied to merging weights, is:

$$K_j^i(\beta) = \left\{ k | k \in K_j^i, \frac{|\log_{10} l_{j,k}^i|}{\sum_{h \in K_j^i} |\log_{10} l_{j,h}^i|} \geq \beta \right\} \tag{3.3}$$

In general, the choice of the size of the validation set is a compromise between getting an estimation which, on the one side, has low noise, and on the other side, is representative of the actual performance of the model on the merging node trajectory in the near future. As a loss function, in the present work, we have adopted the mean squared error (where the error is the distance between the actual position of the merging node and its predicted position). However, the DP strategy is very general, and it applies to any other type of loss function.

Being based on performance metrics which are specific to each merging node, our GL scheme with the DP merging strategy implements model personalization. Indeed, it produces a model which is generally different at each node, being tightly related to the context (the road segment in which the car is located at a given point in time) and the specific prediction task of each node.

Algorithm 7 Decentralized Powerloss at node i in round j

function MERGEMODELSDP($\{w_j^k\}_{k \in K_j^i}$, β)

compute $K_j^i(\beta)$ (eq. 3.3)

for every node $k \in K_j^i(\beta)$ **do**

Compute $l_{j,k}^i$

$$w_j^{i,out} \leftarrow \frac{\sum_{k \in K_j^i(\beta)} |\log_{10} l_{j,k}^i| w_j^k}{\sum_{h \in K_j^i(\beta)} |\log_{10} l_{j,h}^i|}$$

3.4 Convergence Properties

In this section, we show that under some mild assumptions on the system, loss function and network topology, our GL training scheme converges. We first formally define the optimization objectives and key assumptions, which the convergence analysis will follow. For the derivations, we refer to the DA merging strategy; however, the results can be extended to other merging strategies with some marginal modifications.

The key idea underlying the derivation of the proof is based on defining an objective function for the training process, which is a function of the local loss function as well as of the difference, at each node, between the local model and the models received from neighbors. Indeed, a decrease over time of these two components indicates, on one side, that the training process is progressing, and on the other, that nodes are successful in sharing the learned knowledge with their neighbors. Then we show that, when the aforementioned mild assumptions are satisfied, the objective function decreases as the iterations progress.

3.4.1 Notation and assumptions

As already stated, the goal of collaborative model training schemes such as GL is to let nodes jointly improve their models by leveraging both their local datasets and similar data available in the neighbourhood, where a time-varying connectivity graph defines neighbourhoods. Let us consider a time interval $[1, J]$ of duration equal to J rounds, and let us assume for simplicity the starting time of each round to be the same for all nodes. Let $\mathcal{N}$ denote the set of nodes (of cardinality N) which are present in the region in at least one of the slots of the given time interval. Each node $i \in \mathcal{N}$ has a local data distribution μ_i over the space of possible trajectories within the region. It has a local data set $\mathcal{S}_i$ composed of trajectories drawn randomly from μ_i. Each trajectory has a probability p_d (same for all trajectories) of being included

in the local dataset. $\forall i$, we assume μ_i not to change over time, and $\beta = 1$.

In general, at every round, as the training process evolves, the coefficients of the models associated with every node in the region change. Let $(w_j^i)_{j=1}^J$ be the sequence of iterations generated by the GL algorithms running for J rounds from an initial point $w_0^i \in \mathbb{R}^p$. The goal of node i is to learn a model instance whose coefficients w_J^i minimize the expected loss $\mathbb{E}_{z\sim\mu_i}[l(w_J^i; z)]$ after J iterations, where $l(w_J^i; z)$ denotes the mean loss of w_J^i evaluated over every point of the trajectory z, for any choice of the loss function.

Given that the local model instance of a node is trained over its local dataset by using LSTM and Adam optimizer, the goal of the local training phase is to select those model parameters which minimize loss through Stochastic Gradient Descent [3]. Specifically, if we denote with $w_j^{i,loc}$ such a model instance, we have

$$w_j^{i,loc} = \arg\min_{w\in\mathbb{R}^p} \mathcal{L}_i(w; \mathcal{S}_i)$$

$\mathcal{L}_i(w; \mathcal{S}_i)$ is the *local loss function* for the i-th node:

$$\mathcal{L}_i(w; \mathcal{S}_i) = \frac{1}{s_i} \sum_{z\in\mathcal{S}_i} l(w; z) + \lambda_i \|w\|^2 \tag{3.4}$$

where $\lambda_i \geq 0$ is a regularization parameter, and s_i is the cardinality of $\mathcal{S}_i$. In our gossip learning scheme, nodes use information from neighbours to supplement their data through several iterative rounds. We formalize the time-varying model by denoting $G^j = (\mathcal{N}, E, \mathbf{X}^j)$ as a weighted connected graph over the set of nodes, where $E = \mathcal{N} \times \mathcal{N}$ the set of all potential edges between the nodes and $\mathbf{X}^j \in \mathbb{R}^{n\times n}$ is the non-negative weight matrix at round j, where the specific merging strategy is determined the weights. In this model, only the weights $\mathbf{X}^j$ change with time, where $X_{ik}^j > 0$ if and only if $k \in K_j^i$. At round j the two nodes i, k are connected, and their local models and data are used in each other updates. There are two factors influencing how the values of X_{ik}^j change from one round to the other:

- Dynamic changes in the topology. They are due to node mobility, and they are out of the control of the learning algorithms.
- Changes that are caused by the way each node merges the models received from its neighbours. Weights X_{ik}^j are specific to the merging strategy, and they may depend on w_j^i. Note that the three averaging algorithms differ in

computing X_{ik}^j.

Let $\mathbf{w}_j = (w_j^1; \ldots w_j^i; \ldots w_j^N) \in \mathbb{R}^{N \times p}$. The overall goal of our gossip learning schemes is to find an array $\mathbf{w}_J$ of models, one for each of those nodes who spend in the region at least one round in the interval $[1, J]$, which minimizes a given system objective function $Q^J(\mathbf{w}_J)$.

The expression of the overall objective function $Q^j(\mathbf{w}_j)$ at round $j \in [1, J]$ can be written as

$$\begin{aligned} Q^j(\mathbf{w}_j) &= \sum_{i \in \mathcal{N}} Q^{i,j}(\mathbf{w}_j) \\ &= \sum_{i \in \mathcal{N}} \left[Q_{p2}^{i,j}(\mathbf{w}_j) + \nu Q_{p3}^{i,j} \left(\sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) \right], \end{aligned} \tag{3.5}$$

$\nu > 0$ is a trade-off parameter that is used to balance between the minimization of the local loss function and that of differences in model coefficients among nodes. The functions $Q_{p2}^{i,j}$ and $Q_{p3}^{i,j}$ are defined below.

Model merging (phase 2): The objective function for node i at the second phase of round j is

$$Q_{p2}^{i,j}(\mathbf{w}_j) = H\left(w_j^i - \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) \tag{3.6}$$

In this phase, each node aims at minimizing the difference between the coefficients of its local model and those of a weighted average of the model from neighbouring nodes (i.e., those of the merged model). For any vector x, $H(x)$ denotes the sum of the absolute value of the elements of x.

Local training (phase 3): Let $w_{j,p2}^i = \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k$. In the third phase of a round, at each node i the model is trained over the local dataset in a way which aims at minimizing the objective function

$$Q_{p3}^{i,j}(w_{j,p2}^i) = \mathcal{L}(w_{j,p2}^i; \mathcal{S}_i). \tag{3.7}$$

Thus, from (3.6) and (3.7), the expression of the objective function for every node i is

$$Q^{i,j}(\mathbf{w}_j) = H\left(w_j^i - \sum_{k \in \mathcal{N}} X_{ik}^j w_j^k \right) + \nu \mathcal{L}_i \left(\sum_{k \in \mathcal{N}} X_{ik}^j w_j^k; \mathcal{S}_i \right) \tag{3.8}$$

This sum has two distinct components, given by the local losses and the differences between local models. The objective in each round is thus to minimize the weighted

sum of these two components.

We prove in the rest of this section that our DA algorithm converges by showing that the objective function gets closer to the optimal value after every round.

We make the following assumptions on the loss function, which are standard in the analysis of coordinated learning methods in [43, 4].

Assumption 1. For any $w \in \mathbb{R}^p$, the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ is convex in all rounds and $\forall i$.

Proposition 1. *When Assumption 1 holds, in any round j, $Q^j(\mathbf{w}_j)$ is convex.*

The proof proceeds in the same way as in [4].

Assumption 2. $\forall i$, for any $w \in \mathbb{R}^p$, the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ has L_i^{loc}-Lipschitz continuous gradient in all rounds.

Assumption 3. $\forall i$, for any $w \in \mathbb{R}^p$, there exists a $\sigma_i > 0$ such that the local loss function $\mathcal{L}_i(w; \mathcal{S}_i)$ is σ_i-strongly convex.

Assumption 3 implies that $Q^j(\mathbf{w}_j)$ is σ-strongly convex with $\sigma \geq \nu\sigma_i > 0$. That is, for any $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^{N\times p}$,

$$Q^j(\mathbf{w}') \geq Q^j(\mathbf{w}) + \bigtriangledown Q^j(\mathbf{w})(\mathbf{w}' - \mathbf{w}) + \frac{\sigma}{2}\|\mathbf{w}' - \mathbf{w}\|^2, \tag{3.9}$$

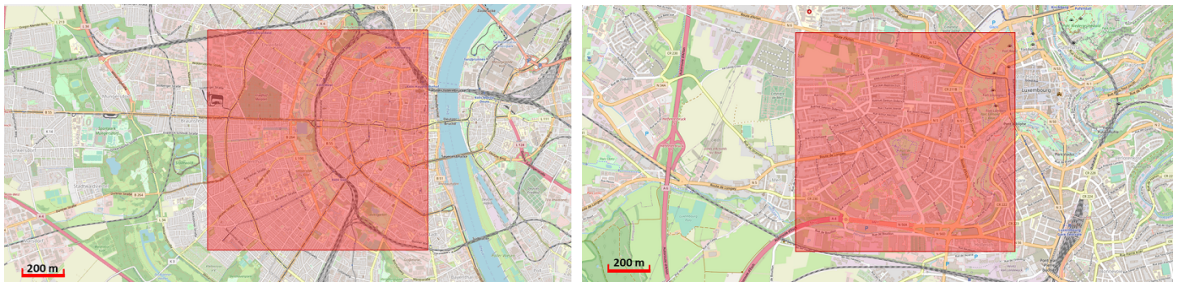
$\forall i$, let $L_i^j = (1 + \nu L_i^{loc})$, $\alpha_i = 1/L_i^j$, and

$$L_{max} = \max_{i\in\mathcal{N}, j\in[1,J]} L_i^t.$$

A desirable property of a GL training strategy is that the objective function decreases as the iterations progress. The following result, which is the main result of GL convergence, states that when assumptions 1 to 3 hold, our GL scheme with DA merging strategy converges.

Theorem 1. *Let Assumptions 1 to 3 hold. For $J > 0$, let $(\mathbf{w}_j)_{j=1}^J$ be the sequence of iterates generated by the GL training algorithm with DA merging, running for J iterations from an initial point $\mathbf{w}_0 \in \mathbb{R}^{N\times p}$. Then $\forall i$ there exist a $j' \in [1, J]$ such that, if we set $\mathbf{w}_*^{i,j'} = \arg\min_{\mathbf{w}\in\mathbb{R}^{N\times p}} Q^{i,j'}(\mathbf{w})$, we have:*

$$\begin{gathered}\mathbb{E}[Q^{i,J}(\mathbf{w}_J) - Q^{i,j'}(\mathbf{w}_*^{i,j'})] \leq \\ \left(1 - \tfrac{\sigma}{nL_{max}}\right)^{J-j'} \mathbb{E}[Q^{i,j'}(\mathbf{w}_0) - Q^{i,j'}(\mathbf{w}_*^{i,j'})]\end{gathered} \tag{3.10}$$



(a) Cologne (Innenstadt) (b) Luxembourg City

Figure 3.1: Map and road grid of the considered scenarios, with the region of interest in red.

For the proof, please refer to Appendix 3.8.1.

3.5 Numerical Assessment

To assess the performance of our GL approach numerically, we considered different measurement-based mobility datasets relative to different cities, different areas within a city, and various time intervals during the day. Specifically, a first setup is based on the LuST dataset [10], consisting of measurement-based vehicular traces from Luxembourg City, covering a time interval of 24 hours. The second scenario is based on the TAPAS Cologne dataset [40], again a measurement-based set of vehicular traces covering the greater urban area of Cologne for a whole day. 3.1 shows the position and size of the regions in Cologne and Luxembourg, respectively, within which our framework was assessed. In both scenarios, the region is a square of side 1 km, covering a large fraction of the city center. We used Keras [9] to implement our GL algorithms, SUMO [29] for vehicular mobility simulation, and the Omnet++ framework [42] for opportunistic communications among vehicles. Unless otherwise specified, we considered a slot duration of one second (typical of the sampling frequency of many present-day car fleet management applications) and a round duration of 15 s, equal for all nodes. Since in both scenarios, the average vehicle speed is 11 m/s (39.6 km/h), the chosen slot duration ensures that, on average, the sets of a given node's neighbors in consecutive rounds differ significantly. At the same time, such a round duration is short enough to allow the node to capture in its local model changes in the context due to mobility. Within each round, we assumed the second and third phases of our GL schemes to take an amount of time that is negligible with respect to that required by the first phase. These assumptions are based

on the fact that the time required by local training is negligible, given the small size of the model (almost 70 KB in our case) and of the local dataset. Moreover, model merging is not a computationally intensive task, consisting of a weighted sum of the model parameters. We considered a forecast horizon of 5 s, compatible with such applications as predictive collision avoidance systems, MEC processor reservation strategies, and 5G beam steering and resource allocation strategies [30].

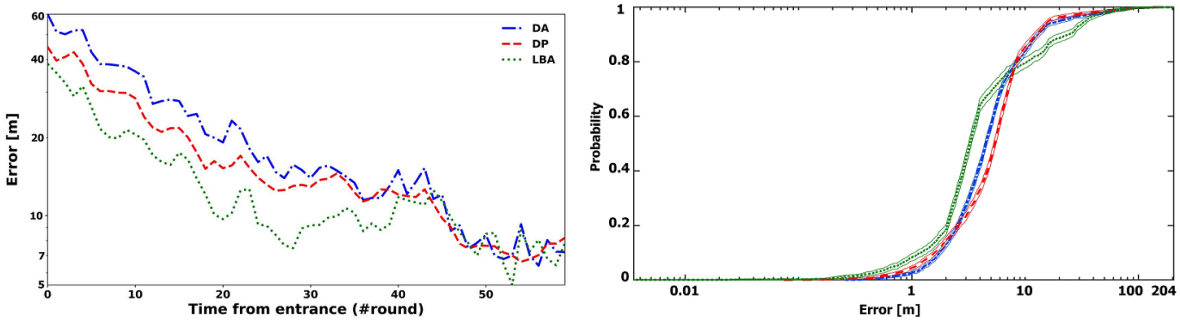
The input of the LSTM model is composed of 12 steps, with a gap of 5 s between two consecutive steps. Thus, our LSTM model requires at least the last minute of the trajectory of a car to issue a trajectory prediction.

For local training, we adopted a mini-batch Gradient Descent approach, with batches of size 32 (as indicated in e.g., [5]), and a 10^{-3} learning rate, as suggested in [14]. Unless otherwise specified, we assume that nodes merge all received model instances during a round. The values of such model hyperparameters as the number of neurons (50), the batch size, and the number of input time steps have been tuned based on an extensive set of simulations.

To determine the local dataset of each vehicle when entering the region, for each scenario (and for each time interval), we have built a *scenario database*, consisting of past trajectories within the given region, except those taking place in the same time interval considered in our experiments. Indeed, as both datasets are relative to a single 24 h period, this choice minimizes the probability for users to have in their local dataset the same trajectory they are taking in the considered time interval. To each vehicle entering the region of interest during the given time interval, we have assigned a local dataset, different for each node, and obtained by sampling uniformly at random the scenario database. As a result, on average, in both scenarios, each vehicle's local dataset contains data corresponding to about 5 minutes of trajectories. These choices have been made, on one side, to avoid the case of nodes having no local dataset at their ingress in the region of interest. Indeed, such a "clean slate" scenario would not represent realistic settings where all nodes possess at least some relevant data. On the other side, collaborative training schemes like GL make sense when nodes are not able to achieve by themselves (i.e. with only local training) a satisfactory level of accuracy because their local dataset is not large enough. For the LBA strategy, deciding on how to partition the region has a strong impact on its performance. After an extensive empirical evaluation, in both of the considered scenarios we defined for the LBA strategy a partition consisting of 9 square cells of side 333.3 m. This choice has been a compromise between, on the one side,

Name	City	Time	Mean sojourn time	Tx radius
Lux rush hour	Luxem-bourg	7:00-7:30	14 min 8 s	150 m
Lux off-peak	Luxem-bourg	16:00-19:00	11 min 12 s	150 m
Cologne off-peak	Cologne	14:00-17:00	4 min 4 s	450 m

Table 3.2: Scenarios considered in our experiments.



(a) Mean error versus time from entrance

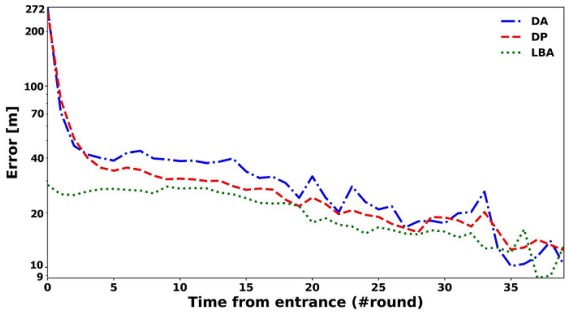
(b) Empirical CDF of the error for the DA, DP and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 50 and 60 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.

Figure 3.2: *Lux rush hour* scenario.

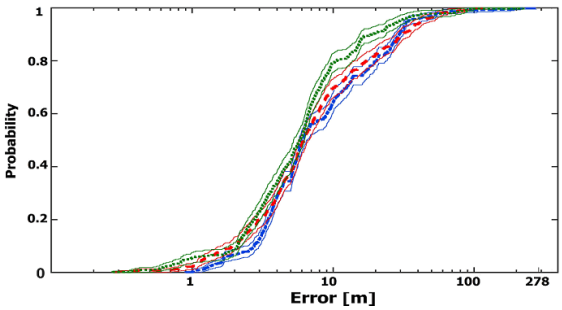
having trajectories that are as homogeneous as possible within the same cell, and the fact that (for the same size of the local dataset) a finer partition would reduce the portions of the local data set associated with each cell. As we have verified experimentally, the latter has a negative impact on the quality of the model trained by a node as it enters the region, and thus on the speed of convergence of the training process.

In addition to our schemes, we have considered the following baseline approaches:

- *Centralized Federated Learning (FL)* (in the version described in [32]), applied to the same two stages LSTM model trained by our schemes. In FL, a parameter server orchestrates the various algorithmic stages and coordinates all the participating nodes. For a fair comparison with our schemes, we have assumed that at any time slot t, the dataset available to the FL server coincides with the union of the local datasets of all the nodes which have spent at least one

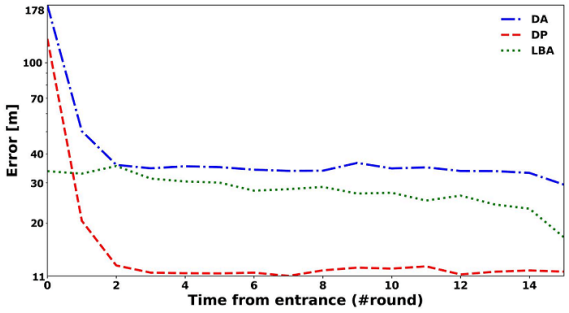


(a) Mean error versus time from entrance

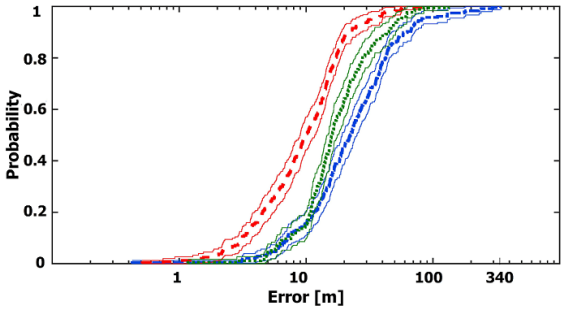


(b) Empirical CDF of error for the DA, DP, and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 30 and 40 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.

Figure 3.3: *Lux off-peak* scenario.



(a) Mean error versus time from entrance



(b) Empirical CDF of error for the DA, DP, and LBA algorithms. Each data point is relative to a single vehicle in the scenario and averaged in the time interval between rounds 12 and 16 from the entrance. Thin lines delimit 95% confidence intervals for the CDF.

Figure 3.4: *Cologne off-peak* scenario.

time slot in the given region, from the beginning of the GL scheme up to time slot t. Namely, for nodes that have exited the region by time t, we consider the local dataset at exit time. For all the others, we consider the local dataset at time slot t. All the model meta-parameters have been set to coincide with those of our GL scheme. We assumed the rounds of the FL scheme to be the same as those of our GL scheme. Again, for fairness of comparison, at every round, we assumed random client subsampling, with an average number of selected clients coinciding with the average number of nodes that each node comes in contact with during a round.

- *Local training*, in which each node trains the two-stage LSTM model only on its local dataset, with no exchange of data or models with neighboring nodes or a server. Such training is performed at the node ingress in the given region. In addition, at regular intervals (whose duration is equal to that of a round), the local model is re-trained over the local dataset, which keeps increasing as the node moves in the given region.

- *DFed Pow*[11]. It is a fully distributed learning scheme that inherits several features from FL schemes. Specifically, in each time slot, each node with its neighbors constitutes a federation (in the same sense as in classical FL schemes) aimed at training the given node's model. Thus, the given node plays the role of the parameter server, whereas its neighbors (who generally change from one slot to the next) play the role of clients. At every time slot, each node sends its model to all neighbors, who train it on their local dataset and send it back, to be merged with those sent by all other neighbors. Thus, in this scheme, there are as many federations as nodes.

- *Dead Reckoning (DR)*, in which each node forecasts its trajectory by extrapolating over its current position, velocity, and direction.

For our experiments, we have considered three different settings. As shown in Table 3.2, they differ in terms of the city within which the given region is located (Luxembourg, Cologne), the start time and the duration of the considered time interval. The first scenario (denoted as *Lux rush hour*) is relative to the time interval 7:00 AM - 7:30 AM in the Luxembourg City region. This corresponds to a rush hour, with a high density of vehicles (for an average of about 300 vehicles in the given region). The second scenario (*Lux off-peak*) is relative to the time interval

4:00 PM - 7:00 PM in Luxembourg City, and it is characterized by much lower traffic intensity. In both these scenarios, the transmission radius has been set to 150 m (e.g. typical of DSRC in urban environments [1]). The third scenario (*Col off-peak*) is relative to the 2:00 PM - 5:00 PM time interval in Cologne city. It is characterized by a much lower density of vehicles and a shorter mean sojourn time with respect to Luxembourg. To enable effective opportunistic model exchanges in such a low-density scenario, we have assumed a transmission range of 450 m, e.g., compatible with BLE version 5 [38].

3.5.1 Performance from entrance time

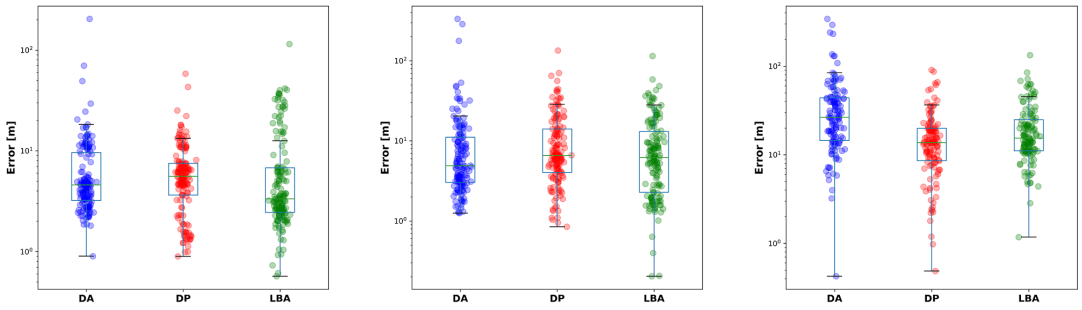
One of the main performance metrics of our algorithms is the *mean error* at the t-th time slot, defined as the distance between the forecasted and the actual position, averaged across all vehicles present in the scenario during that slot. In the first set of experiments, we have characterized the evolution of the mean accuracy achieved by the model instance of each node as a function of the time spent by the node in the region. Indeed, this measures how quickly our GL schemes improve the performance of a model instance over that achievable by training it exclusively over a local dataset. Fig. 3.2 to Fig. 3.4 show the mean error as a function of the vehicle sojourn time for the three scenarios considered. In each setting, results have been averaged across the whole time interval.

As the figures show, despite the short time from bootstrap, in all scenarios, nodes are able to achieve high levels of accuracy after spending only a few minutes in the region. In the Cologne scenario, the mean error decays more rapidly than in the Luxembourg scenarios, because the larger transmission radius in the Cologne scenario brings faster exchanges of models among nodes in the region. At the same time, the shorter mean sojourn time brings the accuracy to stabilize to a higher value than in the two Luxembourg scenarios, as the GL algorithms have less time to progress further.

Note that the values of accuracy achieved by each algorithm have been evaluated on a very conservative worst-case scenario, in which each node has only a small initial dataset, and in which our scheme has been running for at most three hours. The good performance of our schemes in these conditions in spite of these assumptions suggests that in settings where our schemes have been running for longer periods, their performance is likely to be better than predicted by the reported experiments.

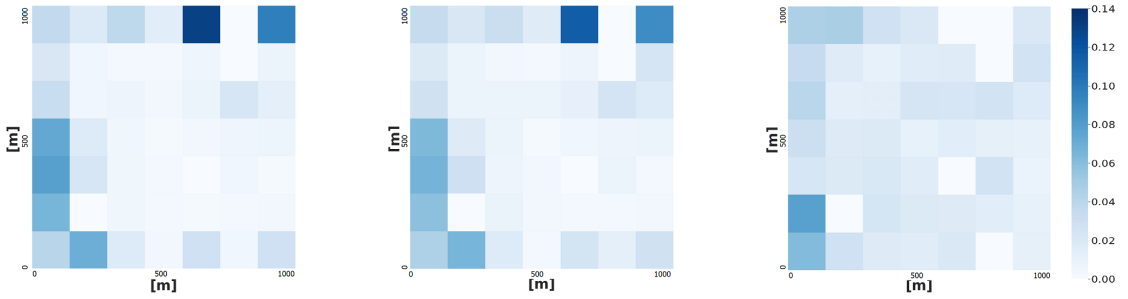
Our results show that, in general, the DP merging strategy performs better than DA, both in terms of convergence speed and mean error, across all experiments. Indeed, while DA's model quality estimate is not directly related to a model's performance at a given point of the region, that of DP is instead a function of each merging model's performance over the most recent part of the given node's trajectory. It is thus close to the performance that the merging model would deliver on the trajectory of the given node in the near future. As for the LBA strategy, results suggest that it is superior to the other two strategies when nodes spend a "long enough" period of time in each cell (as in the Luxembourg scenarios). In the Cologne scenario, instead, as the transmission radius is larger than the side of a cell, every user on average receives and merges model instances from neighboring cells. This averages out models across cells, destroying the ability of LBA to accurately model those mobility patterns which are specific to each cell, and thus the performance advantage of LBA over the other two GL algorithms.

From the plots of the CDFs of prediction error in Fig. 3.2 to Fig. 3.4, it can be seen that in each scenario, and for all merging strategies, the error distribution is heavily skewed towards values higher than the mean. This is more evident in Fig. 3.5, which shows, for each of the three scenarios and time intervals, a scatter-box plot of error. This figure shows how the vast majority of the outliers lie in the upper part of the logarithmic scale. Thus, in these scenarios, the mean error is heavily influenced by a few poor performers. Fig. 3.6 shows the spatial distribution of the error for the DA, DP, and LBA algorithms in the Lux off-peak scenario, averaged over a given time interval. The figure shows that the error is indeed larger at the borders of the considered region. Indeed, border areas contain many nodes that just entered the RZ, and that did not have the time to perform many iterations of our GL scheme. Thus, not only their model is likely to perform worse than average, but also their contribution to the improvement of the model of other nodes is likely to be marginal. This slows down the improvement of model accuracy for those nodes whose trajectory is mostly contained within border areas, who end up collecting few or no high-quality models from neighbors. In another set of experiments, we investigated how our GL algorithms evolve over time from node entrance in the region. To this end, we have tracked the evolution over time of the *weight ratio*, i.e. of the ratio between the weight used in the merging task for the local model, and the average of those attributed to models received from other nodes. Indeed, such a ratio at a given slot t indicates how much the distributed learning process weights



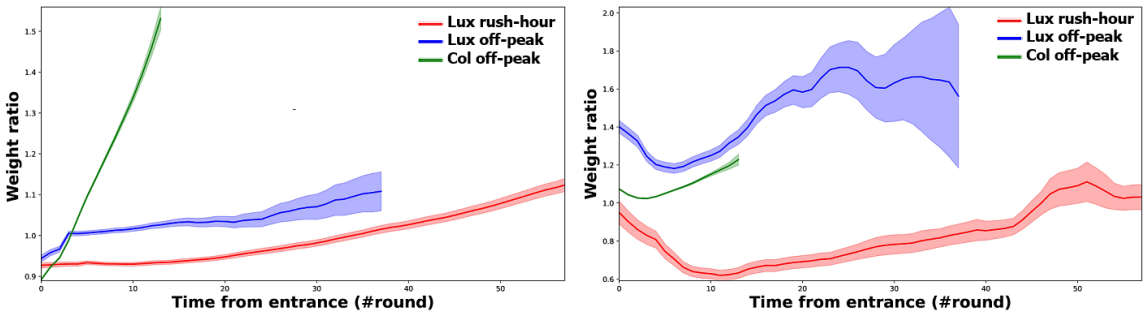
(a) Lux rush hour scenario (b) Lux off-peak scenario (c) Cologne off-peak scenario

Figure 3.5: Scatter-box plot of error for the DA, DP, and LBA algorithms. Each point is the error of a single vehicle, averaged in the time interval between rounds 50 and 60 (Lux rush hour scenario), between rounds 30 and 40 (Lux off-peak scenario), and between rounds 12 and 16 (Cologne off-peak scenario) from node entrance.



(a) DA (b) DP (c) LBA

Figure 3.6: Spatial distribution of error for the DA, DP, and LBA algorithms, averaged across vehicles and in the time interval between rounds 30 and 40 from node entrance, in the Lux off-peak scenario.



(a) DA (b) DP

Figure 3.7: Local models' weight (coefficient) ratio versus time from the entrance for the DA and DP strategies, in the three considered scenarios.

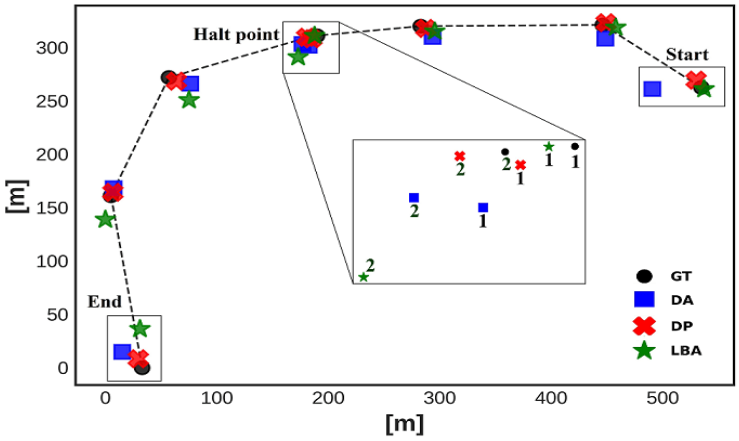


Figure 3.8: Sample trajectory predicted by the DA, DP, and LBA algorithms versus the ground truth (GT) for a vehicle in the *Cologne off-peak* scenario. The time interval between two consecutive points is 15 s.

the knowledge acquired until t versus the knowledge that it could acquire from neighbors. From Fig. 3.7(a) we can see that, as expected, as nodes spend more time in the region, the weight ratio (i.e. the weight given to the local model) for the DA algorithm generally increases, in all of the three scenarios. However, this increase is slow in the Luxembourg scenarios, and a bit faster in the Cologne scenario. This is because in the latter, a larger transmission radius and mean node speed account for a higher rate of contacts among nodes and thus a faster progression of the training scheme. As weights in DA are related to the number of data points incorporated in each model, faster dynamics bring to merge more models, and thus to increase more quickly the amount of data points incorporated in the local model.

The weight ratio in the DP strategy presents slightly different features, due to its being based on the losses of each model to be merged, measured on the local dataset of the merging node. In all of the three scenarios, we observe an initial decrease in the weight ratio. Indeed, in the first rounds, the local model incorporates mainly the data from its local dataset, which might not be very pertinent to the prediction task that the node needs to perform (possibly because the local dataset contains data about other parts of the city than that in which the node is located). Thus, the nodes need to learn more from the environment (i.e. from nodes who spent enough time in the region), and forget, at least in part, what they have learned on their local dataset. After this initial phase, however, the weight ratio again increases over time, as the local model starts to be fit for the node context.

In order to get a better idea of the prediction performance of the models trained

with our three GL algorithms, in Fig. 3.8 we have plotted a portion of a node trajectory, sampled every 15 s (points denoted as GT in the figure), as well as the forecasts of our three algorithms. The differences in accuracy among these algorithms emerging from Fig. 3.8 are in accordance with those from Fig. 3.4. The figure shows that all three algorithms are able to forecast changes in the direction and speed of the vehicle with a good degree of accuracy. Among these however, DP exhibits also good accuracy in predicting a sharp slowdown of the vehicle ("halt point" in the figure) and its duration, a trajectory feature among the most complex to forecast correctly, as it depends also on traffic conditions.

Note that prediction errors can be significantly reduced by simply projecting the estimated vehicle position onto the road grid. We are not implementing this step since we prefer to report the raw performance of the vehicle trajectory estimation rather than an improved estimate which includes some postprocessing algorithm (such as projection or fusion with other estimates, possibly dead reckoning)

3.5.2 Performance from start time

In another set of experiments, we have characterized the convergence properties of our distributed training approach. To this end, we have focused on settings and time intervals in which mobility patterns do not vary significantly, to reliably assess convergence and mean accuracy over time from the start of the GL schemes. Specifically, we considered the *Lux rush hour* scenario. On the one side, its duration (30 min) is long enough to allow our training framework to progress. On the other side, as we verified, it is short enough for vehicular mobility patterns not to vary significantly.

As Fig. 3.9 shows, our three GL algorithms steadily improve the average model performance over time. Note that the mean error in these plots is computed over all nodes present in the scenario at a given time slot, including those whose model has only been trained locally, e.g. because they have just arrived in the given region. As the plot shows, our GL schemes perform significantly better than local training, thus supporting the effectiveness of our collaborative model training strategies. Indeed, the local training strategy only marginally improves its performance over time (except for the first $10 - 20$ rounds), reaching values of mean error more than one order of magnitude larger than those achieved through our collaborative training schemes.

Another essential aspect emerging from these results is that the performance of our DA and DP strategies is very close to that of the equivalent centralized FL scheme, while that of the LBA strategy is significantly better, at least in the Lux rush hour scenario. This further supports the notion that training a model over a fully distributed, serverless architecture does not necessarily come at the cost of performance.

Fig. 3.9 allows also to compare the performance of our schemes with that of the distributed FL strategy denoted as "Flow-FL" [31]. Indeed, as shown in that paper, in the best conditions its performance can be assimilated to that of Federated Learning. This happens when node density and mobility are such as to allow maintaining a global state in a gossip-based shared memory in a reliable fashion for the whole duration of the training algorithm. Thus, in the most favorable conditions for Flow-FL, the considerations made for Federated Learning performance with respect to our GL schemes hold also for Flow-FL.
In order to assess the overall performance of our GL schemes in terms of prediction accuracy, in Table 3.3 we have compared the mean error of our GL strategies over the last two communication rounds of the time interval of each scenario, with that of dead reckoning (DR) strategy. Indeed DR is a natural benchmark for trajectory nowcasting. As these results show, in every scenario there is at least one GL-based strategy that outperforms DR. This shows that our collaborative learning strategies, despite being implemented over a simple LSTM architecture, enable a satisfactory performance in terms of prediction accuracy over a wide range of operating conditions. It also suggests the need for tuning the choice of the specific GL approach as a function of these conditions.

As already stated, the main competing approach to our GL learning schemes is the DFed Pow algorithm of [11]. This approach is applied to a discretized version of trajectory nowcasting, consisting of forecasting in which cell of the given region a vehicle will be at some point in the future. Thus, in order to assess the relative performance of our schemes with respect to it, we have considered the LBA strategy in the Lux rush hour scenario (the best-performing strategy in that scenario). To make performance comparison possible, we cast the regression result of the LBA strategy (i.e. the prediction of the specific location in which the vehicle will be 5 s in the future) into a classification result (i.e. a prediction of the cell in which the node will be). This is implemented by partitioning the given region in 49 square cells of side 150 m (similarly to what is done in [11]). Fig. 3.10 shows that LBA performs

Algorithm	Mean error [m]		
	Lux rush hour	Lux off-peak	Col off-peak
DP	7.52	13.3	12.37
DA	7.18	11.90	31.30
LBA	7.22	10.28	20.25
DR	10.18	11.64	25.19

Table 3.3: Mean error for the last two rounds from the start time of our GL schemes (DA, DP, and LBA) as well as for Dead Reckoning (DR), in the three considered scenarios.

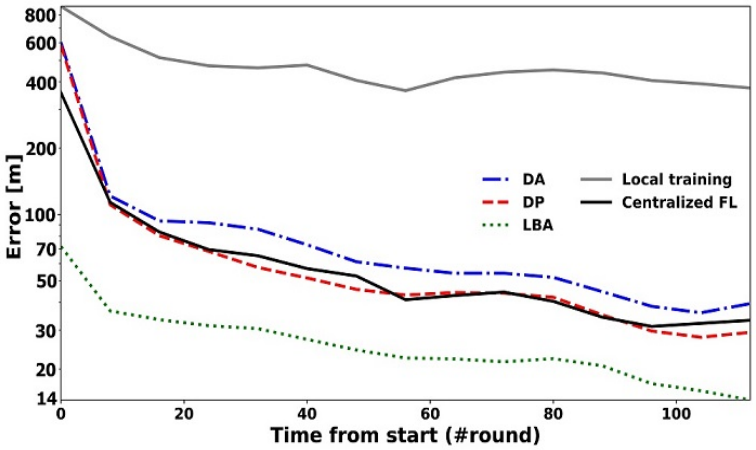


Figure 3.9: Mean error versus time from the start for our GL schemes, as a function of the merging strategy, as well as for centralized FL and the fully local training, in the *Lux rush hour* scenario.

significantly better than DFed Pow in terms of mean accuracy. The low performance of DFed Pow is due to the fact that it does not allow high-performing models to be used by nodes other than the ones to which they belong. For this reason, models from nodes moving in regions with low node density (and thus unfavorable for collaborative training), or whose local dataset is not able to support the training of a high-performing model, consistently experience unacceptably poor performance. Conversely, our collaborative GL schemes enable high-performing nodes to support poor-performing ones, by transferring their models and thus their learned knowledge, and to keep on improving them as new data is constantly being generated and used for collaborative training in the given region.

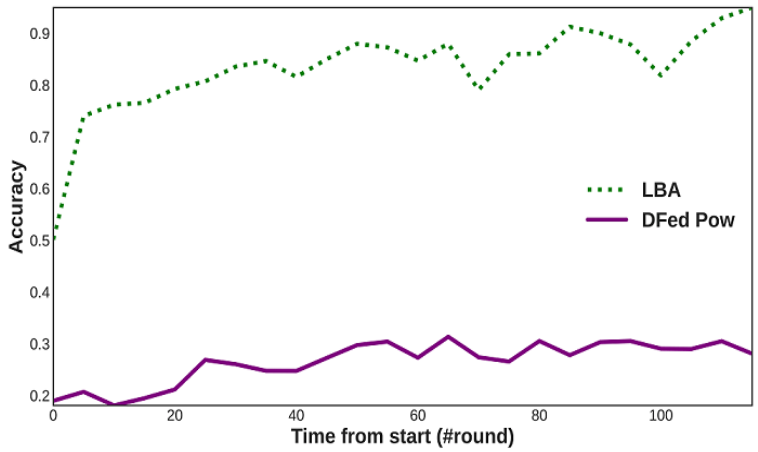


Figure 3.10: Mean accuracy versus time from the start for the LBA strategy, as well as for the DFed Pow algorithm [11], in the *Lux rush hour* scenario.

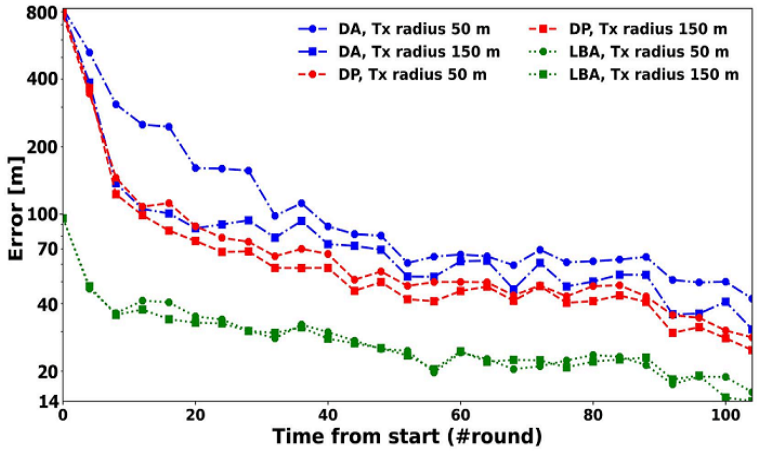


Figure 3.11: Mean error versus time from the start (first 30 minutes), for different values of transmission radius, for the DA, DP, and LBA algorithms, in the *Lux rush hour* scenario.

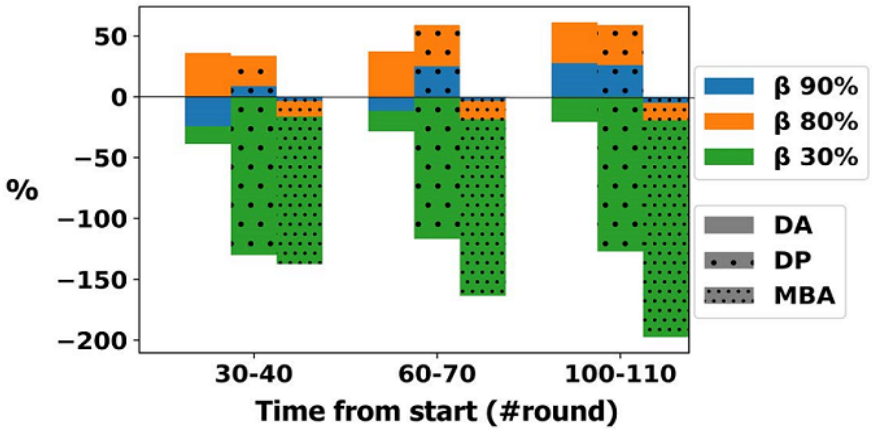
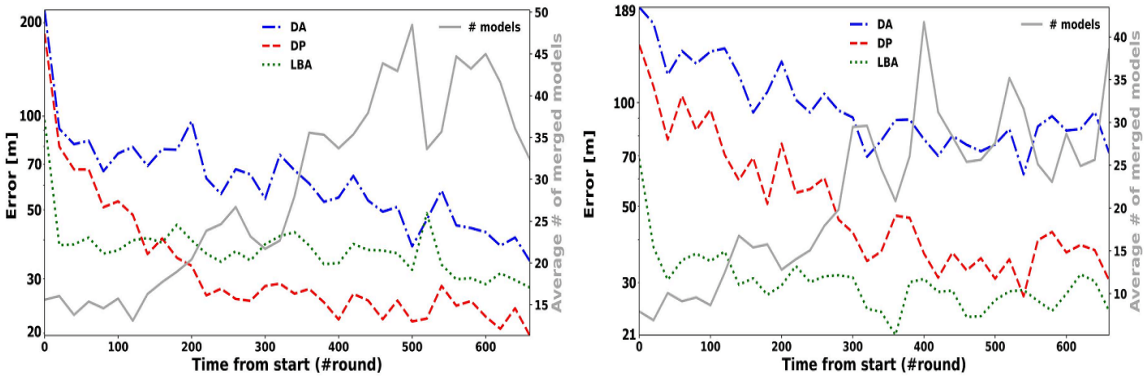


Figure 3.12: % decrease in mean error for the DA, DP and LBA algorithms with respect to the setup with $\beta = 100\%$ versus time from the start, as a function of the cutoff value, Luxembourg scenario.

3.5.3 Impact of transmission radius, and cutoff value

One of the primary parameters affecting the performance of our schemes is the number of models merged at each round. Thus, in another set of experiments, we have characterized its impact on accuracy and convergence speed.

Among the key parameters affecting the performance of our GL schemes, transmission radius has a key role. It determines not only the number of model instances merged at each round, but also the level of heterogeneity (e.g., in terms of accuracy) within the set of merged instances. Indeed, the trajectory prediction task is highly context-specific, particularly in realistic scenarios with nonuniformities in the spatial configuration of the road grid. As Fig. 3.11 shows, in our experiments in DA and DP algorithms, a larger transmission radius (and thus a larger number of models merged at each round) is associated with a decrease of the mean error at any point in time. However, our results suggest that this is less the case for the LBA scheme. As already seen, in LBA (which trains a number of cell-specific models in parallel), a transmission radius comparable to or larger than cell size brings to merging together models associated with neighboring cells, which are, in general, not a good fit for the given cell. In realistic and spatially inhomogeneous settings, this may potentially degrade the performance of LBA, as it destroys the cell-specific features of each model. One of the key features of model merging is that it does not necessarily produce a more accurate model than any of the models to be merged. This is the case, for instance, when those models are not sufficiently affine (e.g., in our setup, in terms of the specific city area within which a trajectory prediction



(a) 2:00 PM - 5:00 PM in the Cologne (b) 4:00 PM - 7:00 PM in the Luxembourg

Figure 3.13: Mean error versus time for the DA, DP and LBA algorithms.

has to be produced), or when some of them do not perform well for the specific prediction task at hand. As explained in our algorithms, the likelihood of this effect manifesting itself is minimized through an appropriate choice of the weights used in the merging process (which are, as we have seen, somehow related to a notion of the relevance of the model for the specific task). In addition, this is also achieved by tuning the cutoff value β, which prevents models with small values of weight from taking part in the merging process.

Thus, in another set of experiments, we assessed the impact of the cutoff value (expressed as the ratio of the sum of the weights retained in the merging process over the sum of the weights of all models to be merged) on prediction accuracy. As Fig. 3.12 suggests, there exists an optimal value of the cutoff, which in the considered settings is between 80% and 90%. Indeed, in all three algorithms, when the value of the cutoff is to retain only 30% of the contributions, the mean accuracy worsens with respect to no cutoff. In contrast, higher percentages of retained models yield an improvement in accuracy. As expected, in our experiments, the performance improvements and the optimal cutoff value highly depend on the specific merging algorithm and several features of the considered scenario, such as mean node density, mobility pattern, and mean vehicle sojourn time in the region. It suggests the need for strategies that adapt β to the specific setup.

3.5.4 Impact of nonstationarity in vehicular mobility

A clear feature arising from the numerical assessment of our algorithms is the high impact that node density and mobility patterns have on accuracy and convergence

rate. Since these features change substantially over time in realistic scenarios, we performed a set of experiments over time windows during which the configuration of vehicular traffic exhibits significant variations. Specifically, we have considered a time interval of three hours, from 4:00 PM to 7:00 PM in the Luxembourg scenario and from 2:00 PM to 5:00 PM in the Cologne scenario. As Fig. 3.13 shows, the substantial differences in a road grid configuration, time of the day, and mobility patterns, in both scenarios, the mean error decreases steadily for increasing time from the start of our schemes, despite variations of up to one order of magnitude in the number of vehicular nodes in the region. These features suggest that, except for minor fluctuations in the mean error, in realistic scenarios, our GL schemes are able not only to adapt in a timely manner to changes in mobility patterns but also to keep on improving the model performance over time.

3.6 Related work

Recently, distributed learning architectures have received a lot of attention from the research community [19, 24, 26, 12, 25, 4, 36, 41], aiming at overcoming some of the main limitations of centralized approaches (such as limited scalability, communication bottlenecks, performance under data imbalance and heterogeneity, under device heterogeneity and churn, to name a few) [19]. Decentralized Federated Learning, based on peer-to-peer communication between agents, has been proposed in [26, 12] as a way to address some of these shortcomings. Under this mechanism, each worker only demands communication with its neighbours for model exchanging in synchronous or asynchronous manners. To achieve a high training efficiency, however, proposed synchronous schemes are based on static topologies, such as a ring [25] or a static mesh [4, 36]. Thus they do not apply to dynamic settings. [41] presents a scheme with full connectivity among nodes, which adopts a hypothesis transfer learning approach to derive merged models. This work shows that even a straightforward GL scheme may enable substantial gains in terms of communication efficiency while at the same time achieving accuracy levels which are comparable to or even better than those of centralized approaches. Overall, the synchronisation requirement in all of these schemes implies that they are severely hampered by heterogeneity in computing power among nodes (particularly in training time). This worsens as the number of workers increases [26, 46]. For this reason, asynchronous solutions [6, 37] have been designed. [6] proposes a solution in which local models are

distributed over a logically fully connected peer-to-peer network. However, the full connectivity requirement still brings scalability and connectivity issues. [37] presents a fully serverless FL approach in which nodes receive a combined model from their neighbours in a static topology, and each one independently performs training on its local dataset in an asynchronous manner. [15, 33] explore the relationship between the (static) connectivity graph structure and convergence rate. Again, these results consider scenarios where each node communicates with all other nodes and/or the connectivity graph is static. [16] addresses the churn issue in a scenario where a set of static nodes learn a single global model through a gossip-based communication scheme. This work shows that, at least in static node scenarios, gossip schemes may achieve a level of accuracy comparable to (if not better than) FL. By considering a static scenario, however, [16] does not account for one of the critical elements of gossiping in a dynamic setting, i.e. the tight link between proximity in space among nodes, context information, correlation in the composition of local datasets, and correlation in tasks among nodes.

[4] proposes the first version of a GL scheme for multi-task learning on a static mesh network, proving its convergence. Unlike our work (and the vast majority of practical cases), such a GL scheme assumes that the relationship among tasks of different nodes is known in advance, and it does not specify how those weights should be derived. This is addressed in [45], which proposes a scheme for learning, in a distributed fashion, both the relationship among tasks and the weights to be used in model merging. However, such a scheme assumes a static node mesh without churn, making it unfit for applications in dynamic scenarios, such as in vehicular/pedestrian settings. Thus, none of these works considers scenarios with a dynamic topology among nodes in which patterns of model exchanges result from opportunistic contacts. This leaves open a set of issues relative to the performance of GL schemes, in terms of convergence and convergence speed, as well as the accuracy of the trained models [2], which we start addressing in the present work.

The issue of short-term vehicular trajectory prediction has recently received much attention, given the growing amount of applications and use cases, both in ITS and Autonomous driving domain, and in 5G and beyond network optimization and dynamic management [8, 18]. The growing availability of a large amount of car/user floating data has generated a large body of works which apply various deep learning techniques to the problem of predicting network-wide vehicle movement patterns in urban scenarios [8, 18, 27, 47]. [13] applies a LSTM architecture to the problem of

vehicle nowcasting in urban scenarios based on a centralized scheme.

The application of a gossip-based learning scheme for the issue of trajectory prediction has been proposed in [31]. The scheme aims at training a *single* ML model. It is based on a distributed implementation of a central coordinated function, with a global state maintained in a gossip-based shared memory, periodically updated via flooding. Consequently, such a single consensus scheme works well only when nodes form a single connected component, which drives the evolution of the learning process. Indeed, nodes who disconnect from such component cannot participate in the process until they reconnect. Differently from our scheme, this feature makes the solution in [31] ineffective in scenarios with more than one cluster, or in sparse scenarios where store-carry-and-forward is the main mode of content diffusion.

In a previous work [11], we presented a distributed scheme applied to a classification problem to derive a forecast about the section of the considered urban region where a vehicle will be at a given time in the future. Specifically, the given region of the plane was partitioned into *cells*, and each vehicle tried to predict in each cell it would be in h slots ahead in the future. Each node implemented a distributed version of FL, periodically sending its model to all neighbours, who train it on their local database and send it back to be merged with those sent by all other neighbours.

However, as shown in [11], this scheme implies twice the amount of model exchanges than the GL schemes in the present paper. In addition, it requires a very high number of local training steps per node, making it unsuitable for resource-constrained (computing and/or energy) devices. Moreover, the schemes in [11] suffer from a very high disparity in performance among nodes, with nodes in regions with low node density consistently experiencing unacceptably poor performance.

Conversely, the approach in the present paper allows high-performing nodes to quickly disseminate their models to those that cannot build a high-performing one. Moreover, in this way, (good) models persist probabilistically in the region over time, and they are constantly improved (and updated, as mobility patterns change over time) even by those nodes which do not possess enough neighbours or data to be able to train a high-performing model themselves.

3.7 Conclusions and future work

Vehicular position nowcasting has recently received much attention due to the growing amount of applications and use cases in 5G/B5G networks. In this paper, we propose a set of gossip learning algorithms for collaborative training of ML models for nowcasting in dense urban environments where node mobility and network churn are high.

We show that very good performance can be achieved in realistic dynamic environments within only a few hours from the initial installation of the position nowcasting application and with the very small initial dataset for each node. Our algorithms are proven to converge under very mild assumptions on the connectivity patterns and the data. We showed that our GL algorithms trained over a fully distributed, serverless architecture perform at least as well as server-based approaches such as federated learning.

Future developments of this work will include, first of all, attempts to reduce the average and variance of the error in the vehicle position estimation, possibly integrating the prediction of the GL model with data available onboard. Another future direction of the investigation will involve extending the analysis of the feasibility of GL training approach to other learning tasks, and a more fine-grained analysis of the impact of the main system parameters on GL performance, by means of synthetic dynamic graphs. Finally, other contributions rely on evaluating the effects of heterogeneity in node mobility, connectivity among nodes, and computing power available at nodes and investigating their effect on the convergence of our schemes and model accuracy.

Bibliography

[1] Khadige Abboud, Hassan Aboubakr Omar, and Weihua Zhuang. Interworking of DSRC and cellular network technologies for V2X communications: A survey. *IEEE transactions on vehicular technology*, 65(12):9457–9470, 2016.

[2] Abdul Aziz Alkathiri, Lodovico Giaretta, Sarunas Girdzijauskas, and Magnus Sahlgren. Decentralized word2vec using gossip learning. In *23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*, 2021.

[3] Florent Altché and Arnaud de La Fortelle. An LSTM network for highway trajectory prediction. In *IEEE ITSC*, pages 353–359, 2017.

[4] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 473–481. PMLR, 2018.

[5] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[6] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.

[7] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, pages 712–719, 2014.

[8] Seongjin Choi, Hwasoo Yeo, and Jiwon Kim. Network-wide vehicle trajectory prediction in urban traffic networks using deep learning. *Transportation Research Record*, 2672(45):173–184, 2018.

[9] Francois Chollet. *Deep Learning with Python and Keras: The practical manual from the developer of the Keras library*. MITP-Verlags GmbH & Co., 2018.

[10] L. Codeca, R. Frank, and T. Engel. Luxembourg SUMO Traffic (LuST) Scenario. In *IEEE VNC*, pages 1–8, Dec 2015.

[11] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Gossip learning of personalized models for vehicle trajectory prediction. In *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–7. IEEE, 2021.

[12] Anis Elgabli, Jihong Park, Amrit S Bedi, Mehdi Bennis, and Vaneet Aggarwal. Communication efficient framework for decentralized machine learning. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2020.

[13] Umberto Fattore, Marco Liebsch, Bouziane Brik, and Adlen Ksentini. Automec: Lstm-based user mobility prediction for service management in distributed mec resources. In *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 155–159, 2020.

[14] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[15] L. Giaretta and S. Girdzijauskas. Gossip Learning: Off the Beaten Path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124, 2019.

[16] István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

[17] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[18] Huatao Jiang, Lin Chang, Qing Li, and Dapeng Chen. Trajectory prediction of vehicles based on deep learning. In *IEEE ICITE*, pages 190–195, 2019.

[19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

[20] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE ITSC*, pages 399–404. IEEE, 2017.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. MLbase: A Distributed Machine-learning System. In *Cidr*, volume 1, pages 2–1, 2013.

[23] Puneet Kumar, Mathias Perrollaz, Stéphanie Lefevre, and Christian Laugier. Learning-based approach for online lane change intention prediction. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 797–802. IEEE, 2013.

[24] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[25] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[26] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052. PMLR, 2018.

[27] Lei Lin, Siyuan Gong, Tao Li, and Srinivas Peeta. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles. In *The Autonomous Vehicles Symposium*, volume 2018, 2018.

[28] Qian Liu, Gang Chuai, Jingrong Wang, and Jianping Pan. Proactive mobility management with trajectory prediction based on virtual cells in ultra-dense networks. *IEEE Transactions on Vehicular Technology*, 69(8):8832–8842, 2020.

[29] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel,

Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *IEEE ITSC*, pages 2575–2582, 2018.

[30] Aamir Mahmood, Luca Beltramelli, Sarder Fakhrul Abedin, Shah Zeb, Nishat I. Mowla, Syed Ali Hassan, Emiliano Sisinni, and Mikael Gidlund. Industrial iot in 5g-and-beyond networks: Vision, architecture, and design trends. *IEEE Transactions on Industrial Informatics*, 18(6):4122–4137, 2022.

[31] Nathalie Majcherczyk, Nishan Srishankar, and Carlo Pinciroli. Flow-FL: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. In *IEEE ICRA*, pages 8836–8842, 2021.

[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[33] Giovanni Neglia, Gianmarco Calbi, Don Towsley, and Gayane Vardoyan. The role of network topology for distributed machine learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2350–2358. IEEE, 2019.

[34] Peter Ondrúška and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proceedings of AAAI*, pages 3361–3367, 2016.

[35] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[36] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[37] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[38] Nordic Semiconductor. Things you should know about bluetooth range. `https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range`. (Accessed on 06/09/2022).

[39] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30:4424–4434, 2017.

[40] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2013.

[41] Lorenzo Valerio, Andrea Passarella, and Marco Conti. Hypothesis transfer learning for efficient data computing in smart cities environments. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2016.

[42] Andras Varga. *OMNeT++*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[43] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[44] Z. Xiao, P. Li, V. Havyarimana, G. M. Hassana, D. Wang, and K. Li. GOI: A Novel Design for Vehicle Positioning and Trajectory Prediction Under Urban Environments. *IEEE Sensors Journal*, 18(13):5586–5594, 2018.

[45] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully decentralized joint learning of personalized models and collaboration graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 864–874. PMLR, 2020.

[46] Liang Zhao, Wen-Zhan Song, Xiaojing Ye, and Yujie Gu. Asynchronous broadcast-based decentralized learning in sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 33(6):589–607, 2018.

[47] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

3.8 Appendix

3.8.1 Proof of Theorem 1

Proposition 2. *It is given $Q^j(\mathbf{w})$ with DA merging strategy. Then $\forall \mathbf{w} \in \mathbb{R}^{N\times p}$, $\forall 0 \leq j-\delta \leq j$ with $\delta \in \mathbb{N}$, $\forall i$,*

$$E[Q^{i,j-\delta}(\mathbf{w})] = E[Q^{i,j}(\mathbf{w})]$$

Proof. Let us consider the equation for the performance estimator in DA at round j:

$$\xi_j^i = \frac{\sum\limits_{k\in K_{j-1}^i} (\xi_{j-1}^k)^2}{\sum\limits_{h\in K_{j-1}^i} \xi_{j-1}^h} = \frac{\sum\limits_{k\in K_{j-1}^i} \xi_{j-1}^k}{\sum\limits_{h\in K_{j-1}^i} \xi_{j-1}^h} \frac{\sum\limits_{k_1\in K_{j-2}^k} (\xi_{j-2}^{k_1})^2}{\sum\limits_{h\in K_{j-2}^k} \xi_{j-2}^h}$$

Proceeding backwards, and substituting recursively the same expression, we get

$$= \frac{\sum\limits_{k\in K_{j-1}^i} \xi_{j-1}^k}{\sum\limits_{h\in K_{j-1}^i} \xi_{j-1}^h} \cdot$$

$$\cdot \left(\frac{\sum\limits_{k_1\in K_{j-2}^k} \xi_{j-2}^{k_1}}{\sum\limits_{h\in K_{j-2}^k} \xi_{j-2}^h} \cdot \left(\dots \cdot \left(\frac{\sum\limits_{k_{\delta-1}\in K_{j-\delta}^{k_{\delta-2}}} (\xi_{j-\delta}^{k_{\delta-1}})^2}{\sum\limits_{h\in K_{j-\delta}^{k_{\delta-2}}} \xi_{j-\delta}^h} \right) \dots \right) \right)$$

This expression is thus the product of δ factors. As we can see, each factor is dependent on the preceding one for the composition of the set K. For instance, in the second factor the set K_{j-2}^k denotes the set of nodes which in round $j-2$ sent their local instance to node k, which is the index of a node considered in round $j-1$ in the first factor. In our GL scheme, the process which assigns a set K of contributors to a given node is determined by the node's position, speed, and the conditions of the wireless channel between two nodes in range, among others. Given the stochastic nature of these factors, we model their compound effect on the composition of the set K by assuming it to be a random collection of node indices among $\mathcal{N}$. We further assume the composition of any two such sets to be independent, when associated with different nodes and/or different rounds. This is clearly an approximation, as nodes close among them will tend to have at least part

of the elements of the set K in common, and the same is true for the same node in consecutive slots. With this assumption, every fraction in the above expression, relative to a given slot, is a random variable independent from those relative to other slots. Thus, in the computation of $E[\xi_j^i]$, the expectation of the product becomes the product of the expectations. We have thus that

$$E\left[\xi_j^i\right] = \left(\prod_{\tau=j-\delta+1}^{j} E\left[\frac{\sum_{k\in\mathcal{K}_\tau}\xi_\tau^k}{\sum_{h\in\mathcal{K}_\tau}\xi_\tau^h}\right]\right) E\left[\frac{\sum_{k\in\mathcal{K}_{j-\delta}}(\xi_{j-\delta}^k)^2}{\sum_{h\in\mathcal{K}_{j-\delta}}\xi_{j-\delta}^h}\right]$$

$$= E\left[\frac{\sum_{k\in\mathcal{K}_{j-\delta}}(\xi_{j-\delta}^k)^2}{\sum_{h\in\mathcal{K}_{j-\delta}}\xi_{j-\delta}^h}\right]$$

Thus $E[\xi_j^i] = E[\xi_{j-1}^i] = ... = E[\xi_{j-\delta+1}^i]$. If we assume that the process of arrivals and departures of a node is stationary and that the sampling function which samples randomly the set of nodes present in the region into a set K to be the same for all nodes, and that the $\xi_{j-\delta+1}^i$ are drawn from a same distribution $\forall i$, then $E[\xi_j^i] = E[\xi_k^j]$ for any couple of nodes i, k present at round j. In the same way, we can show that, for any node i at rounds j and j',

$$E\left[\sum_k X_{ik}^j w_j^k\right] = E\left[\sum_k X_{ik}^{j'} w_{j'}^k\right].$$

□

As a corollary, it is easy to see that $E[Q^{i,j+1}(\mathbf{w}_{j+1})] = E[Q^{i,j}(\mathbf{w}_{j+1})]$. We now prove Theorem 1. Using Taylor's expansion, and assumptions 1 and 2, and considering a step of the stochastic gradient descent for which the variation is only in the dimension i', we get:

$$Q^{i,j}(\mathbf{w}_{j+1}) \;=\; Q^{i,j}\left(\mathbf{w}_j - \frac{1}{L_i^j}[\bigtriangledown Q^{i,j}(\mathbf{w}_j)]_{i'} \times e_{i'}\right)$$

using Taylor's expansion

$$\leq Q^{i,j}(\mathbf{w}_j) + [\bigtriangledown Q^{i,j}(\mathbf{w}_j)]_{i'}\left(-\frac{1}{L_i^j}[\bigtriangledown Q^{i,j}(\mathbf{w}_j)]_{i'}\right)$$

$$\leq Q^{i,j}(\mathbf{w}_j) - \frac{1}{2L_{max}}\left[\bigtriangledown Q^{i,j}(\mathbf{w}_j)\right]_{i'}^2$$

Taking the expectations of both sides,

$$\mathbb{E}[Q^{i,j+1}(\mathbf{w}_{j+1})] \leq \mathbb{E}[Q^{i,j}(\mathbf{w}_j)] - \frac{1}{2nL_{max}}\mathbb{E}\left[\|(\bigtriangledown Q^{i,j}(\mathbf{w}_j))\|^2\right]. \tag{3.11}$$

Here, we used the facts that $\mathbf{w}_j$ does not depend on i' and that i' is chosen randomly among $[1, \ldots, n]$. Let

$$\theta_j^i := \mathbb{E}[Q^{i,j}(\mathbf{w}_j)] - Q^{i,1}(\mathbf{w}_*^{i,1})$$

then

$$\theta_{j+1}^i \quad \leq \quad \theta_j^i - \frac{1}{2nL_{max}}\mathbb{E}\left[\|(\bigtriangledown Q^{i,j}(\mathbf{w}_j))\|^2\right]. \tag{3.12}$$

When $Q^{i,j}$ is σ-strongly convex with modulus $\sigma > 0$, we get

$$Q^{i,j}(\mathbf{w}_*^{i,1}) \geq Q^{i,j}(\mathbf{w}_j) - \frac{1}{2\sigma}|| \bigtriangledown Q^{i,j}(\mathbf{w}_j)||^2. \tag{3.13}$$

This implies that

$$|| \bigtriangledown Q^{i,j}(\mathbf{w}_j)||^2 \geq 2\sigma\left(Q^{i,j}(\mathbf{w}_j) - Q^{i,j}(\mathbf{w}_*^{i,1})\right) \tag{3.14}$$

Combining (3.12) with (3.14) and taking the expectation, yields

$$\theta_{j+1} \leq \theta_j - \frac{\sigma}{nL_{max}}\theta_j = \left(1 - \frac{\sigma}{nL_{max}}\right)\theta_j. \tag{3.15}$$

Recursively applying this inequality $J - j'$ times yields Equation 3.10.

Modeling and Analysis of Network Dynamics Impact

CHAPTER 4

The Upsides of Turbulence: Baselining Gossip Learning in Dynamic Settings

Published in Proceedings of the 24^{th} *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23),*
Pages: 376–381.
Date: October 2023.
Publisher: Association for Computing Machinery.
ISBN: 978-1-4503-9926-5.

Abstract– In dynamic settings, fully distributed gossip-based learning schemes have recently gained interest due to their better scalability, robustness, and enhanced privacy protection compared to server-based architectures. However, existing approaches to their performance characterization either assume stable connectivity among nodes or are ad-hoc for specific trace-based mobility patterns. Thus, in dynamic settings, there is currently a poor understanding of the conditions under which gossip-based learning schemes are feasible, and of their main performance tradeoffs. In this work, we start addressing this issue by performing a first baselining of Gossip Learning (GL) on random TVG, to get a first-order characterization of their main performance patterns in dynamic settings. The use of random TVG enables a fine-grained and accurate characterization of GL effectiveness as a function of the main system parameters while abstracting from scenario-specific features of patterns of communication and mobility (e.g., induced by road grids or measured mobility traces). Our results suggest that GL schemes are robust to node mobility

and comparable in accuracy and convergence speed to Federated Learning architectures, over a wide range of operational conditions. We show that the final model accuracy is robust against data dispersion across nodes as well as against very low rates of exchanges across nodes.

4.1 Introduction

The pervasive diffusion of the IoT and Smart City paradigms, is pushing the bulk of the computing load to elaborate increasing amount of data towards the edge of the network [9]. This trend triggered interest in distributed ML [13] which, by reducing data transfer to the cloud, improves privacy while decreasing bandwidth utilization. However, many distributed schemes scale poorly, as they rely on infrastructure-based information exchanges or a central coordination server. These issues sparked interest in fully distributed schemes, such as GL [11, 5, 14], based on a server-less, fully distributed model training approach and on knowledge transfer among agents via direct, peer-to-peer exchanges of models. They scale well with the number of agents, as each agent contributes to service capacity not only with data but also by adding computing and communication capacity. Several GL schemes have been proposed for a large variety of learning architectures ([5, 14]), scenarios and applications [6, 4, 1]. However, the majority of these schemes assume *static* network topologies (e.g., ring [7] or mesh [12, 14, 2]). In such networks, [5] shows that GL converges and delivers comparable accuracy to that of server-based distributed schemes such as FL. However, these results do not apply to dynamic settings, where node mobility implies a network topology that changes over time, such as in Vehicle-to-Vehicle (V2V) networks or robot swarms. Several studies assess GL in dynamic scenarios (e.g., from fully-connected [8] to volatile vehicular networks [4]) showing that convergence to a high-accuracy models, even in trace-driven mobility scenarios, is possible. However, these assessments are limited to specific trace-based mobility patterns, which are hard to extrapolate to other settings, and do not offer insights into the relationship between the main system parameters and GL's key performance indicators. Specifically, when nodes move, it is currently unclear what are the operating conditions in which GL schemes converge and deliver satisfactory performance and how their performance compares to Centralized Learning or other distributed schemes such as Federated Learning. In this work, we investigate the feasibility of GL in dynamic settings by elaborating a first characterization of the basic mecha-

nisms affecting its performance on random TVG, as a function of the main system parameters and the main structural properties of the time-varying network. The use of synthetic graphs enables a fine-grained and accurate system characterization that abstracts from context-specific spatiotemporal patterns of communication and mobility, such as those found in measurement-based mobility traces. Specifically, our main contributions are:

- We characterize the performance of GL on dynamic random graphs, based on a GL scheme that generalizes Federated Learning (FL) [10] to fully decentralized dynamic settings, including FL as a particular case. We show that GL schemes are robust to node mobility over an extensive range of scenarios, regarding the number of nodes and frequency of inter-node contacts. To the best of our knowledge, we are the first to characterize and assess GL feasibility over non-trivial yet non-trace-driven connectivity patterns.
- We determine the impact of the main system parameters on GL performance. We show that, even in dynamic settings, GL accuracy and convergence speed are comparable to those of centralized Federated Learning schemes.
- We show that the final model accuracy is robust against data dispersion across nodes as well as against very low rates of exchanges across nodes.

These results suggest that node mobility and the lack of coordination among nodes do not cause a performance penalty in GL compared to centralized architectures, such as FL, over an extensive set of system configurations.

4.2 System model

We consider a set V of mobile nodes modeling, e.g., smartphones, UAVs, and connected vehicles. We assume that each node is endowed with an ML model whose architecture is equal for all nodes and which needs to be trained and used by each node to perform a specific inference task. Assuming the same ML architecture for every model in the system is necessary to make model aggregation possible, as the aggregation operations between models are only possible between parameter vectors of the same dimension. Each node is also endowed with a set of data points, denoted as *local dataset*. We assume nodes can communicate directly among themselves through wireless Device-to-Device (D2D) communications. Communication

between two nodes $v_i, v_j \in V$ may occur whenever they are in *contact*, i.e., within the transmission range of each other. We assume time is divided into intervals called *slots*, indexed with $t \in \mathbb{N}$.

4.2.1 Time-Varying Graph Model

We model the mobile nodes' connectivity graph and its evolution over time as a TVG, composed of a set V of nodes and a time-variable set E_t of edges between nodes. An edge $\in E_t$ models the existence of a direct wireless channel between two nodes. We assume the graph to be constant within each time slot and to (possibly) vary only from one slot to the following one. The resulting dynamic graph, denoted as $G = \{G_t = (V, E_t) : t \in \mathbb{N}\}$, is thus a sequence of graphs, each associated with a time slot. The volatility and dynamicity of the wireless channel are modeled by the fact that the set of edges E_t can be different at each slot. These dynamic graphs are often used to model opportunistic gossiping schemes because they simplify assumptions about the network structure while still capturing key characteristics of real-world networks. In particular, they allow varying the number of nodes, the connectivity patterns between nodes, and the frequency and duration of node interactions in a controlled and systematic way. In this paper, we model G as an Erdős-Rényi dynamic graph [3], a type of uniform random graph. Specifically, at any time slot t and for any two nodes $v_i, v_j \in V$, the probability p that an edge exists between them is the same for any i, j, t. This assumption gives the graph a homogeneous structure, ensuring stationary patterns of evolution over time. We further assume that the edges are *conditionally independent* of each other, i.e., $\forall t \in \mathbb{N}, v_i, v_j \in V : \mathbb{P}[(v_i, v_j) \in E_t] = \mathbb{P}[(v_i, v_j) \in E_t | (v_i, v_j) \in E_{t-1}] = p$. The choice of p determines the graph's degree of connectivity at any time slot and, thus, the rate at which new connections are established and terminated between nodes, which is a key aspect of network dynamicity. Moreover, p also determines the mean duration of a link between any two nodes and, thus, the edge density and clustering degree of the graph. We chose Erdős-Rényi dynamic random graphs because they are among the simplest dynamic random graph models of realistic D2D network topologies that provide mathematical guarantees on key graph metrics such as the average number of edges $\mathbb{E}[|E_t|] = p\binom{|V|}{2}, \forall t \in \mathbb{N}$, the nodes' degree distribution $\mathbb{P}(d(v) = k) = \binom{|V|-1}{k} p^k (1-p)^{|V|-1-k}$, and the p threshold for the almost-sure graph's connectivity $p > \frac{\ln |V|}{|V|}$. Assuming conditional independence of edges allows modeling worst-case edge dynamics, where changes in connectivity patterns

are abrupt and fast. Such model choice approximates well real-world scenarios in which node mobility and time-slot duration are sufficiently high to make the inter-slot edge dependence negligible. Furthermore, mobile networks' connectivity can be modeled with a random graph when the GL message exchange dynamics are considerably slower than the nodes' physical mobility dynamics. Even though the stochastic properties of random graphs are well-studied in the literature (e.g., node contact rate and duration, graph topology's feature distribution, etc.), the performance evolution of distributed ML applications running on such dynamic networks are still unexplored, which calls for Monte-Carlo-like assessment of such systems.

4.2.2 Gossip Learning Operation

Algorithm 8 Basic GL algorithm. The ML model weights and the set of neighbors of node v in time slot t are denoted by w_t^v and K_t^v, respectively. The loss of node k's model on node v's dataset is denoted by l_k^v and its formulation is task-specific (e.g., cross-entropy for classification tasks).

$w_0 \leftarrow$ INITIALIZE()
for $\forall v \in V$ **do** $w_0^v \leftarrow w_0$
loop ▷ $\forall v \in V$ executes this loop in parallel, start from $t \leftarrow 0$
  $w_t^v \leftarrow \text{TRAIN}(w_t^v)$
  for $\forall k \in K_t^v$ **do** Send w_t^v to k, Receive w_t^k from k
  for $\forall k \in K_t^v \cup \{v\}$ **do** Compute l_k^v
  $w_{t+1}^v \leftarrow \left(\sum_{k \in K_t^v \cup \{v\}} w_t^k 2^{-l_k^v} \right) \cdot \left(\sum_{k \in K_t^v \cup \{v\}} 2^{-l_k^v} \right)^{-1}$ ▷ MERGE
  $t \leftarrow t + 1$

We detail the operation of the basic GL algorithm (Algorithm 8) run by each node. In this work, we assume all models are initialized with identical random weights, which has been shown to improve convergence speed and accuracy [10]. Starting from $t = 0$, in every time slot, the algorithm proceeds through three *phases*, which we assume to be synchronized across nodes. In the first phase (*training*), each node trains its local model instance on the local dataset. In the second phase (*exchanging*), each node sends its local model instance to its neighbors and receives their local instance. For simplicity, we assume model exchanges to be instantaneous, disregarding the negative impact of network interference on model transmission speed for finite model transmission lengths. Finally, in the third phase (*merging*), each node *merges* the models received from neighbor nodes with its local model (i.e., it computes a linear combination of them) to produce a *meta-model*, similarly to what parameter servers do in centralized FL algorithms [10]. The weights of the merging operation are computed via the *Decentralized Powerloss* (DP) strategy [4],

where each weight is a function (Algorithm 1, line 7) of the loss computed over the context-specific validation set. We chose the DP merging strategy as it has proven superior performance to other state-of-the-art merging approaches [4]. These three phases are repeated until a termination criterion is met. For instance, after a maximum number of iterations is attained or when the average model's accuracy exceeds a threshold. In this work, we assumed the termination criterion is met when the global model's accuracy does not improve more than a fixed threshold over a fixed number of gossip rounds.

4.3 Performance Evaluation

4.3.1 Simulation Setup

We assess the performance of GL schemes on time-varying graphs considering the case in which a set V of homogeneous nodes in the system need to train an ML model to perform an inference task such as handwritten digit recognition (MNIST dataset, $m = 10$ classes, image size $n = 28$) or object recognition (CIFAR-10 dataset, $m = 10$ classes, image size $n = 32$) from a set of images. Each node in the system has a *local dataset* of equal size for all nodes and each data point in the system can belong to at most one local dataset. All local datasets are i.i.d. and partitioned in 85% training set and 15% validation set. Let us denote the union of all local datasets in the system as the *global dataset* of size γ. This study assumes the global dataset is built as a random sample of γ dataset samples from one of the two source datasets (MNIST or CIFAR-10). By varying γ, we modulate the total amount of data in the system available for the collaborative model training. To each global dataset, we associate a *global test set* obtained by random sampling 20% of the source datasets and ensuring that the global dataset and test set are disjoint. We assume that the global dataset is equally distributed across all nodes in the scenario, meaning that each node's local dataset size is $\gamma/|V|$. This choice allows us to assess the impact of information fragmentation on GL performance for a fixed global dataset size.

To perform both inference tasks, we assume that nodes use supervised models trained with Mini-Batch Stochastic Gradient Descent (SGD) with Categorical Cross-Entropy loss, early-stopping patience of 20 epochs, batch size of 32, momentum of 0.9, and a 10^{-4} learning rate. Specifically, we assumed every node in the system executes a Convolutional Neural Network (CNN), whose architecture and hyperpa-

rameters (identical for all nodes) are designed for effective shape feature extraction. Layer 1 (input) is a 2D Convolution with 32 filters and 3x3 kernel. Layer 2 is a 2x2 Max Pooling. Layer 3 is a 100-neuron Dense layer with ReLu activation. Finally, Layer 4 (output) is a 10-neuron Dense layer with SoftMax activation. Further hyperparameter optimization is out of the scope of this work and is left to future investigation. We compare the performance of GL with the following baselines:

- *Centralized Learning*, where a server collects the local datasets from all participants, aggregates them into a global dataset, and locally trains a global model on it.
- *Local Learning*, in which each node trains its local model using only its local dataset without exchanging data or models with other nodes or a centralized server. It is derived from our GL reference scheme by considering TVGs with $p = 0$. When $p = 0$, each node's local model does not change over time compared to the one trained at $t = 0$.
- *Federated Learning* [10], in which a server collects the models from all participants at each iteration, aggregates them, and redistributes the aggregated model to all participants. For comparison, we assume models are aggregated with Decentralized Powerloss (as in Algorithm 8, line 7). FL can be derived as a special case of our GL scheme when applied to TVGs with $p = 1$ (i.e., complete graphs).

We empirically verified that an early-stopping patience of 20 epochs for the local training was sufficient to detect convergence accurately. We assume the random coefficients of the initial ML model at $t = 0$ are the same for all nodes because independent random model initializations converge to different weights after training, reducing global model accuracy when merging them [10]. We measure the *accuracy* A_t of each node's local model at time slot t, defined as the fraction of the model's correct predictions out of all predictions for the global test set. We denote a node's accuracy at $t = 0$, before gossiping starts, as A_0 (*initial accuracy*), and a node's accuracy at convergence as A_C (*final accuracy*). We define the gossip *convergence time* C as the number of time slots required for the system to converge. We assumed our schemes to have converged when the local models' accuracy (averaged across all nodes) did not improve by more than 0.5% over the last 10 rounds (gossip patience).

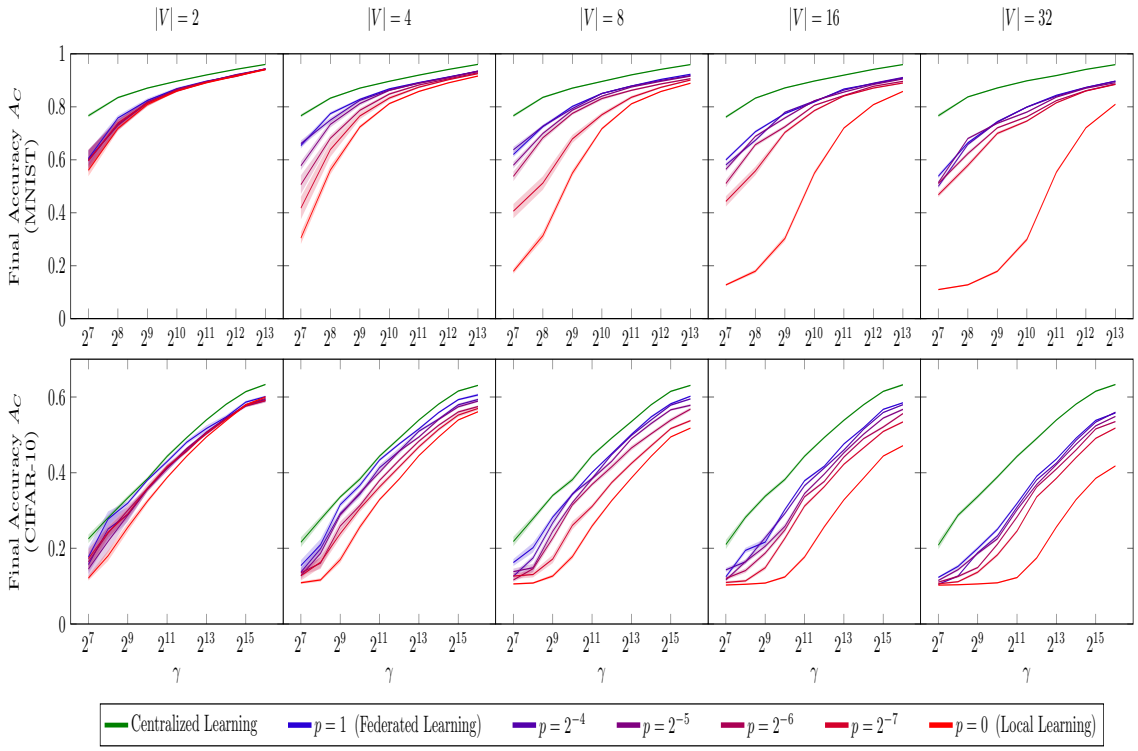


Figure 4.1: Impact of global dataset size γ, number of nodes $|V|$, and probability of connection p on the models' final accuracy A_C at convergence, over 15 repeated experiments. Curves are mean central tendencies for distributions of A_C for each user in the system, surrounded by an asymptotic confidence interval at 95% level.

4.3.2 Simulation Results

Figure 4.1 shows the detrimental effect of *information fragmentation* (i.e., the partitioning of the global dataset into $|V|$ local datasets) on the average accuracy of isolated local models, and how distributed learning schemes, such as Federated Learning and Gossip Learning can mitigate them. Information fragmentation also introduces an upper bound to the average accuracy achievable through distributed learning schemes, which decreases as the fragmentation (i.e., the number of nodes $|V|$) increases. We call *fragmentation loss* the distance between the Centralized Learning accuracy and the maximum Gossip Learning accuracy (i.e., achieved in scenarios with $p = 1$, equivalent to centralized Federated Learning). We show that the more a network is connected (i.e., larger p), the larger the opportunity for distributed-learning schemes to improve local model accuracy by aggregating other models' knowledge from external local datasets. Larger global dataset sizes increase accuracy for all compared learning methods (Centralized, Local, Centralized Federated, Gossip) and all topologies (number of nodes $|V|$ and connectivity p), confirming the expected behavior. In Figure 4.1, the green curve (Centralized Learning) is the same for all $|V|$, as it only depends on the global dataset size, and achieves the highest accuracy compared to the other baselines because the model is trained with all available information in the system. As the number of nodes $|V|$ increases, the size of the local dataset decreases, resulting in the gap between the accuracy A_C for centralized and local learning schemes becoming proportionally larger due to information fragmentation. As the probability of connection p increases, the network topology becomes more connected and each node has a higher expected number of neighbors (namely, in the order of $p|V|$) at each time slot with which to exchange trained local models and perform aggregation (gossiping). We call *gossip opportunity* the distance between the local learning curve $p = 0$ and the Federated Learning curve $p = 1$. We observe that, for every number of nodes and any global dataset size, the higher the probability of connection p the greater the average accuracy A_C of local models, showing the positive impact of network connectivity on average model accuracy. Even though the highest accuracy is always achieved in fully connected scenarios (i.e., $p = 1$), the "speed" at which the accuracy improves from $p = 0$ scenarios to $p = 1$ scenarios (i.e., the function between p and the achievable accuracy) depends on $|V|$. In particular, we observe that both gossip opportunity and fragmentation loss increase with $|V|$, and that the distance between the accuracy for $p = 1$ scenarios and the accuracy for the other simulated

scenarios (i.e., $0 < p \leq 1$) decreases as $|V|$ increases. This shows that for some values of connection probability p, the higher the fragmentation $|V|$, the closer the accuracy is to the maximum achievable. Figure 4.2 shows the knowledge diffusion properties of GL for varying global dataset size and information fragmentation. As $|V|$ increases, the initial accuracy's variability is progressively less explanatory of the final accuracy's variability. This effect shows that the larger the network size $|V|$, the more GL schemes can disseminate the knowledge contained in the nodes' trained local models throughout the system, providing each node with a local model with similar accuracy performance. The global dataset size is positively correlated with both initial and final accuracy and a moderate clustering effect around lower values of initial accuracy especially for lower global dataset sizes. This effect is due to the modest size of local datasets for scenarios where γ is small and $|V|$ is large, where local datasets do not have any sample for several classes to predict (for example, in MNIST, local datasets that do not include any dataset sample for certain digits). Figure 4.3 shows a matrix heatmap in which each cell is the Spearman's correlation coefficients $\rho_{i,j}$ between a pair of system parameters or performance metrics i and j, across all collected data. Scenarios with a higher number of nodes, and therefore a higher information fragmentation, require a statistically higher number of gossip rounds to converge. We observe a negative correlation between the number of nodes $|V|$ and both the initial and final accuracy, and a positive correlation between the number of nodes $|V|$ and the difference between the final and initial accuracy. This evidence supports what Figure 4.1 suggested, i.e., that a higher information fragmentation harms the final accuracy by introducing a *fragmentation loss*, but it also represents the biggest opportunity for GL to improve model accuracy over the local training case. Furthermore, higher information fragmentation provides nodes with smaller local datasets, which reduces their initial accuracy.

Across different scenarios, the probability of connection p appears less than weakly correlated with the convergence time C, final accuracy A_C, and accuracy gain $A_C - A_0$ due to gossiping, i.e., $|\rho_{p,C}|, |\rho_{p,A_C}|, |\rho_{p,A_C-A_0}| < 0.15$. However, stratifying the collected data by different values of $|V|$ revealed stronger correlations between p and the three metrics C, A_C, and $A_C - A_0$ for some values of $|V|$. For example, for $|V| = 4$ we observe $\rho_{p,A_C-A_0} = 0.34$ and a $\rho_{p,C} = 0.16$, while for $|V| = 32$ we observe $\rho_{p,A_C-A_0} = 0.07$ and a $\rho_{p,C} = -0.17$. This effect shows that the probability of connection p impacts the convergence time C and the accuracy gain due to gossiping $A_C - A_0$ differently depending on the information fragmentation.

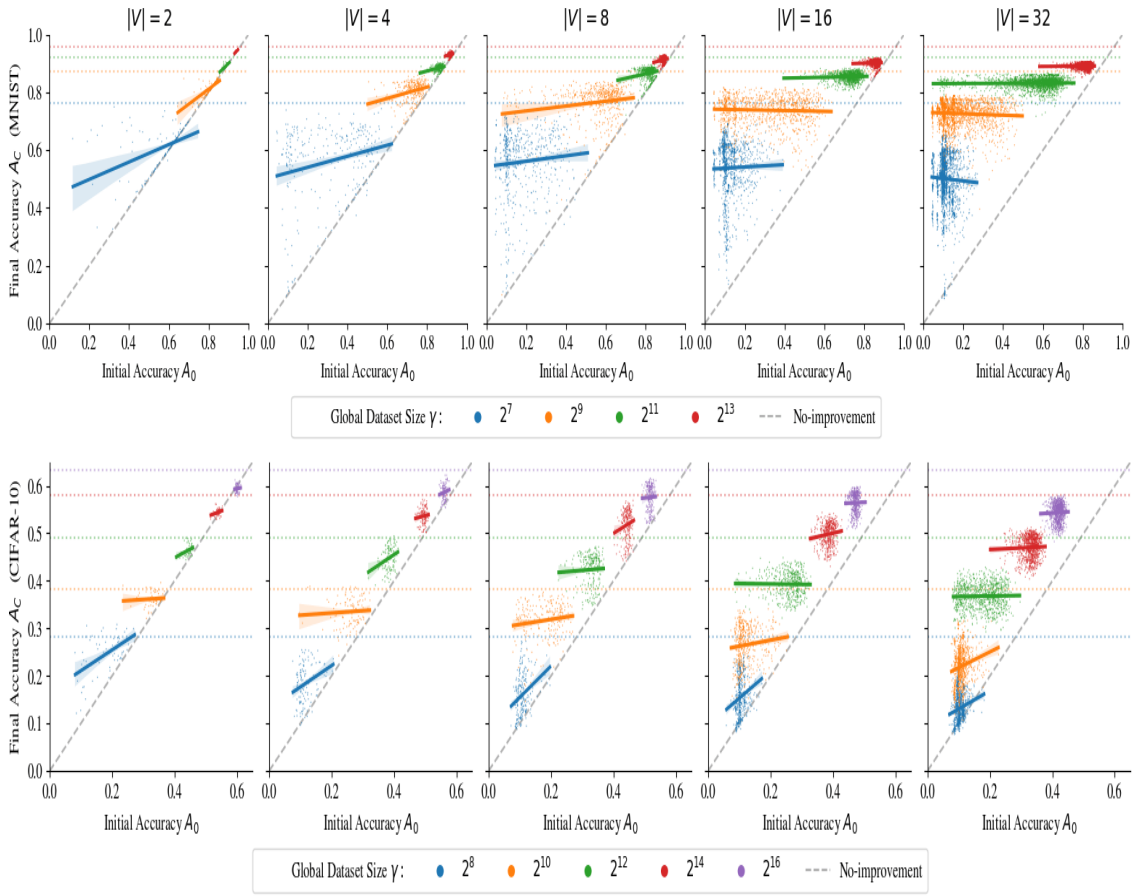


Figure 4.2: Relationship between global dataset size γ, initial accuracy A_0 (before the gossiping starts), and final accuracy A_C at gossip convergence for a varying number of nodes in the system over 15 repeated experiments. Each point in the scatter plot represents the initial and final accuracies for one user in the system. For each global dataset size γ and number of nodes $|V|$, a linear regression line is shown and surrounded by a 95% confidence band computed using bootstrapping. The "no-improvement" dashed line represents when a node's accuracy at convergence is the same as its accuracy before gossiping. The dotted lines represent the average accuracy of a centralized model trained on the global dataset.

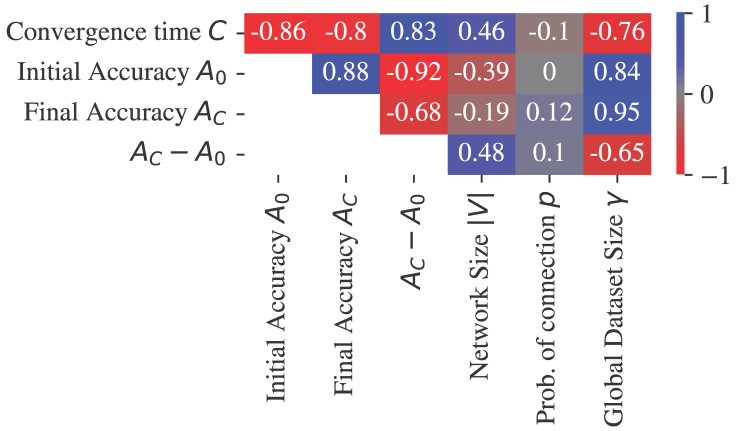


Figure 4.3: Spearman's correlation coefficient matrix between system parameters (network, dataset) and performance metrics (accuracy, convergence) for the MNIST dataset results. Each correlation coefficient belongs to a 95% confidence interval computed using the Fisher transform, with a difference between the upper and lower bound always < 0.022.

Figure 4.4 shows that larger global datasets reduce convergence time on average, that larger network sizes $|V|$ increase the convergence time average and standard deviation, and that models with higher initial accuracy tend to converge faster. This evidence matches what is shown by the correlation coefficients in Figure 4.3. However, stratifying the data by the size of the global dataset reveals that the initial accuracy does not influence the overall system's convergence speed, as evidenced by the negligible slope observed in the regression lines. Figure 4.5 shows in the top row that, for a large range of different global dataset sizes, the probability of connection has a very limited impact on the convergence speed, as the associated ECDFs are very close and their confidence intervals often overlapping. Conversely, Figure 4.5 shows in the bottom row that, independently from the probability of connection, the global dataset size has a much larger impact on convergence time, as the ECDFs significantly shift to lower values for larger dataset sizes. Compared to Figure 4.4, Figure 4.5 stratifies the convergence time analysis by the probability of connection, highlighting its limited impact on the metric.

4.4 Conclusion

We assessed Gossip Learning in a class of dynamic networks modeled by Time-Varying Graphs, advancing the knowledge in the field, which mainly targeted static networks so far. Our results show that Gossip Learning significantly enhances the

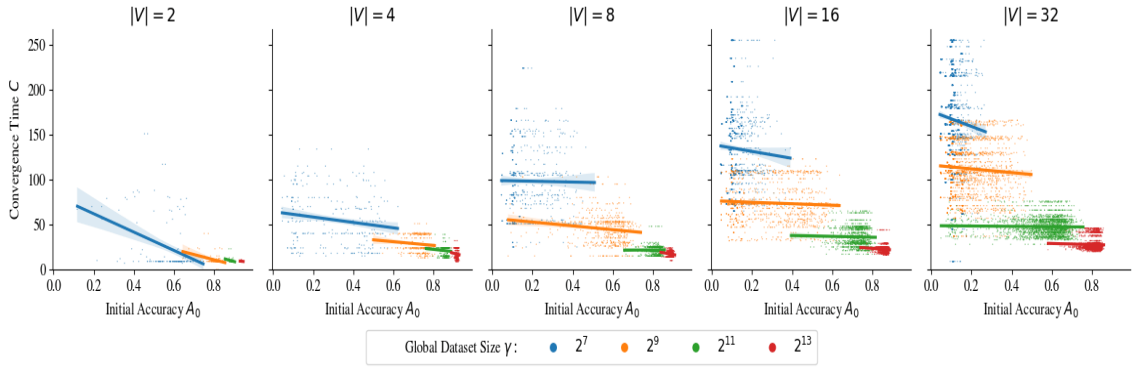


Figure 4.4: Scatter plot of convergence time C against initial accuracy A_0, for the MNIST dataset. Each point represents one user in the system. We show a linear regression line for the points belonging to each global dataset size γ and number of nodes $|V|$, surrounded by a 95% confidence band computed with bootstrapping.

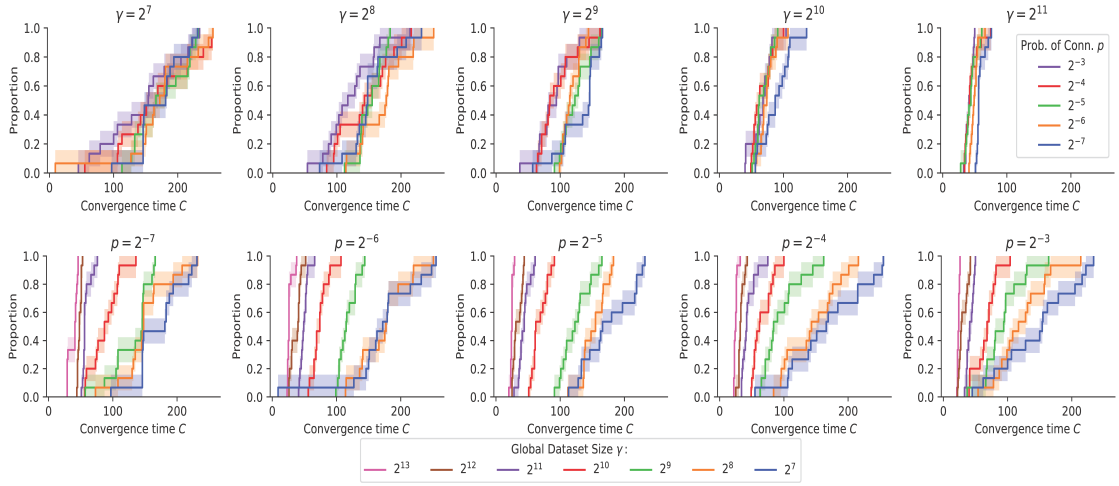


Figure 4.5: Empirical Cumulative Distribution Functions (ECDFs) of convergence time C for $|V| = 32$ nodes and MNIST dataset, at different values of global dataset size γ and probability of connection p. Confidence intervals are Kolmogorov-Smirnov bounds at 95% level.

average model accuracy and convergence time compared to local learning in networked systems whose topology varies over time and when communication capacity between nodes is constrained. As the number of network nodes increases, Gossip Learning's model accuracy gradually approaches that of the more communication-intensive classic Federated Learning (Figure 4.1) and proportionally outperforms local training (Figures 4.1 and 4.3), highlighting Gossip Learning's scalability and information-spreading capability (Figure 4.2). As the network connectivity (i.e., the inter-node connection probability) increases, Gossip Learning's model accuracy approaches that of more communication-intensive distributed learning schemes such as Federated Learning (Figure 4.1), showing Gossip Learning's communication-overhead efficiency. We show that even with a tiny inter-node connection probability (e.g., 2^{-7}), Gossip Learning's model accuracy is near-optimal in large-scale scenarios. For a fixed global dataset size, Gossip Learning converges at a similar speed of more communication-intensive schemes such as Federated Learning, regardless of initial model accuracy (Figure 4.4) or inter-node connection probability (Figure 4.5). We plan to extend this work by analyzing the impact of dependent connectivity patterns on model accuracy and convergence time.

Bibliography

[1] Abdul Aziz Alkathiri, Lodovico Giaretta, Sarunas Girdzijauskas, and Magnus Sahlgren. Decentralized word2vec using gossip learning. In *23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*, 2021.

[2] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 473–481. PMLR, 2018.

[3] Augustin Chaintreau, Abderrahmen Mtibaa, Laurent Massoulie, and Christophe Diot. The diameter of opportunistic mobile networks. In *Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, 2007.

[4] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Vehicle position nowcasting with gossip learning. In *IEEE WCNC*, pages 728–733. IEEE, 2022.

[5] István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

[6] Chengxi Li, Gang Li, and Pramod K. Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147, 2022.

[7] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[8] Nathalie Majcherczyk, Nishan Srishankar, and Carlo Pinciroli. Flow-FL: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. In *IEEE ICRA*, pages 8836–8842, 2021.

[9] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358, 2017.

[10] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[11] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[12] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[13] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), mar 2020.

[14] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully decentralized joint learning of personalized models and collaboration graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 864–874. PMLR, 2020.

CHAPTER 5

Exploring Gossip Learning in Dynamic Networks through Time-Varying Graphs

This chapter is based on a journal version planned for submission to IEEE TMC. An earlier version published in IEEE Annual Congress on Artificial Intelligence of Things (AIoT), Pages: 53-59. Date: July 2024. Publisher: IEEE. ISBN: 979-8-3503-9229-6.

Abstract– Gossip Learning (GL) has emerged as a promising paradigm for decentralized machine learning, offering scalability, robustness, and privacy preservation. Yet, most existing studies assume static connectivity or trace-driven mobility, failing to capture the dynamic and evolving nature of real-world networks. Consequently, the feasibility and performance trade-offs of GL under realistic dynamic conditions remain poorly understood.

This work addresses this gap by introducing novel classes of Time-Varying Graphs (TVG) that extend classical random graphs—Erdős-Rényi (ER), Barabási-Albert (BA), and Stochastic Block Model (SBM)—to dynamic settings through controlled edge persistence and node churn mechanisms. These TVG enable systematic exploration of the impact of temporal network evolution on decentralized learning, while abstracting away application-specific biases.

Extensive simulations reveal that increasing connectivity or dynamicity does not always enhance GL performance. Instead, optimal performance arises from balancing exploration (through dynamicity) and exploitation (through stable links). Sparse networks benefit from higher dynamicity to promote information diversity, whereas denser networks achieve superior results with moderate edge persistence.

Furthermore, while high churn introduces variability, it does not hinder convergence; temporary performance drops are primarily caused by the arrival of untrained nodes, rather than departures. Overall, our findings demonstrate that GL can approach the performance of centralized learning across a wide range of dynamic conditions, providing new insights into the design of robust decentralized learning systems in real-world environments.

5.1 Introduction

Distributed ML [28] has become a core enabler of scalable, real-time intelligence across various domains, including bioinformatics [20], transportation [11], and social networks [29]. By distributing the computation across multiple agents, these systems can process large data volumes efficiently while preserving locality. This is particularly critical in Artificial Intelligence of Things (AIoT) environments, where nodes generate vast streams of data and must operate under real-time and privacy constraints [27]. However, many distributed learning frameworks still rely on centralized coordination or infrastructure-heavy communication, limiting their scalability and resilience in dynamic settings.

To address these challenges, fully decentralized approaches have gained increasing attention. One such paradigm is GL [22, 12], which facilitates model training via peer-to-peer exchanges of models among agents. GL operates without a central server and scales naturally with the number of participants, as each node contributes its own data, compute, and communication resources.

Recent years have seen numerous GL schemes proposed for various architectures [31, 12] and applications [16, 6, 2]. However, these approaches often assume static network topologies—such as ring [18] or mesh [26, 31, 4]—which do not reflect the realities of mobile and volatile networks. While convergence of GL in static settings is well-documented [12], the same cannot be said for networks characterized by mobility and churn, such as V2V networks or robot swarms.

Some studies explore GL under dynamic conditions, including fully connected [19] and vehicular network topologies [8, 7]. Although they demonstrate that GL can converge even under trace-driven mobility, they are typically limited to specific scenarios and lack generalizability. These works do not provide a systematic understanding of how network parameters, such as connectivity, edge persistence, and

churn, affect learning behavior. Without this understanding, it remains unclear how to optimize GL for performance and robustness in diverse dynamic environments.

The study in [5] provides a first step by analyzing GL on random TVGs, focusing on how connectivity and scale influence convergence. However, it assumes a fixed number of agents and neglects node churn. Similarly, [25] and [23] investigate topological influences and failure robustness in decentralized federated learning, but only in static or limited settings.

Crucially, existing literature often assumes edge independence—ignoring the persistence of links over time—and omits node churn. These omissions limit the applicability of results to real-world systems, where communication links have memory and node populations are in flux.

In this paper, we address these gaps by introducing a generalized framework for evaluating GL in time-varying networks that incorporate both edge persistence and node churn. Our key contributions are:

- We propose TVER, TVBA, and TVSBM, time-varying extensions of classical ER, BA, and SBM graph models, which capture realistic dynamicity by modeling edge memory and churn dynamics.
- We develop a principled evaluation framework for GL on these TVG, decoupled from scenario-specific traces, allowing broad generalization across dynamic settings.
- Through extensive simulations on two benchmark datasets and diverse network conditions (e.g., sparse vs. dense graphs), we analyze how system parameters such as average degree, edge persistence, and churn affect convergence, accuracy, and learning stability.
- Our findings reveal that TVBA graphs exhibit strong resilience under varying connectivity and edge persistence, while TVER and TVSBM benefit from moderate dynamicity to mitigate performance loss. We also challenge the assumption that more connectivity or churn constantly improves performance, showing instead that optimal learning depends on achieving a balance between exploration and exploitation. High churn introduces temporary fluctuations but does not inherently hinder the learning process.

These results provide actionable insights for designing GL algorithms that are

robust to volatility and adaptable to the structural and temporal characteristics of real-world networks.

The rest of this paper is organized as follows. Related works are discussed in Section 5.2. Section 5.3 defines the system model, followed by a review of GL fundamentals in Section 5.4. The proposed TVG models are introduced in Section 5.5, with experimental setup in Section 5.6 and results in Section 5.7. Finally, Section 5.8 concludes the paper and outlines directions for future work.

5.2 Related works

Distributed learning has become a cornerstone of modern ML systems, enabling scalability, resilience, and privacy preservation across diverse environments [14, 17, 9, 26]. By distributing the learning process across multiple agents, distributed architectures overcome limitations of traditional centralized systems, such as communication bottlenecks and vulnerability to single points of failure [14].

Despite these advances, conventional distributed frameworks often rely on infrastructure-based or centralized aggregation mechanisms, which constrain scalability and robustness in dynamic or resource-constrained environments. This limitation has driven growing interest in serverless, fully decentralized approaches, with GL emerging as a promising paradigm [22]. In GL, model updates are exchanged peer-to-peer among neighboring agents, enabling decentralized knowledge transfer without centralized coordination. Various GL algorithms have been proposed for different learning architectures and applications [12, 11, 6], demonstrating resilience to heterogeneity and asynchrony.

Several studies have highlighted that the performance of GL critically depends on the spatio-temporal patterns of information exchange among nodes [12]. For instance, [7] investigates GL in vehicular networks, focusing on how the number of models merged per round affects learning performance. Although valuable in its specific context, this work relies on trace-based mobility. It thus cannot disentangle the effects of network dynamics from those of environmental and application-specific factors.

To draw more generalizable conclusions, recent research has moved toward a graph-theoretical modeling approach, representing agents as nodes and their communications as edges. This abstraction enables a systematic evaluation of how

topological properties, such as heterogeneity, clustering, degree distribution, and dynamic connectivity, impact the robustness and efficiency of decentralized learning.

[24, 23] investigate the resilience of decentralized learning under node failures and data corruption scenarios. They demonstrate, using BA graphs, that connectivity loss impacts learning accuracy more severely than data loss, and that knowledge retention mechanisms can enhance robustness. However, these studies are limited to static topologies and do not account for the effects of evolving network structures or churn, which are characteristic of real-world systems.

[25] broadens this perspective by analyzing how different static network topologies—ER, BA, and SBM—influence decentralized learning. The results highlight that topological features, such as the presence of hubs or community structures, can significantly impact convergence speed, final model accuracy, and communication efficiency. Still, the analysis remains limited to static graphs and does not account for dynamic reconfiguration or mobility-driven changes in connectivity.

A more dynamic view is proposed in [5], where GL performance is evaluated on a time-varying extension of the ER model. This work demonstrates that GL can achieve robust accuracy even under dynamic connectivity changes, offering essential insights into its potential for mobile and unstable environments. Nonetheless, the model assumes a constant number of nodes over time, thereby excluding churn effects—a crucial aspect in real-world systems where agents frequently join and leave the network.

In this work, we fill this gap by introducing three families of dynamic graphs, TVER, TVBA, and TVSBM, that jointly model memory, node churn, and different structural properties (randomness, scale-free behavior, and community structure). Unlike previous studies, our models allow both the connectivity patterns and the node set to evolve over time while preserving key features of the corresponding static counterparts. This enables a systematic evaluation of how network dynamics affect GL performance across a wide range of scenarios, leading to a deeper and more generalizable understanding of decentralized learning in realistic dynamic environments.

5.3 System model

We consider a set of mobile devices (e.g., smartphones, UAVs, connected vehicles)—hereafter referred to as nodes—that move within a region of space and can establish direct, peer-to-peer wireless connections whenever they are within transmission range of each other.

Time is divided into discrete slots, with $t \in \mathbb{N}$ representing the t-th time slot. We model the resulting network as a TVG, represented by a sequence of graphs, where each graph captures the instantaneous network topology at a given time slot. Specifically, we assume that the connectivity graph remains constant within each time slot but may evolve between slots.

Let V_t denote the set of nodes present at time t, and E_t the set of edges, where an edge $e_{i,j} \in E_t$ exists if nodes v_i and $v_j \in V_t$ can directly communicate via a bidirectional wireless channel. The TVG is thus defined as: $G = \{G_t = (V_t, E_t) : t \in \mathbb{N}\}$.

We distinguish between two types of systems based on node dynamics. In a *closed system*, the set of nodes remains constant over time, that is, $V_t = V_{t-1} \quad \forall t \in \mathbb{N}^*$. Conversely, in an *open system*, nodes may join or leave the network, resulting in $V_t \neq V_{t-1}$ in general. In the open system, we assume that node arrivals and departures follow stationary processes with equal mean rates, denoted by λ, ensuring that the expected number of active nodes remains constant over time.

The presence of an edge between two nodes v_i and v_j at a given time slot may vary across time slots, depending on the network evolution dynamics. In general, $\forall t \in \mathbb{N}^*$, we have $E_t \neq E_{t-1}$.

Each node maintains a local instance of a ML model, which shares the same architecture across all nodes. Each node also possesses a private local dataset used to train its model instance. In the following sections, we describe how nodes collaboratively train their models through opportunistic peer-to-peer exchanges, without relying on any centralized parameter server or coordinator, thus enabling fully decentralized learning across a dynamic network.

The main notations used throughout this work are summarized in Table 5.1.

Table 5.1: List of symbols and their descriptions.

Symbol	Description
t	Time slot label
V	Set of mobile nodes (vertices)
$\|V\|$	Number of mobile nodes
γ	The size of global dataset
E	Set of edges between nodes
$\|E\|$	Number of presented edges between nodes
$\tilde{E}$	Set of all possible edges in graph
N	Maximum number of edges in the graph
k	Average node degree in the graph
v_i	The i-th node in the graph, where $i \in \{1, 2, 3, \dots\}$
$e_{i,j}$	The edge which connect nodes v_i and v_j
λ	Nodes' arrival and departure rate
p_2	Edge persistence probability in all proposed TVGs
p_1	Initial edge creation probability in TVER
p_{in}	Initial intra-community edge creation prob in TVBSM
p_{out}	Initial inter-community edge creation prob in TVBSM
B	Total number of blocks in TVBSM
n_b	Number of nodes per block in TVBSM

5.4 Basics of Gossip Learning Operation

In this work, we adopt a decentralized GL protocol inspired by [8], where each node maintains its own local model, also referred to as a meta-model, that it uses for prediction tasks and updates throughout the learning process. Thus, the number of meta-models in the system equals the number of active nodes, and model personalization is implicitly embedded into the learning framework. While our experiments are based on a specific implementation, the analytical methodology and evaluation framework apply to a broad class of decentralized learning schemes.

All nodes present in the region at time $t = 0$, as well as those that arrive at $t > 0$, are initially endowed with an identical default model instance. This initial model may be pre-trained or reflect prior knowledge; however, in our study, we deliberately initialize all models with random parameters. This choice allows us to isolate and assess the effects of data locality, model exchange, and network structure on learning performance, without any bias from prior knowledge.

Starting from $t = 0$, the GL process at each node progresses through three synchronized phases:

- Local Training: Each node independently trains its local model instance using its private dataset. This step enables each node to encode local knowledge into its model parameters.
- Model Exchange: Each node broadcasts its current model instance to its one-hop neighbors—i.e., those nodes it can directly communicate with in the current time slot.
- Model Merging: Upon receiving models from its neighbors, each node aggregates them with its model using a weighted averaging strategy, thereby forming a new meta-model. This procedure is similar in spirit to the *FedAvg* algorithm [21], but operates in a decentralized, pairwise manner.

These three phases are repeated at each time slot until a stopping condition is met, such as reaching a target accuracy or observing negligible performance improvement over a predefined number of slots.

This decentralized, peer-to-peer learning protocol is particularly well-suited for dynamic network environments, as it allows each node to autonomously learn from

its data while opportunistically incorporating information from nearby peers, without relying on any centralized server or coordination mechanism.

5.5 Time-Varying Graphs (TVG)s

GL schemes are strongly influenced by the spatio-temporal patterns of information exchanges among nodes. To enable a scenario-independent, systematic, and controlled evaluation of how network dynamics affect decentralized learning, we propose three classes of TVG: extensions of the classical ER [10], BA [3], and SBM [13] graphs.

All proposed TVG incorporate two key dynamic properties: memory and node churn, offering a more realistic representation of real-world dynamic systems where the future connectivity depends on the past network state, and nodes can join or leave the network.

The edge persistence probability p_2 governs the memory: when $p_2 = 0$, the network is memoryless, and edges are fully rewired at each time slot; when $p_2 = 1$, the network remains static. Intermediate values $0 < p_2 < 1$ allow partial memory, enabling gradual network evolution. Node churn is modeled by an arrival and departure rate λ, ensuring a statistically stable average network size over time while accommodating dynamic participation of nodes.

Importantly, the proposed TVG are constructed to preserve the key structural properties of their static counterparts—such as average node degree, diameter, and clustering coefficient—even in the presence of churn and memory. This design enables a controlled and realistic evaluation of decentralized learning processes under dynamic network conditions without fundamentally altering the underlying topological characteristics.

In the following subsections, we detail the construction and evolution processes for each TVG class.

5.5.1 Time-Varying Erdős-Rényi (TVER)

The classical ER model [10] defines a static random graph where each pair of nodes is connected with a fixed probability. To capture realistic network dynamics, we extend this model into a time-varying framework, referred to as TVER. TVER graphs pro-

vide a simple yet powerful abstraction for homogeneous wireless environments, such as ad-hoc networks with uniformly distributed nodes, where connectivity evolves over time due to node mobility and intermittent communications.

The TVER model incorporates both memory and churn, governed by the following parameters:

- *Arrival edge creation probability* (p_1): The probability that an edge exists between two nodes at time slot $t = 0$ or when a new node arrives at later time slots: $\mathbb{P}[e_{i,j} \in E_0] = p_1$, $\mathbb{P}[e_{i,j} \in E_t \mid v_i \text{ or } v_j \notin V_{t-1}] = p_1$.
- *Edge persistence probability* (p_2): The probability that an edge present at time $t-1$ persists at time t, provided both nodes remain in the network: $\mathbb{P}[e_{i,j} \in E_t \mid e_{i,j} \in E_{t-1}, v_i, v_j \in V_t \cap V_{t-1}] = p_2$.
- *Post-arrival edge creation probability* ($(1-p_2)p_1$): The probability that a new edge forms between two existing nodes that did not previously share a connection: $\mathbb{P}[e_{i,j} \in E_t \mid v_i, v_j \in V_{t-1} \text{ and } e_{i,j} \notin E_{t-1}] = (1-p_2)p_1$.

At $t = 0$, a static ER graph is generated based on p_1. For $t > 0$, existing edges are preserved or rewired according to p_2, and newly arriving nodes form connections according to p_1. Departed nodes and their incident edges are removed. The complete evolution is formalized in Algorithm 9.

Assuming a stationary regime (equal mean arrival and departure rates λ), the expected number of edges $\mathbb{E}[|E|]$ stabilizes and is given by: $\mathbb{E}[|E|] = p_1 \cdot N$, where $N = \binom{n}{2}$ is the maximum number of possible edges among n nodes. Therefore, the average node degree k satisfies: $k = p_1(n-1)$.

Importantly, the TVER model preserves the key topological properties of the static ER graph—including average node degree, clustering, and diameter—while enabling controlled dynamic evolution over time.

5.5.2 Time-Varying Barabási-Albert (TVBA)

The BA model [1] generates scale-free networks exhibiting power-law degree distributions, where a few highly connected nodes (hubs) dominate the structure. Extending this model to dynamic settings, we introduce the TVBA graph. TVBA graphs realistically represent heterogeneous networks, such as social networks, where hubs emerge and connectivity evolves over time.

Algorithm 9 TVER Graph Evolution Procedure. The notations are described in Table 5.1

function ER(V, p_1)
  Initialize $E = \emptyset$
  for each node pair (v_i, v_j) where $i \neq j$ **do**
    With probability p_1, add edge $e_{i,j}$ to E
  return $G(V, E)$

function DYNER($V_{t-1}, E_{t-1}, V_t, p_1, p_2$)
  Initialize $E_t = \emptyset$
  for each node pair $(v_i, v_j) \in V_t \cap V_{t-1}$ **do**
    if $e_{i,j} \in E_{t-1}$ **then**
      With probability p_2, retain $e_{i,j}$ in E_t
    With probability $(1 - p_2)p_1$, create new edge $e_{i,j}$
  for each new arrival node $v_i \in V_t$ **do**
    for each node $v_j \in V_t, j \neq i$ **do**
      With probability p_1, add edge $e_{i,j}$
  return $G_t(V_t, E_t)$

procedure GENERATETVER(V_0, p_1, p_2)
  $G_0 \leftarrow$ ER(V_0, p_1)
  for each time slot $t > 0$ **do**
    Update V_t by processing arrivals and departures at rate λ
    $G_t \leftarrow$ DYNER($V_{t-1}, E_{t-1}, V_t, p_1, p_2$)

The generation of a TVBA graph is based on two parameters:

- *Preferential attachment probability* (h): Controls the extent to which newly formed edges are influenced by node degree. When $h = 0$, pure preferential attachment governs new connections; when $h = 1$, edges are assigned uniformly at random, reducing the graph to an ER structure.
- *Edge persistence probability* (p_2): Determines the likelihood that existing edges persist across time slots, introducing memory into the network evolution.

At initialization ($t = 0$), a static BA graph is created using a preferential attachment mechanism formalized by:

$$\mathbb{P}\big[(v_j, v_i) \in E \,\big|\, v_i \in \mathcal{A}\big] = h\frac{1}{|\mathcal{A}|} + (1-h)\frac{\deg(v_i)}{\sum_{\forall v \in \mathcal{A}} \deg(v)} \tag{5.1}$$

where $\mathcal{A}$ denotes the set of nodes already connected to the network, and $\deg(v_i)$ is the degree of node v_i, where $v_i \in \mathcal{A}$. For $t > 0$, existing edges are retained with probability p_2. Nodes with degrees lower than average node degree k form new edges according to the preferential attachment rule, ensuring that the graph maintains its scale-free characteristics while dynamically evolving. Newly arriving nodes are integrated similarly through preferential attachment.

The detailed evolution process is presented in Algorithm 10.

The TVBA model retains the defining properties of scale-free networks, such as heavy-tailed degree distributions and the emergence of hubs, while supporting memory and churn, thus offering a versatile tool for analyzing decentralized learning in dynamic, heterogeneous environments.

5.5.3 Time-Varying Stochastic Block Model (TVSBM)

The third class of TVGs considered in this work extends the classical SBM [13]. SBM graphs are widely used to model networks with community structures, where nodes are grouped into blocks and edge probabilities depend on block membership. We introduce the TVSBM graph to capture both temporal edge persistence and dynamic block interactions, reflecting the evolution of real-world community-based networks.

The parameters defining a TVSBM graph are:

Algorithm 10 TVBA Graph Evolution Procedure. The notations are described in Table 5.1.

function BA(V, k, h)
  Initialize $E = \emptyset$
  $\mathcal{A} \leftarrow$ randomly select $k + 1$ nodes from V
  for each node pair $(v_i, v_j) \in \mathcal{A}$ **do**
    Add edge $e_{i,j}$
  $\mathcal{B} \leftarrow V \setminus \mathcal{A}$
  for each $v_i \in \mathcal{B}$ **do**
    Connect v_i to k nodes from $\mathcal{A}$ using preferential attachment
    Update $\mathcal{A} \leftarrow \mathcal{A} \cup \{v_i\}$
  return $G(V, E)$

function DYNBA(V_t, E_{t-1}, p_2)
  Initialize $E_t = \emptyset$
  for each node pair $(v_i, v_j) \in V_t$ **do**
    if $e_{i,j} \in E_{t-1}$ **then**
      Retain edge with probability p_2
  Identify nodes with $\deg(v_i) < k$
  Attach under-connected nodes using preferential attachment
  return $G_t(V_t, E_t)$

procedure GENERATETVBA(V_0, k, h, p_2)
  $G_0 \leftarrow$ BA(V_0, k, h)
  for each time slot $t > 0$ **do**
    Update V_t by processing arrivals and departures at rate λ
    $G_t \leftarrow$ DYNBA(V_t, E_{t-1}, p_2)

- *Number of blocks (B)*: The number of blocks (communities) into which nodes are partitioned. Each node, upon its arrival, is assigned uniformly at random to one of the B blocks, with probability $1/B$.

- *Initial intra-community edge creation probability (p_{in})*: The probability that an edge exists between two nodes within the same block at $t = 0$ or when a new node arrives:
  $\mathbb{P}[e_{i,j} \in E_0 \mid b_{v_i} = b_{v_j}] = p_{in}$,
  $\mathbb{P}[e_{i,j} \in E_t \mid (v_i \text{ or } v_j \notin V_{t-1}) \quad \text{and} \quad (b_{v_i} = b_{v_j})] = p_{in}$.

- *Initial inter-community edge creation probability (p_{out})*: The probability that an edge exists between two nodes in different blocks at $t = 0$ or upon node arrival:
  $\mathbb{P}[e_{i,j} \in E_0 \mid b_{v_i} \neq b_{v_j}] = p_{out}$,
  $\mathbb{P}[e_{i,j} \in E_t \mid (v_i \text{ or } v_j \notin V_{t-1}) \quad \text{and} \quad (b_{v_i} \neq b_{v_j})] = p_{out}$.

- *Edge persistence probability (p_2)*: The probability that an edge between two existing nodes persists from $t-1$ to t:
  $\mathbb{P}[e_{i,j} \in E_t \mid e_{i,j} \in E_{t-1} \quad \text{and} \quad v_i, v_j \in V_t \cap V_{t-1}] = p_2$.

- *Temporal intra-community edge creation probability* $(1-p_2)p_{in}$: The probability that a new intra-block edge forms at $t > 0$ between nodes already present:
  $\mathbb{P}[e_{i,j} \in E_t \mid (v_i, v_j \in V_{t-1}) \quad \text{and} \quad (b_{v_i} = b_{v_j})] = (1-p_2)p_{in}$.

- *Temporal inter-community edge creation probability* $(1-p_2)p_{out}$: The probability that a new inter-block edge forms at $t > 0$:
  $\mathbb{P}[e_{i,j} \in E_t \mid (v_i, v_j \in V_{t-1}) \quad \text{and} \quad (b_{v_i} \neq b_{v_j})] = (1-p_2)p_{out}$.

The graph evolution proceeds as follows: At $t = 0$, each node is assigned to a block, and edges are created according to p_{in} and p_{out}. For $t > 0$, existing edges may persist with probability p_2, while new edges form according to block memberships and the temporal probabilities $(1-p_2)p_{in}$ and $(1-p_2)p_{out}$. Newly arriving nodes are randomly assigned to blocks and establish connections accordingly. The full evolution process is detailed in Algorithm 11.

The expected number of edges at stationarity is: $\mathbb{E}[|E|] = \frac{n}{2}\left[(n_b - 1)p_{in} + (n - n_b)p_{out}\right]$, where $n_b = n/B$ denotes the average block size. Accordingly, the average node degree k is given by: $k = (n_b - 1)p_{in} + (n - n_b)p_{out}$.

Algorithm 11 TVSBM Graph Evolution Procedure. Table 5.1 defines the notations.

function SBM(V, p_{in}, p_{out}, B)
  Initialize $E = \emptyset$
  for each node $v_i \in V$ **do**
    Assign v_i to a block with probability $1/B$
  for each node pair (v_i, v_j) where $i \neq j$ **do**
    if $b_{v_i} = b_{v_j}$ **then**
      Add edge $e_{i,j}$ with probability p_{in}
    else
      Add edge $e_{i,j}$ with probability p_{out}
  return $G(V, E)$

function DYNSBM($V_{t-1}, E_{t-1}, V_t, p_{in}, p_{out}, p_2$)
  Initialize $E_t = \emptyset$
  for each node pair $(v_i, v_j) \in V_t \cap V_{t-1}$ **do**
    if $e_{i,j} \in E_{t-1}$ **then**
      Retain edge with probability p_2
    if $b_{v_i} = b_{v_j}$ **then**
      Create edge with probability $(1 - p_2)p_{in}$
    else
      Create edge with probability $(1 - p_2)p_{out}$
  for each new arrival node $v_i \in V_t$ **do**
    Assign v_i to a block with probability $1/B$
    for each node $v_j \in V_t, j \neq i$ **do**
      if $b_{v_i} = b_{v_j}$ **then**
        Create edge with probability p_{in}
      else
        Create edge with probability p_{out}
  return $G_t(V_t, E_t)$

procedure GENERATETVSBM($V_0, p_{in}, p_{out}, p_2, B$)
  $G_0 \leftarrow$ SBM(V_0, p_{in}, p_{out}, B)
  for each time slot $t > 0$ **do**
    Update V_t with arrivals and departures at rate λ
    $G_t \leftarrow$ DYNSBM($V_{t-1}, E_{t-1}, V_t, p_{in}, p_{out}, p_2$)

Thus, the TVSBM model offers a flexible and realistic framework to study the dynamics of decentralized learning in evolving networks with structured communities.

5.6 Experimental Setup

Following the insights gained from our proposed TVGs, we now detail the experimental framework designed to evaluate the performance of GL schemes under varying network dynamicity. Our experiments are carefully structured to ensure reproducibility and to isolate the effects of network evolution on collaborative learning.

5.6.1 Datasets and Data Partitioning

We assess the performance of GL schemes over the proposed TVGs by considering a set of V nodes performing an inference task. Two standard benchmarks are employed: MNIST [15] for handwritten digit recognition and FASHION-MNIST [30] for fashion item classification. While MNIST offers a relatively simple grayscale dataset, FASHION-MNIST presents more intricate patterns, providing a more challenging task. Both datasets consist of 10 classes and 28×28 pixel grayscale images.

For each experiment, we define a *global dataset* (γ) as a fixed-size subset of the reference dataset. The global dataset is partitioned equally among nodes, ensuring each node possesses a *local dataset* of size $\gamma/|V|$. In dynamic scenarios, node arrivals introduce new data points, while departing nodes remove their associated data, maintaining a constant average data volume equal to γ across time slots.

Local datasets are generated by randomly sampling without replacement from the global dataset, ensuring independent and identically distributed (i.i.d.) data across nodes. To assess model generalization, we employ a disjoint *global test set* comprising 30% of the original dataset size. This setup, inspired by [25], enables us to isolate the effects of network topology and dynamicity on the learning process.

5.6.2 Model Architecture and Training Parameters

Each node independently trains a CNN model with identical architecture and hyperparameters. The model employs categorical cross-entropy loss, a batch size of 32, momentum of 0.9, and a learning rate of 10^{-4}. Unless otherwise specified, local

training at each node proceeds for a single epoch per time slot, striking a balance between computational cost and responsiveness to network changes.

Simulations terminate upon reaching either a maximum of 300 time slots or performance convergence, defined as less than 0.5% improvement in average accuracy across 30 consecutive slots. Model accuracy (ratio of correct predictions) and loss (categorical cross-entropy) are recorded as primary performance metrics.

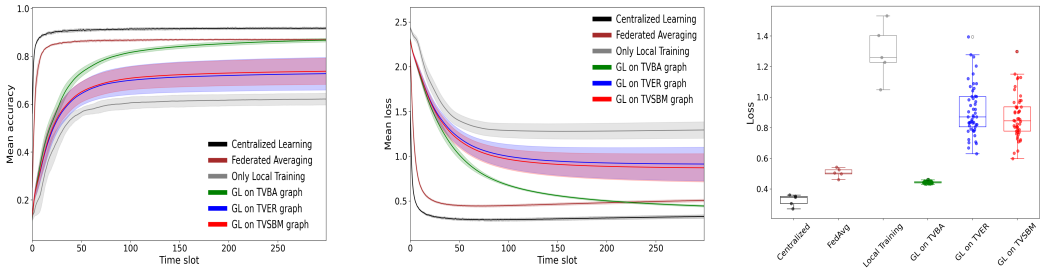
5.6.3 Network Configuration and Dynamics

Dynamicity in the network is modeled using two parameters: the edge persistence probability p_2 and the churn rate λ. Here, $p_2 = 1$ denotes a static network, whereas $p_2 = 0$ indicates complete topology rewiring at every time slot.

Node churn follows a Poisson point process with balanced arrival and departure rates, proportional to the initial node count $|V_0|$. This preserves the stationary nature of the network over time. Unless otherwise stated, we consider a network with $|V| = 50$ nodes, a setting that ensures a manageable trade-off between complexity and representativeness.

In TVSBM graphs, we fix the number of communities to five, yielding clear intra-community structures. A higher intra-community edge probability relative to inter-community connectivity mirrors real-world clustered networks [25] and maintains consistent community separation. To ensure a strong and interpretable community structure, we set the intra-community and inter-community connection probabilities to achieve a target average node degree k. Specifically, the intra-community connection probability is determined by: $p_{in} = \frac{k}{n_b - 1 + \frac{n - n_b}{n/(k+1)}}$. The inter-community connection probability p_{out} is then set as: $p_{out} = \frac{p_{in}}{n/(k+1)}$. This configuration ensures that intra-community connections are significantly more likely than inter-community ones, resulting in well-separated communities. It closely mirrors real-world network structures where nodes are densely connected within communities but sparsely connected across them [25].

To characterize the network, we monitor structural metrics. Degree Centrality measures the direct connectivity of nodes. Betweenness Centrality captures the extent to which nodes facilitate communication as intermediaries. The Maximum Shortest Path per node captures the longest shortest path from each node to any other reachable node, reflecting the network's reachability and resilience, assigning -1 to isolated nodes. These metrics provide a comprehensive understanding of how



(a) Mean accuracy over time slots (b) Mean loss over time slots (c) loss distribution-at convergence

Figure 5.1: Performance comparison of GL on the three proposed TVG models against centralized learning, FedAvg, and only local training. Experiments are conducted with $\gamma = 2000$, $k = 1$, $p_2 = 1$, and using the MNIST dataset. Shaded areas indicate 95% confidence intervals.

graph structure impacts GL performance under dynamic conditions.

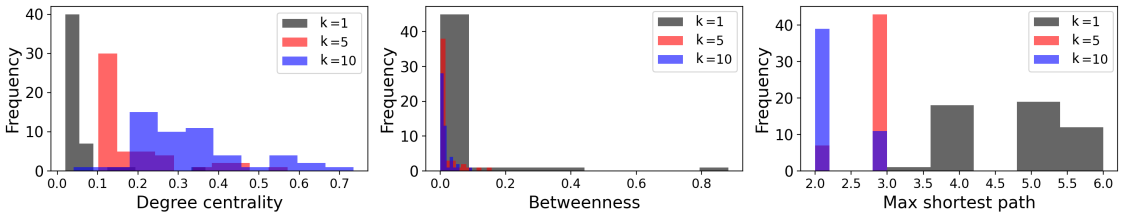
5.6.4 Baseline Learning Strategies

To contextualize the performance of GL, we compare it against three baseline learning schemes:

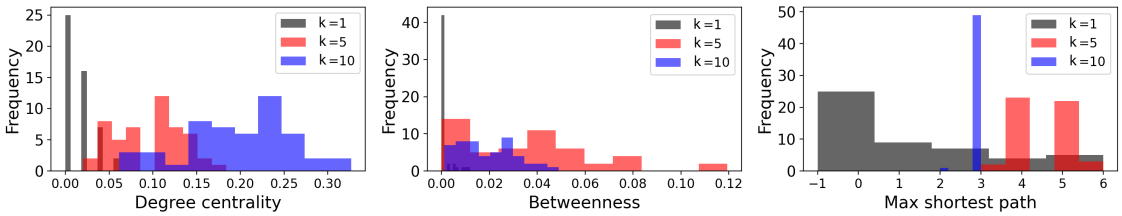
- *Centralized Learning (CL)*: A single centralized model trained on the entire global dataset.
- *Federated Averaging (FedAvg)* [21]: Clients train local models on their data and periodically send them to a server, which aggregates the models using a dataset-size-weighted average.
- *Only Local Training*: Each node trains exclusively on its local dataset without any form of collaboration.

5.7 Experimental Results and Discussion

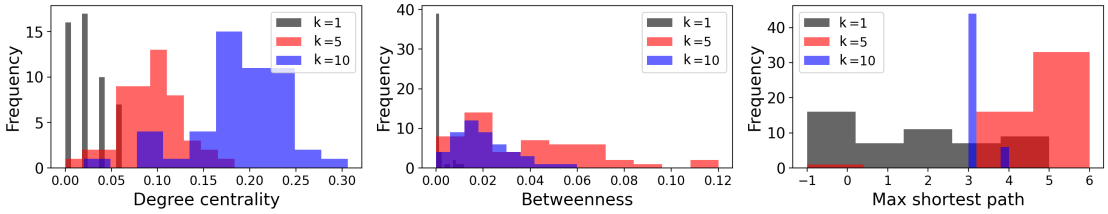
Figure 5.1 presents a comparative analysis of GL performance on the proposed TVG models against baseline methods. Figures 5.1(a) and (b) show that GL consistently outperforms Only Local Training across all graph models, despite operating over sparse networks with a mean node degree of $k = 1$. Notably, in TVBA graphs, the mean accuracy achieved by GL closely approaches that of FedAvg, with a small gap to centralized learning, highlighting the efficiency of scale-free structures for



(a) TVBA



(b) TVER

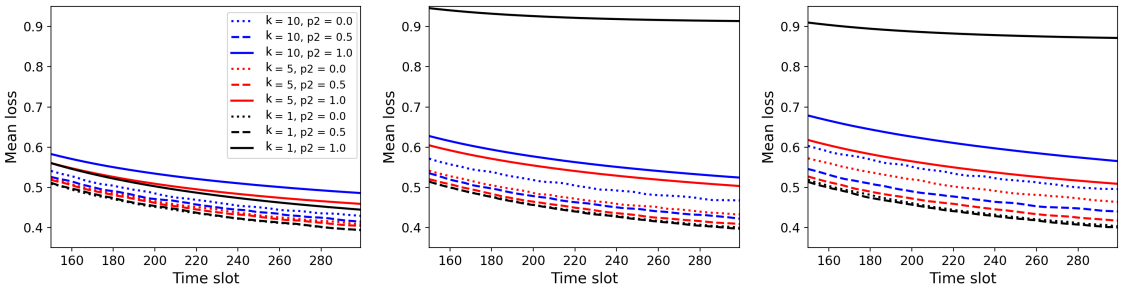


(c) TVSBM

Figure 5.2: Degree centrality, betweenness centrality, and maximum shortest path distributions for TVBA, TVER, and TVSBM graphs with different average node degrees $k \in \{1, 5, 10\}$, and $p_2 = 1$. This highlights the structural differences between the networks and their potential impact on the performance of GL algorithms.

decentralized training. The mean loss curves in Figure 5.1(b) reinforce this trend: GL over TVBA converges to significantly lower loss values compared to TVER and TVSBM, and achieves nearly comparable performance to FedAvg.

Figure 5.1(c) illustrates the distribution of loss values across nodes at convergence. GL operating on TVBA graphs exhibits a narrow loss distribution, indicating consistent performance across nodes. Conversely, in TVER and TVSBM graphs, the variance in loss is notably higher. This suggests that while some nodes benefit from efficient model aggregation, others may suffer from partial or complete isolation, resulting in degraded local performance. These trends reflect the influence of network topology on collaborative learning. The superior performance of GL on TVBA graphs suggests a more favorable structural property that enables faster and more robust information diffusion. In contrast, the more homogeneous and possibly fragmented structures of TVER and TVSBM can lead to connectivity issues and performance degradation.



(a) GL on TVBA (b) GL on TVER (c) GL on TVSBM

Figure 5.3: Impact of network topology, node connectivity (k), and dynamicity (p_2) on GL performance, evaluated on the MNIST dataset with $\gamma = 2000$. The figure reports the mean loss for GL over the last 150 time slots for three proposed TVG, highlighting how variations in k and p_2 affect model convergence and information flow.

To further explain the differences observed in baseline performance, we investigate the structural properties of the underlying networks in Figure 5.2. In TVBA graphs, Figure 5.2(a), nodes display higher degree centrality and lower maximum shortest paths even at low connectivity $k = 1$, facilitating efficient model diffusion and robustness against node failures. The existence of hubs with high betweenness centrality promotes the formation of short and redundant communication paths.

In contrast, TVER and TVSBM graphs, Figures 5.2(b) and (c), exhibit a more homogeneous structure with limited redundancy. For $k = 1$, nodes are weakly connected, leading to isolated nodes (maximum shortest path of -1 and fragmented information flow. As k increases, both TVER and TVSBM graphs improve their connectivity, reducing the maximum shortest path and increasing the number of communication pathways; however, they still remain less robust compared to TVBA graphs.

In the previous experiments, it remained unclear whether the observed performance differences among the proposed TVGs were primarily due to intrinsic topological features or to the limited information flow caused by sparse connectivity and staticity. To address this question, we further investigate the role of network connectivity by varying the average node degree k, and the role of dynamicity by adjusting the edge persistence probability p_2.

Figure 5.3 shows that increasing the node degree or introducing network dynamicity does not always uniformly improve GL performance. For instance, in the static case ($p_2 = 1$), GL on TVBA graphs achieves slightly better performance when $k = 1$,

whereas TVER and TVSBM graphs achieve their best performance around $k = 5$. As discussed earlier (see Figure 5.2), these configurations correspond to scenarios with higher betweenness centrality, indicating the presence of key bridging nodes that enhance information dissemination across the network. However, an excessive reliance on a few highly connected nodes can create vulnerabilities, where the failure or isolation of such nodes disrupts communication paths.

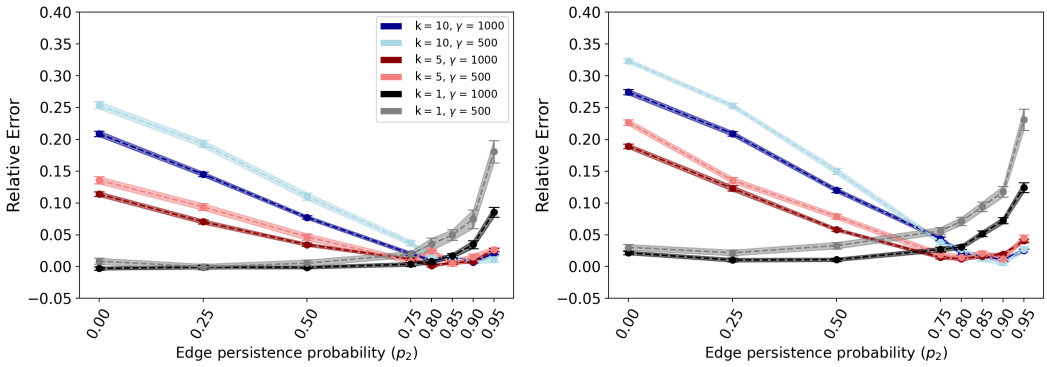
Importantly, Figure 5.3 highlights that network dynamicity significantly improves the performance of GL, especially in sparse networks (e.g., $k = 1$), by enabling better information mixing over time. When p_2 is reduced from 1 to lower values, nodes that were previously isolated or poorly connected can eventually exchange information through new temporary connections. This is particularly beneficial in TVER and TVSBM graphs, where the static maximum shortest path is higher. Conversely, for TVBA graphs, the benefit of dynamicity is marginal, as even the static topology already ensures short communication paths.

However, the figure also shows that excessive connectivity (e.g., $k = 10$) in highly dynamic networks ($p_2 = 0$) can lead to degraded performance. This phenomenon, known as over-averaging, occurs because frequent model exchanges cause premature convergence to suboptimal solutions by diluting the diversity of information. Thus, the best performance across all TVGs is generally achieved when balancing exploration and exploitation, particularly around $p_2 = 0.5$, which provides a controlled amount of new information while retaining valuable existing connections.

Overall, these results emphasize that optimal GL performance in dynamic networks requires carefully tuning both the degree of connectivity and the level of network memory, with the right balance depending on the underlying topology.

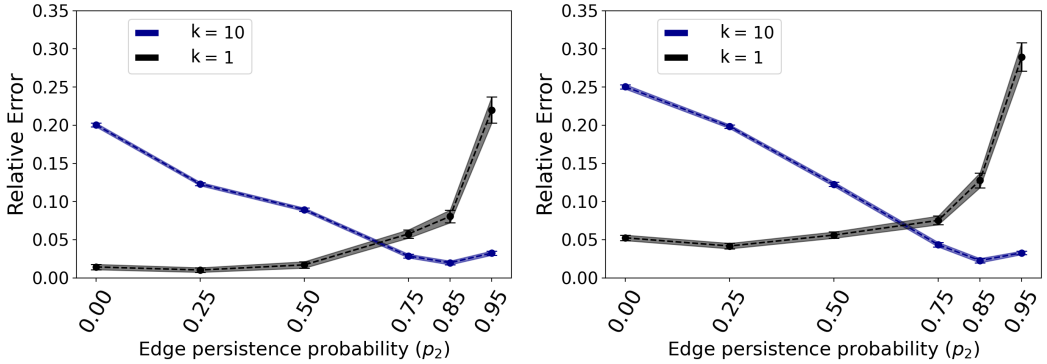
Following the analysis of connectivity and topology in closed networks, we extend our investigation to assess how varying levels of dynamicity influence the performance of GL. In particular, we focus on TVER and TVSBM, as TVBA already demonstrated resilience across different settings.

In this set of experiments, we explore a wide range of edge persistence probabilities (p_2) to systematically examine how the balance between exploration (dynamic connections) and exploitation (stable connections) affects learning efficacy. We also analyze the interplay between dynamicity, local dataset size, and task complexity, considering both the MNIST and Fashion-MNIST datasets.



(a) GL on TVER (b) GL on TVSBM

Figure 5.4: Relative error versus edge persistence probability (p_2) for different average node degrees (k) and global dataset sizes (γ) on the MNIST dataset. The relative error indicates the percentage variation in loss relative to the mean loss achieved by centralized learning. Shaded areas represent the 95% confidence intervals.



(a) GL on TVER (b) GL on TVSBM

Figure 5.5: Relative error versus edge persistence probability (p_2) for $k \in \{1, 10\}$ and $\gamma = 2000$ on the FASHION-MNIST dataset. Shaded areas represent the 95% confidence intervals.

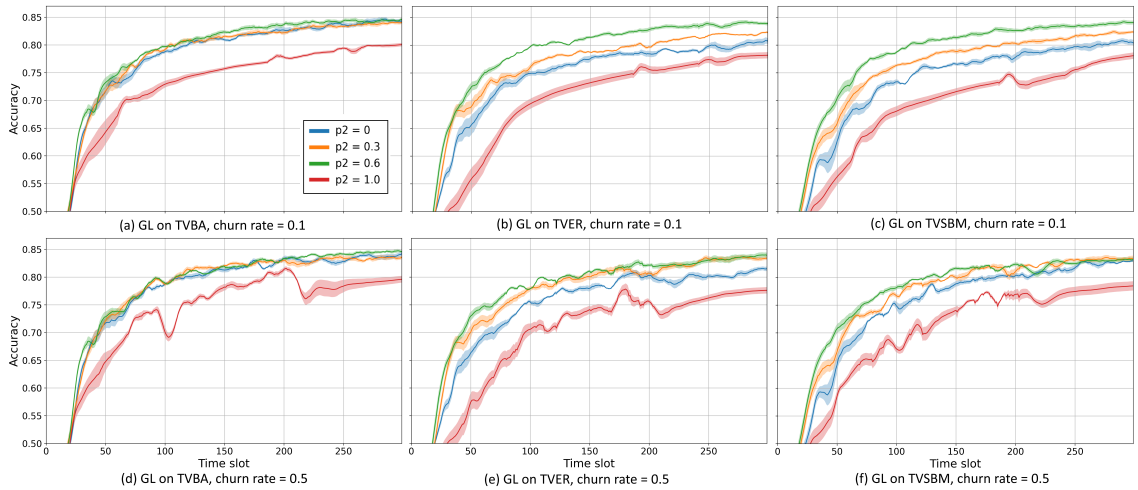


Figure 5.6: Comparison of GL algorithms on three proposed TVG under varying edge persistence probability (p_2) and churn rate (λ). Parameters are set to $k = 10$, and $\gamma = 500$. The first row presents results with $\lambda = 0.1$, and the second row with $\lambda = 0.5$. Shaded areas represent the 95% confidence intervals.

Figures 5.4 and 5.5 present the relative error, defined as

$$\text{Relative Error} = \frac{\text{Mean Loss} - \text{Centralized Loss}}{\text{Centralized Loss}},$$

evaluated as a function of p_2 for various node degrees k and global dataset sizes γ. Across all configurations, we observe that GL can achieve performance comparable to centralized learning by appropriately tuning p_2. In sparse networks (low k), optimal performance is attained at lower p_2 values, reflecting the importance of frequent topology changes for ensuring sufficient information mixing. Conversely, in more connected networks, the best results are obtained at intermediate p_2 values (around 0.75 to 0.85), where a balance between exploration and stability is maintained.

The results also reveal that smaller local datasets exacerbate sensitivity to dynamicity. In scenarios with limited local data ($\gamma = 500$), deviations from the optimal p_2 lead to larger fluctuations in performance. This indicates that limited data availability amplifies the importance of maintaining an appropriate dynamicity level for efficient model aggregation. In real-world dynamic networks, churn is inevitable. In this section, we analyze the effects of node arrivals and departures, combined with network dynamicity, on the performance of GL over the proposed TVG.

Figures 5.6 (a)-(f) illustrate the performance of GL across different churn regimes. At low churn rates ($\lambda = 0.1$), Figures 5.6 (a)-(c) show that the network remains relatively stable, and the learning process is not significantly disrupted. The models

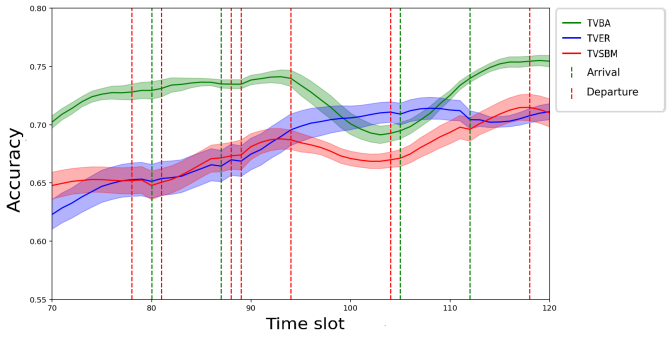


Figure 5.7: Performance variability of GL across three proposed TVG based on node arrival and departure events. Parameters are set to $k = 10$, $p_2 = 1$, and $\lambda = 0.5$. Shaded areas represent the 95% confidence intervals.

adapt smoothly to minor network variations without significant performance degradation.

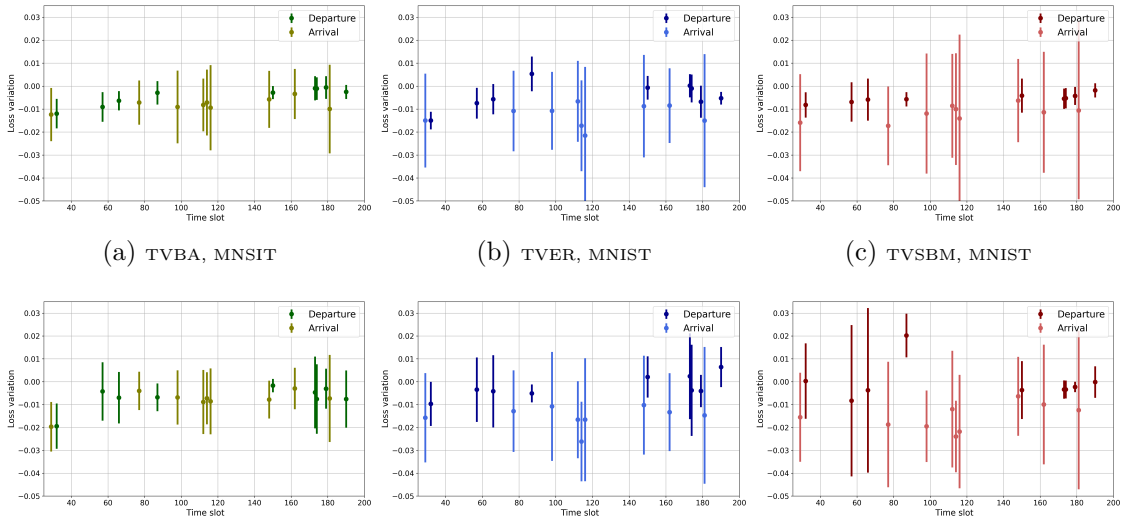
In contrast, at high churn rates ($\lambda = 0.5$), Figures 5.6 (d)-(f) indicate that although the overall learning process remains functional, larger fluctuations emerge. The arrival and departure of nodes introduce instability, necessitating more time for the system to reach consistent performance levels.

A critical observation is the poor performance of static networks ($p_2 = 1$) under churn. Without dynamic rewiring, static networks exhibit severe instability, as seen by high variance and performance collapses, particularly in Figure 5.6 (d). Static structures result in poor spectral properties, slow mixing times, and limited adaptability, thereby hindering the effective integration of new nodes and degrading global learning.

Interestingly, Figures 5.6 (e) and (f) show that dynamic networks (TVER and TVSBM) benefit from high churn when p_2 is moderate or low. In particular, TVSBM demonstrates remarkable robustness at $p_2 = 0$ and $p_2 = 0.3$, leveraging its community structure to assimilate new nodes effectively and maintain performance.

To gain deeper insights, Figure 5.7 examines the fine-grained impact of arrivals and departures over time. Overall, TVBA exhibits superior robustness to churn, thanks to the presence of highly connected hubs typical of scale-free networks. Although departures of hubs initially disrupt performance (notably after time slot 90), the network rapidly recovers through self-organizing dynamics, as formalized in Algorithm 10.

The TVER topology shows moderate sensitivity to node departures due to its more uniform degree distribution, while TVSBM demonstrates performance vari-



(a) TVBA, MNSIT (b) TVER, MNIST (c) TVSBM, MNIST

(d) TVBA, FASHION-MNSIT (e) TVER, FASHION-MNIST (f) TVSBM, FASHION-MNIST

Figure 5.8: Loss improvement across node churn events for GL over three proposed TVG. Loss improvement is defined as the difference between the mean loss at the previous time slot (a) and at the churn event time slot (b), i.e., $a-b$. Positive values indicate a loss reduction. Experiments are performed with $\lambda = 0.3$, $p_2 = 1$, and $k = 10$. Top row: MNIST dataset; bottom row: FASHION-MNIST dataset. Error bars represent the standard deviation at churn events.

ability depending on whether departures affect critical inter-community bridges or intra-community cohesion. The loss of bridging nodes notably increases clustering and isolates communities, thereby hindering the effective diffusion of models.

Finally, Figure 5.8 quantifies the instantaneous impact of node churn on performance by analyzing the change in loss at each event. Across all graphs and both datasets, node arrivals typically lead to a temporary degradation in mean performance (higher loss) and an increase in standard deviation. At the same time, departures do not significantly degrade performance, except in cases where the departing nodes serve critical roles in maintaining inter-community links or act as hubs. This behavior arises because newly arrived nodes initially lack trained knowledge and need time to synchronize with the network's learning state.

Through extensive simulations, we systematically evaluated the interplay between network structure, dynamicity, and churn on the performance of GL over diverse TVG. Our results highlight several key observations: (i) network topology critically shapes learning outcomes, with scale-free networks (TVBA) providing greater robustness; (ii) moderate dynamicity optimally balances exploration and exploitation, enhancing performance across varying datasets and connectivity regimes; (iii) churn, particularly at high rates, introduces variability but does not inherently

hinder convergence if sufficient dynamic connectivity exists; (iv) node departures, unless involving critical hubs or bridges, have limited detrimental effects, while node arrivals introduce temporary degradation. These insights contribute to a deeper understanding of how GL schemes operate under realistic dynamic conditions.

5.8 Conclusion

GL algorithms offer a promising approach for dynamic environments, leveraging scalability, privacy preservation, and decentralized computation. However, their performance trade-offs under dynamic network conditions remain poorly understood, due to prevalent assumptions of stable node connectivity or reliance on trace-based mobility patterns.

This work introduced a novel evaluation framework for GL schemes by proposing three classes of TVG: dynamic extensions of the classical ER, BA, and SBM graphs, incorporating node churn and edge persistence probability as key parameters governing network evolution.

Through extensive simulations, we provided critical insights into the interplay between network topology, dynamicity, and learning performance. Results show that in sparse and static networks, the underlying topology critically determines the effectiveness of GL, with TVBA graphs demonstrating notable resilience and achieving performance comparable to Centralized Learning. Enhancing connectivity and introducing moderate dynamicity effectively mitigates the limitations observed in TVER and TVSBM graphs. However, the assumption that higher node degree or increased dynamicity universally improves performance is challenged. Instead, optimal performance emerges from a careful balance between exploration (dynamicity) and exploitation (network persistence).

Furthermore, our findings reveal that low churn has minimal impact, while high churn induces fluctuations but does not inherently hinder learning, particularly in dynamic networks with strong connectivity. Notably, GL over TVSBM graphs benefit from high churn when edge persistence is low, leveraging the underlying community structure. While node arrivals introduce temporary performance degradation due to the initial lack of training, node departures generally have a limited negative impact, unless they involve critical nodes.

These results underscore the importance of jointly considering network topology,

connectivity dynamics, and churn in designing robust GL algorithms for real-world applications.

As future work, we aim to explore adaptive GL algorithms capable of dynamically adjusting key parameters (e.g., aggregation weights, learning rates) in response to network state changes. Additionally, extending our evaluation framework to incorporate heterogeneous data distributions among nodes would further enhance the applicability of decentralized learning in complex and large-scale dynamic environments.

Bibliography

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Abdul Aziz Alkathiri, Lodovico Giaretta, Sarunas Girdzijauskas, and Magnus Sahlgren. Decentralized word2vec using gossip learning. In *23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*, 2021.

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[4] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 473–481. PMLR, 2018.

[5] Antonio Di Maio, Mina Aghaei Dinani, and Gianluca Rizzo. The upsides of turbulence: Baselining gossip learning in dynamic settings. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '23, page 376–381, New York, NY, USA, 2023. ACM.

[6] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Vehicle position nowcasting with gossip learning. In *2022 IEEE WCNC*, pages 728–733, 2022.

[7] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. A gossip learning approach to urban trajectory nowcasting for anticipatory ran management. *IEEE Transactions on Mobile Computing*, pages 1–17, 2023.

[8] Dinani, Mina Aghaei and Holzer, Adrian and Nguyen, Hung and Marsan, Marco Ajmone and Rizzo, Gianluca. Gossip learning of personalized models for vehicle trajectory prediction. In *2021 IEEE WCNC*, pages 1–7, 2021.

[9] Anis Elgabli, Jihong Park, Amrit S Bedi, Mehdi Bennis, and Vaneet Aggarwal. Communication efficient framework for decentralized machine learning. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2020.

[10] P Erdős and A Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[11] Michel Gendreau, Gianpaolo Ghiani, and Emanuela Guerriero. Time-dependent routing problems: A review. *Computers & Operations Research*, 64:189–197, 2015.

[12] István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

[13] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[14] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

[15] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[16] Chengxi Li, Gang Li, and Pramod K. Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147, 2022.

[17] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[18] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[19] Nathalie Majcherczyk, Nishan Srishankar, and Carlo Pinciroli. Flow-FL: Data-driven federated learning for spatio-temporal predictions in multi-robot systems. In *IEEE ICRA*, pages 8836–8842, 2021.

[20] Micol Marchetti-Bowick, Junming Yin, Judie A. Howrylak, and Eric P. Xing. A time-varying group sparse additive model for genome-wide association studies of dynamic complex traits. *Bioinformatics*, 32(19):2903–2910, 06 2016.

[21] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[22] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[23] Luigi Palmieri, Chiara Boldrini, Andrea Passarella, and Marco Conti. Robustness of decentralized learning to nodes and data disruption. *arXiv preprint arXiv:2405.02377*, 2023.

[24] Luigi Palmieri, Chiara Boldrini, Lorenzo Valerio, Andrea Passarella, and Marco Conti. Exploring the impact of disrupted peer-to-peer communications on fully decentralized learning in disaster scenarios. In *2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–6. IEEE, 2023.

[25] Luigi Palmieri, Chiara Boldrini, Lorenzo Valerio, Andrea Passarella, and Marco Conti. Impact of network topology on the performance of decentralized federated learning. *Computer Networks*, 253:110681, 2024.

[26] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[27] Sung-Ho Sim and Yoon-Su Jeong. Multi-blockchain-based iot data processing techniques to ensure the integrity of iot data in aiot edge computing environments. *Sensors*, 21(10), 2021.

[28] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), mar 2020.

[29] Chi Wang, Jie Tang, Jimeng Sun, and Jiawei Han. Dynamic social influence analysis through time-dependent factor graphs. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 239–246, 2011.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[31] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. Fully decentralized joint learning of personalized models and collaboration graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 864–874. PMLR, 2020.

Energy-Efficient GL for Resource-Constrained Devices

CHAPTER 6

QoS-Aware Delivery of Vehicular Named Data in Fragmented Networks

This chapter is based on a journal version submitted to IEEE TMC. An earlier version published in IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Pages: 234–243. Date: August 2020. Publisher: IEEE. ISBN: 978-1-7281-7374-0.

Abstract– Many relevant vehicular network applications are information-centric in nature. However, the performance of current Named Data Networking (NDN) approaches in vehicular ad hoc networks suffers when the network is sparse, partitioned, and disconnected, e.g. due to channel volatility. Moreover, existing strategies for NDN-based content retrieval in these contexts do not allow for QoS-aware minimization of resource utilization, limiting their practical viability.

To address these issues, we propose DeepNDN, a communication strategy based on the synergy between NDN and probabilistic spatial content caching that enables content retrieval in fragmented and dynamic network topologies while meeting strict latency requirements in a resource-efficient manner. We propose a deep learning approach to DeepNDN management, based on locally tuning content caching and replication according to spatiotemporal patterns of node mobility and content requests and the cost of locally available resources. The numerical assessment of the proposed approach in several realistic scenarios shows that it is able to outperform a set of relevant baselines while achieving the target performance in a resource-efficient manner.

6.1 Introduction

The amount of data generated, exchanged, and consumed in vehicular networks is expected to increase exponentially in the near future [7], driven by the progressive adoption of Intelligent Transportation System (ITS) services and applications and by the diffusion of coordinated autonomous driving. In such a context, direct vehicle-to-vehicle (V2V) communications play a pivotal role, enabling low-latency data exchanges while offloading the cellular infrastructure from information exchanges with mainly local relevance. This includes, for instance, sensor and telemetry data from vehicles exchanged in the context of smart city applications or disaster response [3]. Additionally, bulky 3D models and point clouds are exchanged among vehicles to support autonomous driving, enhance situation awareness for road safety, or facilitate context-based Augmented Reality (AR) vehicular applications [24, 22].

In this work, we focus on the problem of delivering a content object (e.g., a warning message, a map update, or an element of a 3D model, to name a few) via direct V2V communications to those vehicles that request it within a given region of space, and some maximum delay. V2V communications can be inadequate for supporting such a service when relying on host-based protocols such as IP and point-to-point information exchanges [10, 55, 18]. This drawback is particularly evident in highly dynamic scenarios or those settings characterized by (locally) sparse spatial distribution of vehicles, resulting in fragmented and rapidly varying network topologies. Indeed, these features often bring unstable inter-vehicular connectivity, transmission errors, and high and unpredictable latencies, making it difficult to implement reliable and timely exchanges [10, 18, 29].

The Named Data Networking (NDN) [13] communication paradigm, a prominent implementation of Information-Centric Networking (ICN), has been proposed to overcome these shortcomings. It is based on in-network caching and information retrieval according to the content name instead of the host location. NDN takes advantage of content redundancy in the network to minimize the number of transmissions required to deliver a given content, potentially improving the effectiveness of content retrieval on the occurrence of network topology changes [49]. Several key issues, however, hamper NDN's efficient and successful use in V2V communications [27, 55]. Specifically, as current NDN approaches typically require the existence of paths between content producers and requesters, their applicability is restricted to connected components in network scenarios that are *fragmented*, i.e. in which at

any time instant on average, every node is connected with only a small subset of the other nodes in the region. At the same time, in highly dynamic scenarios, they are severely limited by the instability of those paths [12, 2, 45].

In sparse environments, where store-carry-and-forward is the primary mode of communication, several Delay-Tolerant Networking (DTN) schemes for location-based probabilistic content caching and delivery have been proposed, such as Floating Content (FC) [41, 20], hovering information [5], and others [35]. They allow tuning opportunistic content replication to minimize the probability of content disappearance from a given region (due to churn) and achieve a target delivery ratio. However, they do not support content retrieval with strict latency requirements, and they are very inefficient (implying many unnecessary content replications) as intra-cluster communication strategies, as well as whenever the content object is of interest to only a fraction of the nodes in the region [32].

A promising approach to address these limitations is to combine NDN for intra-cluster content exchanges (for resource efficiency and low latency) with DTN schemes (for robustness against network fragmentation and sparsity) [37, 21, 54, 27]. Although the majority of existing results are designed for single point-to-point exchanges, in networks with slowly changing, relatively stable topologies and mobility patterns, making them unsuitable for dynamic environments, and very inefficient whenever the same content is of interest to several users in the region.

In this paper, we consider a region of space in which users regularly turn into requesters for content objects, and in which infrastructure support (in the form of orchestration of content replication) is pervasive. All the aforementioned works leave the following key issues:

- How to implement a resource-efficient NDN content delivery service with tight delay constraints in a fragmented vehicular network to maximize direct V2V content exchanges?
- How to orchestrate the content delivery process to achieve a target performance (in terms of minimum delivery ratio and maximum latency) in a resource-efficient manner, while ensuring content persistence?

In this paper, we address these issues by proposing a new scheme called DeepNDN, which efficiently achieves a target delivery ratio while satisfying latency and persistence constraints. The main contributions of this paper are:

- We propose DeepNDN, a communication scheme for content retrieval in vehicular networks, based on the joint application of NDN and probabilistic spatial content caching, capable of adapting to a wide range of network topologies and dynamic settings.
- We present a learning-based approach for the dynamic management of DeepNDN, which allows for achieving a target minimum delivery ratio in a resource-efficient manner by proactively adapting the content replication and availability to local conditions. It employs a Convolutional Neural Network (CNN) architecture to effectively capture the complex relations between spatiotemporal patterns of mobility and the performance of the content delivery service. A flexible cost function allows accounting for heterogeneity in the node population (e.g., in spatiotemporal patterns) and various levels of resource availability.
- We assess the performance of DeepNDN over measurement-based mobility traces in different urban settings and traffic configurations, and compare it against a set of relevant baseline approaches. Results suggest that our approach is effective and outperforms baselines regarding resource efficiency and the ability to satisfy tight QoS constraints.

The paper is structured as follows. In Section 6.2 we review the state-of-the-art. Section 6.3 presents the system model, followed by the problem formulation in Section 6.5. Our deep learning management algorithm is illustrated in Section 6.6 and assessed numerically in Section 6.7. Finally, Section 6.8 concludes the paper.

6.2 Related Work

Starting with [36], the problem of efficiently implementing information-centric content delivery in dynamic and fragmented vehicular network scenarios has received substantial interest in recent years.

A first class of results assumes that only part of the nodes is moving, and thus it focuses on enabling NDN to cope with some of the effects of node mobility and network fragmentation [11, 2, 45, 54]. For instance, [11, 2] focuses on the issues of receiver and producer mobility. It proposes an approach that combines FC and NDN and is based on keeping a trace of the movements of receivers and producers.

[45] proposes a strategy for coping with disruptions due to changes in the topology of the NDN network. [54] proposes a version of NDN based on epidemic forwarding, which proves effective in sparse scenarios. However, these approaches apply to scenarios with relatively infrequent configuration changes, and they do not scale, as they imply a significant amount of communication overhead. [4] develops an opportunistic NDN content discovery scheme, which allows nodes to localize cached copies of data in off-path vehicles using the trails left in the past from data messages. Yet it leaves open the issue of effectively delivering content within a short delay in a fragmented and dynamic scenario. [37] considers disaster scenarios with a network topology that, though fragmented, is relatively stable. It proposes a scheme that integrates NDN and DTN routing with the notion of node clusters and "data mules" for inter-cluster content retrieval. [21] proposes a protocol stack integrating the NDN and DTN architecture, addressing some interoperability issues. However, it leaves open the issue of how to orchestrate the two modes of communication to achieve a given delivery ratio in content delivery efficiently. [15] proposes an approach based on geocasting the requested content object in a delay-tolerant manner, via opportunistic replications, from the region around the producer to that in which the requester is located. However, these solutions do not address how to guarantee a target performance (in terms of delivery ratio and maximum delay) in a fragmented and dynamic network, in a resource-efficient manner. In addition, they focus on the issue of optimizing the delivery of contents which are relevant *for a single requester* only, and which do not scale to multiple requesters. In the present paper, instead, we focus on the problem of efficient delivery of content to *a population* of requesters in a QoS-aware manner.

In vehicular NDN, several approaches exploit the fixed network infrastructure to cope with intermittent connectivity between vehicles [52, 25, 48, 19]. [25] employs Road Side Unit (RSU) for sending the IM and retrieving the content object. [48] exploits a global network topology view that the infrastructure can provide to proactively inject the content on selected vehicles to improve the overall content delivery. In [53], the infrastructure is used to exchange content among different areas, while the NDN mechanism delivers the content object within each area. [6] proposes a segment and provider-aware scheme. It employs a dynamic hierarchical naming structure and utilizes a gossip protocol to assess the distribution proportions of content providers. This strategy aims to optimize network resources by mitigating unnecessary packet transmission. The system relies heavily on struc-

tured names, leading to nodes dismissing any incoming interest messages that fail to adhere to the established naming structure. However, unlike DeepNDN, none of these schemes provide QoS guarantees for the content delivery service. In addition, in these schemes content download from the infrastructure plays a key role, taking a potentially heavy toll on the utilization of infrastructure-based network resources. Instead, DeepNDN can reduce content delivery via infrastructure-based communications to the minimum amount sufficient to achieve the target QoS. Our cost-based approach allows accounting for various conditions and constraints on the availability of infrastructure support for content delivery. As we show in Section 6.7, this feature of DeepNDN translates into no content download from infrastructure in many of the considered scenarios and network conditions.

One of the most relevant approaches for the present work is the CEDO scheme [38]. It proposes a purely opportunistic NDN-based dissemination scheme, which does not rely on any assumption on the network's structure, its patterns of variation over time, or its degree of connectedness. Interests stay in the PIT until they are satisfied. A message summarising all pending interests is exchanged whenever a contact is detected. The receiver of such a message sends back all data messages that it stores in the cache. Regular Interest forwarding via FIB is disabled. Unlike DeepNDN, however, it does not minimize resource utilization nor support any QoS guarantee. Moreover, it does not address the issue of content disappearance from the region of interest due to fluctuations in the population of producers.

6.3 System Model

6.3.1 Assumptions and Notation

We consider a set of nodes moving on a road grid in the plane, modeling vehicles, pedestrians, and bike users. We assume that each node knows its position in space at any point. Every node embeds one or more wireless network interfaces (such as IEEE802.11p, IEEE802.11bd, Bluetooth, or cellular D2D [23]) to communicate directly in device-to-device (D2D) mode with other nodes in its vicinity. We say that two nodes are in *contact* when they are within each other's transmission range, thus able to exchange information. In addition, each node is endowed with a cellular network interface. Coherently with the 5G (and beyond) paradigm, we assume a coordination and management function (e.g., possibly implemented by a Software

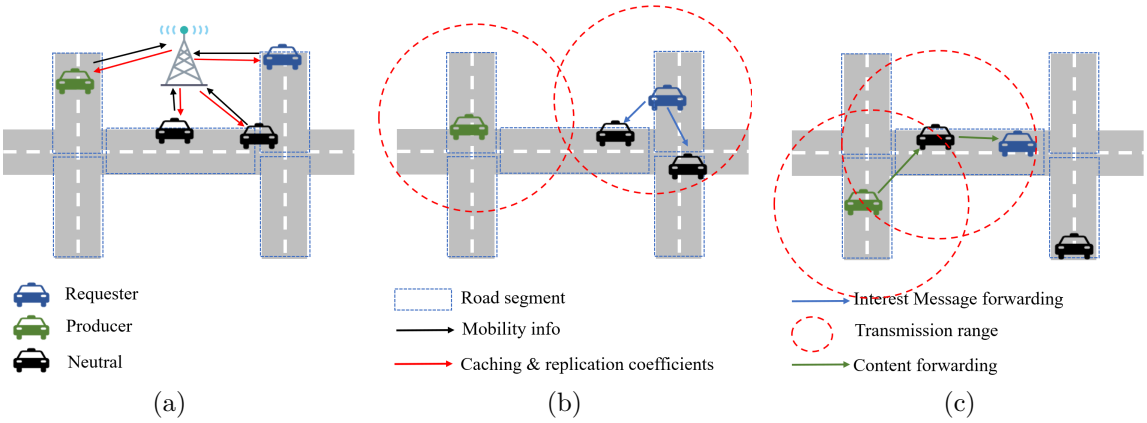


Figure 6.1: DeepNDN Overview. a) Nodes exchange control information with the controller; b) The requester forwards the Interest Message (IM) to nodes within its range; c) A producer for the requested content object comes in contact with a neutral node that received the IM, and it forwards the content object to the requester via the neutral node.

Defined Network Controller (SDNC) [39]) within the cellular access network periodically collects data about node mobility to optimize network operations.

In this work, we focus on the *optimal content delivery problem*, i.e. on the issue of efficiently delivering a content object to those nodes who request it within a given region of the plane (henceforth denoted as *region of interest*) and a specific time interval (the *observation interval*, which we denote as T). The goal is to deliver the content object in a way that minimizes its resource cost (which is typically a function of the use of cellular connectivity, as well as user storage and D2D communication resources, among others), while guaranteeing some target QoS, in terms of maximum latency of the delivery process. We assume that the management function aims at solving the optimal content delivery problem, by monitoring patterns of content requests within the region of interest, and by offloading some of the content delivery tasks from the cellular network to V2V communications. Note however that our approach is independent of the specific criteria used to decide what content object to offload.

We assume nodes are partitioned into C classes, and let c be the class label. The partition criterion can be based on mobility patterns, whether nodes are interested in the content object being delivered, and the availability and cost of host-based resources. We assume the road grid within the region of interest to be partitioned into I *segments*, each generally of different size and shape, and let i be the segment label. Similarly, we assume the observation interval to be divided into Z sub-intervals, so that node mobility patterns can be considered not to vary substantially within each

Table 6.1: Main notation used in the paper.

Name	Description
$i \in 1, .., I$	Road segment label
$c \in 1, .., C$	Node class label
$z \in 1, .., Z$	Sub-interval label
T	Duration of the observation interval
d	Maximum delay for content delivery
$\mu_{i,c,z}$	Probability to become a requester in segment i and sub-interval z for a node from the $c-$th class
$b_{i,c,z}$	Replication probability in segment i and sub-interval z for a node from the $c-$th class
$k_{i,c,z}$	Caching probability in segment i and sub-interval z for a node from the $c-$th class
t_f	Interest Message (IM) broadcast period
h	Max number of hops for IM forwarding
r	Mean delivery ratio
ϵ	Target maximum probability of content disappearance

sub-interval, and let $z \in 1, .., Z$ be their label. The criteria used for partitioning the region of interest and the observation interval can be chosen to achieve a given tradeoff point between, on one side, the computational complexity and resource efficiency of our approach, and its effectiveness on the other.

With reference to a given content object, at any point in time, each node within the region of interest is in one of the following three states:

- *neutral*, when it does not possess the content object, and it has not requested it;
- *requester*, when it has requested the content object, but it does not possess it yet;
- *producer*, when it possesses the content object.

Each node enters the considered region of interest in the neutral state. At the beginning of the observation interval, a non-empty subset of nodes in the scenario are producers (they might have generated the content themselves, or downloaded from the infrastructure), while all the others are neutral. As time progresses, each node may become a requester during its sojourn within the region of interest. In what follows, to model a process of content request arrival driven by context (e.g., by proximity to a point of interest, and/or proximity in time to some event happening in its vicinity), we assume that every time a class c node enters a segment i during sub-interval z, it becomes a requester with a probability $\mu_{i,z,c}$, generally different for each segment, node class, and sub-interval. Note, however, that our approach applies to any type of request arrival process, including nonstationary ones. Nodes outside the region of interest are always neutral, and nodes who leave the region

of interest become neutral (if they are not already so). We assume a requester becomes a producer if it receives the content within a *maximum delay* d from when it requested it. Such maximum delay can be tuned to model requirements from different applications. For instance, in proactive user-based caching schemes, d is set according to the time in the future, after which a user or application is very likely to request the content.

This work considers two different configurations for infrastructure support to content delivery. In the first one, denoted as *full D2D*, nodes cannot retrieve the content via cellular connectivity. Thus, if a node does not get its requested content via D2D by the maximum delay, it returns to a neutral state. In the second scenario instead (denoted as *cloud fallback*), after the maximum delay has passed, nodes may retrieve the content from the cloud via cellular connectivity, thus becoming producers.

6.4 DeepNDN operation

In this section, we describe the proposed DeepNDN communication paradigm. It aims to solve the optimal content delivery problem through the synergy between NDN communication mechanisms and strategies for location-based probabilistic content caching. The implementation of our DeepNDN communication paradigm is based on a simple version of *vehicular NDN* mechanisms [16], which relies on interest message (IM) flooding, and a neighbor-based forwarding approach [1]. Note, however, that our approach can be extended and adapted to accommodate a wide range of implementations and variants of NDN protocols.

We consider the case in which the content object can be exchanged as the payload of a single message. When this is not the case, the content object can be partitioned into a set of segments, each of which fits a single message, and each segment may be retrieved and delivered independently through the network.

We assume content objects are identified unambiguously by their names. Content naming may be human-readable (e.g. flat schemes such as in [27] or attribute-based [3]), but our methodology general and abstracts from any specific choice of naming scheme.

Nodes also possess unique names (denoted as identifiers or IDs). The identifier is used to trace the path from the requester to the content producer. This allows

for efficient routing and retrieval of content in our system.

When a node becomes a requester for a given content object, every t_f seconds, it broadcasts an IM for that object to all nodes within its range until it receives the content object. Any node receiving the IM checks whether it possesses the object. If this is not the case, it records the ID of the node from which it received the IM and the ID of the requested content object in the *Pending Interest Table* (PIT) and rebroadcasts it. The corresponding entry in the PIT table is removed when the node from which it received the IM is no longer within its range. When the IM reaches a producer for that content object, the producer replies by sending back the requested object in a *data message* (DM), which is routed back to the requester by exploiting the information stored in the PIT of the nodes traversed by the IM.

We assume each node is equipped with a content store (CS) for caching content objects. When a DM is received, the node checks whether the content object in the DM is present in the CS. If this is not the case, the content object is stored in the CS with a given probability. Tuning such a probability allows for modulating the number of producers in the region of interest.

To limit the overhead of this protocol and prevent broadcast storms, we assume that the IM cannot be forwarded more than h hops from the requester that originated it.

We assume each IM is valid until d seconds have passed from when the originating node became a requester. After that, independently of whether the requester gets the content object (a *delivery* event) or not (a *miss* event), the IM is not forwarded anymore, and at each node, all the PIT entries corresponding to that IM are removed.

When a node leaves the region of interest, it discards all content objects stored in the CS and all PIT entries.

A DeepNDN scheme for delivering a given content object is entirely described by the array of parameters $(\mathbf{b}, \mathbf{k})$, with $\mathbf{b} = \{b_{i,c,z}\}$ and $\mathbf{k} = \{k_{i,c,z}\}$. These arrays identify the content replication and CS caching strategies over all segments for each node class and sub-interval of the observation interval. Specifically, whenever in sub-interval z, a producer from class c located in the *i-th* road segment comes in contact with a neutral node located in the *j-th* road segment, the content object is replicated to the neutral node with probability one if the neutral node sends to the producer an IM for that content. Otherwise, it is replicated with probability $b_{i,c,z}$,

and stored in the CS of the neutral node with the probability $k_{i,c,z}$. Moreover, if a node from class c in the *i-th* segment receives a content object for which it is not a requester, it checks the PIT, and it forwards the content object to its neighbors according to the correspondent entries (as shown in Figure 6.1). If the PIT is empty, it replicates the content object with probability $b_{i,c,z}$ (independently for each node) for nodes in contact with it. Finally, it stores it in its CS with probability $k_{i,c,z}$. Note that we assume exchanges are always unicast (one-to-one). However, the proposed scheme can be easily extended to include the effects of multicasting and broadcasting. When a producer from class c enters a segment i, it keeps the content object with probability $k_{i,c,z}$, and discards it otherwise. Again, this is to model context-related popularity for a given content object.

As producer nodes ultimately leave the road grid, the population of producer nodes varies over time. In particular, at any point in time, there is a nonzero probability that the content object disappears from the grid.

The probability for such an event to take place, which we denote as the *probability of content disappearance*, is thus a key performance parameter for DeepNDN. Indeed, such an event forces the coordination function to intervene by directly delivering content to requesters via cellular connectivity, at least until enough producers are present in the region of interest, thus decreasing the ability of the DeepNDN scheme to offload the cellular network.

Therefore, parameters $(\mathbf{b}, \mathbf{k})$ also determine the probability for a given content object to persist probabilistically in the road grid for the whole duration of the observation interval.

Another key performance metric for the DeepNDN scheme is the *delivery ratio*, i.e. the mean fraction of requesters that get the content object within the maximum delay. It is thus one of the main measures of the effectiveness of the DeepNDN scheme.

As already stated, we assume that the management function is constantly monitoring mobility patterns and requests for content objects in the region of interest. Based on these data, it is also computing forecasts to decide which content objects should be delivered via DeepNDN, and thus via direct D2D communications, and which ones should be downloaded from the cellular base stations. The mechanism by which these forecasts are implemented is out of the scope of the present paper. For each of those content objects, the management function establishes a time in-

terval during which the DeepNDN-based delivery should be active (the *observation interval*), based on the estimated time during which the content object will be relevant for users in a given region of space (henceforth denoted as *region of interest*). It elaborates detailed forecasts of mobility patterns and requests. In addition, it establishes a target delivery ratio and a maximum delay for receiving requested content objects based on application requirements. The detailed algorithm by which these parameters are chosen is highly application and context-dependent, thus out of the scope of the present work.

Based on all these elements, it determines how to modulate parameters $(\mathbf{b}, \mathbf{k})$ over space and time to satisfy the given performance constraints over the observation interval. In the next section, we present our approach to derive these parameters in such a way as to achieve the target service performance efficiently.

6.5 Problem Formulation

From the description in the previous section, it emerges that the performance of the DeepNDN communication scheme critically depends on the values of the replication and caching probabilities associated with each road segment, node class, and sub-interval. In this section, we elaborate an approach for deriving those parameters $(\mathbf{b}, \mathbf{k})$ that allows achieving a target minimum delivery ratio r_0 during the whole observation interval T, while minimizing a cost function accounting for the amount of resources (bandwidth and storage) employed by the scheme, and while guaranteeing that the probability of content disappearance is below a target maximum value.

The cost function we adopt is the sum of two components. The first one accounts for the amount of storage the scheme employs on each user's device. Its expression is given by the sum, over all nodes and segments, of the amount of time that a node has been a producer for the given content object in that segment, multiplied by the content object size (in bits) L. Specifically, $\forall\ i, c$, let $W_{i,c,z}$ denote the storage unitary cost for class c nodes in segment i and sub-interval z, $\mathcal{N}_{i,c,z}$ the set of class c nodes traversing segment i during sub-interval z. $t_{n,z,i}$ is the amount of time in sub-interval z that the n-th node $\in \mathcal{N}_{i,c,z}$ spends in segment i while storing the content object. The cost component accounting for the utilization of storage resources is thus

$$S(\boldsymbol{b}, \boldsymbol{k}) = L \sum_{i,c,z} W_{i,c,z} \sum_{n \in \mathcal{N}_{i,c,z}} t_{n,z,i}. \tag{6.1}$$

The second component of the cost function accounts for the utilization of communication resources. Its expression is:

$$\Gamma(\boldsymbol{b}, \boldsymbol{k}) = L \sum_{i,c,z} \left[\left(\gamma^{O}_{i,c,z} + \frac{l}{L} \gamma^{IM}_{i,c,z} \right) X^{D2D}_{i,c,z} + \gamma^{G}_{i,c,z} X^{G}_{i,c,z} \right] \tag{6.2}$$

Where:

- $\gamma^{O}_{i,c,z}$ (resp. $\gamma^{IM}_{i,c,z}$) is the total number of device-to-device transfers of content (of IM, respectively) in road segment i which have taken place during sub-interval z for nodes from class c. We assume that the transfer cost between nodes belonging to two different segments or classes is split into equal parts among the two segments (resp. classes). Similarly, the cost of a transfer spanning more than a single sub-interval is distributed among the corresponding sub-intervals, proportionally to the time taken in each of them. Note that the cost of an IM transfer is scaled by the ratio of the size of the IM with respect to that of a content object (that is, by l/L).

- $\gamma^{G}_{i,c,z}$ is the total number of content downloads via cellular connectivity from class c nodes in road segment i during sub-interval z.

- $X^{D2D}_{i,c,z}$ (resp. $X^{G}_{i,c,z}$) is the cost per byte of a D2D transfer (respectively, of a content download via cellular link). Note that we assume, in general, such costs may vary according to node classes (e.g. due to the use of different cellular technologies for connectivity, such as 4G vs 5G, vs NB IoT), but also over space and time (e.g. due to variations in levels of traffic load in the cellular network, and in resource availability).

Note that such a cost function does not account for the messages exchanged for the diffusion of the caching and replication parameters to each node at the beginning of the observation interval. However, the resource cost of such a process is not a function of the specific caching and replication strategy. Such diffusion can be implemented in several ways, with different levels of resource utilization. Thus, optimizing such a process is out of the scope of this work.

As mentioned, given the high volatility and dynamism of the settings to which DeepNDN applies, there is always a nonzero probability that the content object "disappears" from the region of interest at some point during the observation interval. When this happens, there are no more producers in the whole area of interest.

This event may be caused by, e.g., specific spatial distribution and mobility patterns of nodes, which prevent effective content replication and/or the permanence of producers in the region. In the full D2D case, this implies that no node can become a producer anymore. In the cloud fallback case, instead, it implies that at least for a while after content disappearance, requests can be satisfied only via content retrieval from the cloud. In any case, it is undesirable, as it prevents content delivery, if not via the cellular network. If the probability of content disappearance is too high, it ends up defeating the purpose of implementing DeepNDN, as content delivery would end up taking place primarily via cellular connectivity (in the cloud fallback case) or stop taking place altogether (in the full D2D case). Let $P_z(\boldsymbol{b}, \boldsymbol{k})$ denote the probability of content disappearance during sub-interval z, and $r_z(\boldsymbol{b}, \boldsymbol{k})$ the mean delivery ratio across the whole region of interest. An optimal DeepNDN scheme for the delivery of a content object is a solution to the following optimization problem:

Problem. ***(Optimal DeepNDN schemes)***

$$\begin{aligned} &\underset{\boldsymbol{b},\boldsymbol{k}}{\textit{minimize}} && \Gamma(\boldsymbol{b}, \boldsymbol{k}) + \beta S(\boldsymbol{b}, \boldsymbol{k}) \\ &\textit{subject to,} \quad \forall z, && \\ & && r_z(\boldsymbol{b}, \boldsymbol{k}) \geq r^0 \\ & && P_z(\boldsymbol{b}, \boldsymbol{k}) \leq \epsilon \end{aligned} \tag{6.3}$$

ϵ is the maximum target probability of content disappearance within sub-interval z, while coefficient β modulates the relative weight of the two cost components. By tuning β as well as the unitary costs, it is possible to adapt the cost function to settings with different resource availability on user devices and in the cellular network and to different incentive schemes for resource sharing and cooperation.

Note that, in the case in which different content objects have to be delivered via DeepNDN in the same region of interest, the optimal strategy for each content type is derived by solving Problem 6.3 separately for each of them, as each content type may come with its performance constraints (i.e., target delivery ratio, maximum delay, and observation interval). However, the approach presented can be easily extended to a formulation that optimizes the overall cost for several different contents.

Problem 6.3 involves performance parameters (such as the delivery ratio and the probability of content disappearance) whose dependence on system parameters and input constraints is complex to model analytically in a heterogeneous setting without

relying on strong assumptions that would limit the accuracy of the results. In the next section, we present a learning-based approach to this optimization problem to address this issue.

6.6 A Deep Learning algorithm for resource-efficient DeepNDN management

In this section, we describe our approach to solving Problem 6.3, based on a specific machine learning architecture, suited for modeling data with grid-like topology. The choice of a deep learning approach is due to the high complexity of the relationships between the system parameters, the caching and replication strategies, and the performance of our DeepNDN scheme, particularly in non-homogeneous settings. Indeed, data-based approaches are fostered by the increasingly large amount of data (on network operations, as well as on user patterns of mobility and of service demand) collected by the cellular infrastructure in the 5G paradigm and beyond [26]. The specific choice of a CNN architecture is due to its ability to capture the complex relations between elements of a multidimensional set of system parameters [14]. In particular, when applied to realistic, measurement-based scenarios, CNN architectures have been shown to enable higher accuracy and efficiency in capturing spatial correlations (in our case, between spatial features of the region of interest) than other learning approaches, such as Decision Tree or Random Forest [30]. Other architectures, such as RPN or XNN, may also be a good fit, but optimising this choice is always possible without changing the substance of our approach, and it is thus out of the scope of our work.

As stated, we assume that, starting from collected data on node mobility and content request patterns, the coordination and management function elaborates forecasts on their spatiotemporal evolution, as well as on communications and storage resources availability. In this way, it identifies opportunities for resource optimization via offloading part of content delivery tasks. The setup and operation of our DeepNDN communication scheme are divided into three phases:

- **Data collection and elaboration**. The management function regularly collects and records node mobility traces across the whole region of interest. A training set is generated by associating the communication features with the measured mobility features and node contact patterns. At system bootstrap,

this is performed through simulations, while at runtime, communication features are collected from DeepNDN operations. Finally, the CNN is trained on this data.

- **Computation of a DeepNDN strategy**. When a decision to offload the delivery of a content object to the DeepNDN communication scheme is taken, the management function in the cellular infrastructure (e.g., an SDN controller) determines the target delivery ratio, maximum delay, and the duration of the observation interval based on application-level QoS requirements. Based on resource availability and load levels of the cellular infrastructure, it also determines the acceptable value of ϵ (the maximum target value of the probability of content disappearance) and whether a miss event should imply a content retrieval from the cloud, or not. Note that the method used by the management function to perform these tasks is out of the scope of this work. After this, it uses the trained CNN to compute, in real-time, a set of parameters $(\mathbf{b}, \mathbf{k})$ that allows satisfying the constraints in Problem 6.3 on target minimum delivery ratio r_0 and content disappearance probability in every sub-interval, while minimizing the cost function.

- **Deployment**. The system provides to all nodes in the region of interest, and to all those which enter it during the observation interval, the parameters $(\mathbf{b}, \mathbf{k})$ which orchestrate the DeepNDN scheme for the given content. Again, how to optimally deliver parameters is out of the scope of this work.

In cases where mobility and/or request patterns deviate significantly from the forecasted ones during operations, new forecasts are elaborated by the management function, and a new set of coefficients is computed and adopted by all nodes in the region of interest. In this way, our approach can cope with inaccuracy in those forecasts of patterns of mobility and requests on which the derivation of the DeepNDN caching and replication strategy is based.

In what follows, we describe in detail the three phases and the algorithms involved.

6.6.1 Data collection and elaboration

As already stated, we assume that, at regular time intervals, the management function computes a set of aggregate metrics relative to node mobility and wireless

communications in each segment. Let y be the label of these time intervals. With $\hat{\mathbf{m}} = \{\hat{m}_{i,c,y}\}$, we denote the *mobility array*, consisting of the values of these aggregate parameters for the whole region of interest, for each time interval. Specifically, for each segment, node class, and time interval, the mobility array contains the average node speed, the average number of nodes, and the average number of nodes in contact (the list of nodes able to exchange beacons according to the given communication protocol [47])(see Table 6.2).

Such a choice of features (which constitute the input parameters for our CNN model) represents only one of many possible ones. However, these have been chosen as they are typically used as input to the main existing models (both analytic and data-based) of probabilistic content caching based on opportunistic replications [34, 31, 33]. Moreover, in our evaluations, such a choice has been shown to enable a high degree of accuracy in accounting for the peculiarities of node mobility patterns and their effects on the performance of the content delivery process.

While collecting aggregate metrics, the system applies a *label generation* procedure, i.e. a randomization procedure, over the collected data. Specifically:

- For each time interval y for which the system has collected aggregate metrics, a set of random caching and replication schemes, of the form $(\{\hat{b}_{i,c,y}\}, \{\hat{k}_{i,c,y}\})$ is generated, together with a random process of request arrivals, i.e. a set of coefficients $\{\hat{\mu}_{i,c,y}\}$. We denote them as $\hat{\mathbf{b}}$, $\hat{\mathbf{k}}$ and $\hat{\boldsymbol{\mu}}$, respectively.
- For each set of parameters $(\hat{\mathbf{m}}, \hat{\mathbf{b}}, \hat{\mathbf{k}}, \hat{\boldsymbol{\mu}})$, a set of simulations are performed based on the given random caching and replication strategy, and on the user trajectories during the y-th time interval, and on the given random request arrival process. The parameters measured for each segment and time interval are the mean number of requesters per node class, the mean delivery ratio, the mean number of producers per node class, the mean time required to satisfy a content request, and the mean number of nodes transmitting (in D2D mode and via infrastructure based cellular communications, respectively) at a given time instant, per node class. Moreover, the fraction of simulation runs in which the content disappeared for the whole region of interest is measured for each time interval. These parameters, measured via simulations, constitute the *communications feature vector* $\mathbf{u}$, and they are the basis for the estimation of the resource utilization of the random strategy, and its associated costs.

The array $(\hat{\mathbf{m}}, \mathbf{u})$ denotes the *Segment Features Vector* associated with $(\hat{\mathbf{m}}, \hat{\mathbf{b}}, \hat{\mathbf{k}}, \hat{\boldsymbol{\mu}})$

Table 6.2: Features of mobility, communications, and the request arrival process, collected for each time interval y, to build the training and validation sets.

Array	**Feature**
Mobility ($\hat{\mathbf{m}}$)	Average number of nodes in class c and segment i
	Average node speed for class c in segment i ([m/s])
	Average number of nodes in contact in segment i
Communications ($\mathbf{u}$)	Average number of requesters in segment i for class c
	Average number of producers in segment i for class c
	Average number of transmitting nodes in segment i for class c (D2D)
	Average number of transmitting nodes in segment i for class c (cellular)
	Mean delivery ratio in segment i ($[s^{-1}]$)
	Mean request delay in segment i ([s])
	Fraction of simulation runs with content disappearance

for each time interval (Table 6.2). Note that each vector $(\hat{\mathbf{m}}, \mathbf{u})$ is normalized to avoid numerical issues in the subsequent phases of the process. This process is repeated several times for each array $\hat{\mathbf{m}}$, as a function of the desired size of the training and validation sets. To create datasets composed by unbiased quintuplets $(\hat{\mathbf{m}}, \mathbf{u}, \hat{\mathbf{b}}, \hat{\mathbf{k}}, \hat{\boldsymbol{\mu}})$, which do not differ in any systematic way, we apply a covariate adaptive randomization [42], which uses the method of minimization by assessing the imbalance of sample size among several covariates. This technique ensures no a priori knowledge of group assignment and prioritizes the configuration sampling over the feature. Indeed, for a given feature set $\hat{\mathbf{m}}$, several configurations $(\hat{\mathbf{b}}, \hat{\mathbf{k}}, \hat{\boldsymbol{\mu}})$ are tested with respect to the covariance of the measured delivery ratio and meantime for satisfying a request.

Once such a dataset is collected, it is split into a training set and a test set using the stratified technique [43, 46]. The output of this phase is thus a data set composed of quintuplets of the form $(\hat{\mathbf{m}}, \mathbf{u}, \hat{\mathbf{b}}, \hat{\mathbf{k}}, \hat{\boldsymbol{\mu}})$. Note that this phase can be performed entirely offline.

At runtime, while the DeepNDN scheme is applied to the region of interest, these datasets are enriched in an ongoing manner with new elements. Specifically, as the management function keeps on measuring new patterns of requests and mobility, together with the corresponding performance metrics, it includes in the data set these elements derived by directly measuring performance (precisely, those in the communications feature vector) during DeepNDN operations. In addition, also in this phase, new elements of the data sets can be added through the simulation-based procedure just described. Indeed, the contribution to the given data set of

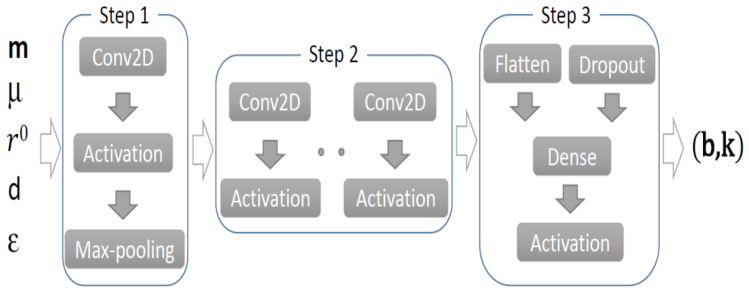


Figure 6.2: Architecture of the Convolutional Neural Network used to compute resource-efficient replication and caching strategies $(\mathbf{b}, \mathbf{k})$ for the DeepNDN communication scheme. Its inputs are the forecasted mobility features $\mathbf{m}$, the forecasted coefficients of the request arrival process $\boldsymbol{\mu}$, the target delivery ratio r^0, the maximum delay d, and the maximum acceptable probability of content disappearance ϵ within each sub-interval.

those elements derived through simulations plays a key role in enabling the system to effectively configure a DeepNDN scheme, e.g. for relatively infrequent scenarios, such as road accidents or disasters, for which few measured data are typically available.

6.6.2 Computation of a DeepNDN strategy and deployment

Once the observation interval and its partition in sub-intervals have been defined, let $\mathbf{m}$ be the mobility feature array, and $\boldsymbol{\mu}$ the coefficients of the request arrival process for the given observation interval. Given this input, the management function derives through our CNN the replication and caching strategies $(\mathbf{b}, \mathbf{k})$ that achieve the target delivery ratio while minimizing the resource cost for each sub-interval in the whole observation interval. These parameters are then delivered to all nodes in the scenario. In those cases in which, during our scheme operation, new forecasts of mobility become available, the system computes a new strategy $(\mathbf{b}, \mathbf{k})$ for the remaining portion of the observation interval based on such forecasts. It injects the new values of replication and caching parameters into all the nodes. Similarly, when mobility patterns and content requests vary significantly during the observation interval, the management function may split such interval into sub-intervals (defined in such a way to ensure sufficient uniformity of the mobility patterns within each of them) and apply the dimensioning procedure we have described to each of them.

6.6.3 Convolutional Neural Network Architecture

In this section, we describe the overall structure of the CNN based on our DeepNDN dimensioning approach. Figure 6.2 outlines the structure and operation of

our CNN. The number of layers in each of the three steps determines the *depth* of the learning architecture and, hence, the level of complexity of the correlations that it can capture. Such correlations between different road segments result from wireless propagation effects, the spatiotemporal patterns of mobility, the content request, and their mutual interactions. Indeed, the ability of CNN to capture both intra-segment and inter-segment relations between features, and in particular, to model complex correlations even among segments that are distant in space and over different time intervals, makes such a learning approach a perfect fit for the problem of optimally orchestrating a DeepNDN scheme in realistic settings.

In step 1, our CNN learns the local feature correlations, such as the spatiotemporal correlations between speed and node density or the ratio between requester and producer. More specifically, the convolution layer (Conv2D) captures the correlation between features of the same road segment. Instead, the activation layer selects the most relevant local features. Finally, the max-pooling layer enhances computational performance by reducing the data dimensionality [14]. Its output is the most relevant local features per road segment.

The goal of step 2 is to enable the CNN to learn the structure of complex inter-segment dependencies, particularly those involving segments that are distant in space among them and those involving different time intervals, thus improving model accuracy. Therefore, step two comprises multiple instances of the convolutional layer, each followed by an activation layer, which selects the most relevant inter-segment features. The number of Conv2D layers, as well as the kernel size used in these layers, may be chosen in such a way as to achieve a target balance between the computational complexity and accuracy of the CNN.

Step three processes the previous results to reduce data dimensionality and produce the coefficients defining our strategies for DeepNDN management. It is composed of a Flatten layer for data dimensionality reduction, a Dropout layer that contributes to avoiding overfitting issues, and a Dense layer that reshapes data. The last layer is an Activation layer, which selects the best strategy among a set of possible candidates, providing as output the coefficients $(\mathbf{b}, \mathbf{k})$. The machine learning parameters (i.e., number of neurons and layers) of the above three steps have been set according to the CNN learning rate derivative [14].

A crucial aspect of the performance of our approach is the computational load required by the three phases that compose it. The amount of computation required

by the training process is directly related to the size of the training set and the number of segments and features considered. To reduce such a computational load, we have adopted a discretization of the caching and replication parameters, casting Problem 1 into a classification problem [14]. However, all the operations in this phase can be executed offline (i.e., before the need for content offloading and pre-fetching arises). Hence, their computational complexity has no impact on the performance of the scheme (e.g. in terms of delay between the beginning of the content distribution process and the application of the CNN-based caching and replication strategy).

The computations required to derive the DeepNDN strategy for specific content are performed when the infrastructure makes an offloading decision. As this decision is not made in real-time but based on forecasts (of say, a few hours at least), it does not impact the time required to implement the caching and replication strategies.

6.7 Numerical Evaluation

In this section, we assess numerically the performance of our DeepNDN scheme in both synthetic and measurement-based mobility scenarios. Our main goal is the characterization, in a variety of scenarios and conditions, of the effectiveness and efficiency of the spatio-temporal strategies emerging from our deep-learning-based management approach when compared to the main relevant baseline algorithms.

6.7.1 System setup

We assume nodes implement the IEEE 802.11p wireless access protocol [23], with a maximum transmission power of 20 milli watt, a minimum signal attenuation threshold of -89 dB, and a minimum path loss coefficient of 2, which give a transmission radius of 100 m. These values correspond to typical settings in a vehicular environment, e.g. in the WAVE protocol [23]. We simulate content exchanges among nodes using Veins on the OMNeT++ simulator [44, 51], whereas SUMO [28] has been adopted for vehicular mobility simulation over a road grid. We use Keras [8] to implement our DeepNDN strategy.

For each segment, mobility features have been measured using a sampling interval of 1 second to capture the dynamics of mobility and content diffusion accurately. In all scenarios, we have used a training set size of $2 \cdot 10^5$ data points, which has proven sufficient to achieve a high level of accuracy, and we have performed a $10-$fold

cross-validation. For testing, we assumed a test set size of $5 \cdot 10^3$ data points, proven sufficient to achieve a 95% confidence interval of at most 3% in all of the considered settings. Unless stated otherwise, the coefficient β in the cost function in Problem 1 has been set to 1 to give equal weight to storage and communication costs. The default values of target delivery ratio (0.9), probability of content disappearance (0.01), and observation interval duration (1800 second) have been chosen to model settings in which the DeepNDN scheme must be effective over short time intervals, and in which the vast majority of content deliveries must be implemented via D2D communications, e.g. due to peaks of traffic load on the cellular infrastructure. Given such a relatively short time duration, we have considered the observation interval to be composed of a single sub-interval. By default, the maximum acceptable content retrieval delay has been set to 5 second to model applications that rely on very short-term predictions of content requests. Unless otherwise specified, given the high dynamicity of the scenarios considered, we have assumed that IMs are broadcasted every 1 s, for the NDN mechanisms to be able to promptly adapt to changes in network topology. The strategies for content delivery we have considered in our assessment are:

- *(Unconstrained) DeepNDN*, i.e. the communication strategy outlined in Section 6.6, for which we assume there are no limits to the maximum number of hops to which an IM can be forwarded.

- *(Single hop) DeepNDN:* This is derived from Section 6.6 too. However, in this strategy, IMs can only be forwarded from the requester to nodes in direct contact with it. This strategy minimizes the communication overhead due to the forwarding of IM, which in scenarios with large node clusters, may jeopardize the communication channels, potentially affecting the performance of the content delivery process.

- *Single hop/unconstrained NDN.* This scheme corresponds to the NDN implementation described in Section 6.4, but with no opportunistic content replication. More precisely, it is derived by the DeepNDN scheme of Section 6.6 (single hop or unconstrained) by setting to zero the opportunistic replication parameters $\mathbf{b}$, and to one the caching probability parameters $\mathbf{k}$. In this scheme, the content is routed from the producer to the requester only if a stable (multi-hop) path is found between the former and the latter, and it is cached only by nodes on the path (in addition to the requester). This scheme

is thus representative of the vast majority of existing NDN approaches.

- *All-on DeepNDN*, derived from unconstrained DeepNDN, by setting all caching and replication coefficients to one. It shows the performance achievable (and the corresponding resource utilization) when DeepNDN employs all available user-based bandwidth and storage resources instead of tuning its coefficients for resource efficiency.

- *CEDO [38].* This is the main existing approach that addresses the problem of optimal delivery of named data in fragmented network settings (Section 6.2). Differently from classical NDN schemes, it does not rely on IM forwarding, but only on opportunistic content exchanges.

- *SPG-ICN [6]* is a segment and provider-aware strategy designed for vehicular networks. It aims to minimize communication overhead by avoiding unnecessary transmissions. The scheme employs a dynamic hierarchical naming structure, incorporating vehicular context information, specifically the segment and vehicle direction. Each vehicle discards IMs received from vehicles moving in the opposite direction and filters interest/data packets from segments different from its own. By limiting IM forwarding and unnecessary content transmission, this approach reduces overhead. Additionally, SPG-ICN uses a gossip protocol to estimate the spatial density of content providers. Producers replicate the content if the density of content providers falls below a target minimum value.

6.7.2 Baseline scenario

In the first set of simulations, we aimed to characterize the main performance patterns of DeepNDN in a controlled scenario with stationary synthetic mobility patterns. This choice guarantees some level of homogeneity in space and over time of mobility patterns, enabling fine control over node densification and clustering patterns while eliminating spurious effects due to non-stationarity. We have considered the full D2D configuration, in which a miss event (or an event of content disappearance from the region of interest) does not imply content retrieval from the cloud. This represents the most challenging configuration for our content distribution problem, as QoS constraints for content delivery must be satisfied only via D2D exchanges.

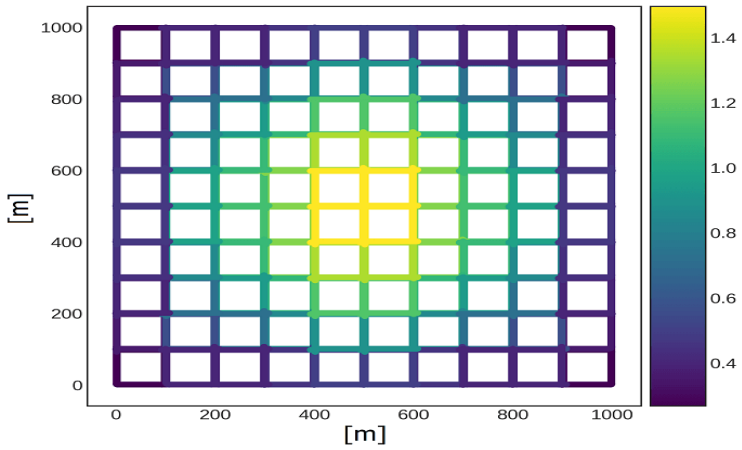


Figure 6.3: Mean number of neighbours per node in Manhattan scenario.

We have considered a region of interest given by a Manhattan grid of 10 by 10 square blocks, each of side 100 meter. The partition into road segments has been such that every portion of the road between two consecutive crossroads consists of two road segments (one per each roadway), plus a segment covering the centre of each crossroad, for a total of 460 road segments in the whole region of interest.

A single class of nodes is present in this scenario, modelling vehicles that move according to the Manhattan mobility model [17]. Specifically, they adopt a turn probability of 0.25, equal for both left and right turns, and a speed uniformly distributed between 30 and 50 kmph, as common in several urban settings in fluid traffic conditions. Nodes enter the region of interest from segments at the border at a rate of 1/0.0024 second per border segment. This brings an average of 100 vehicles in the region at any time and a mean sojourn time of 92.2 seconds. Throughout all the simulations, these settings never gave rise to a network topology consisting of a single connected component. Instead, small, short-lived clusters periodically formed at each crossroad, particularly at the centre of the grid, due to vehicles queuing Fig. 6.3. At the beginning of the observation interval, we assumed that no node is a requester and that each node has a 0.1 probability of possessing the content. This value has been chosen as, in our experiments, it has proven to be high enough to minimize the likelihood of content disappearance during the initial transient of content diffusion while being low enough to bring to a short duration of the above-mentioned transient. As for what concerns the request arrival process, we assumed every node entering the grid to be a requester with a probability of 0.9. Such a choice models the case in which the need for a given content object is triggered by the ingress to the region of interest (e.g. a map of available parking lots in a given

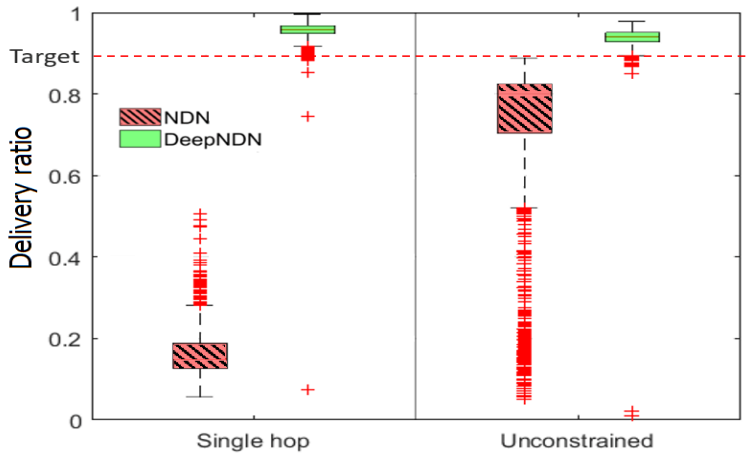


Figure 6.4: Box plot of delivery ratio for our DeepNDN and NDN strategies, in both the single hop and unconstrained configurations. For the NDN schemes, there is no maximum delay for content retrieval, and failed deliveries are limited to the case in which requesters get out of the region of interest without receiving the content object. Baseline scenario, with full D2D configuration.

Table 6.3: Resource utilization of DeepNDN in the baseline (Manhattan) scenario, for a 98% confidence interval of at most 3%.

Forwarding strategy	**Mean content availability**	**Mean concurrent exchanges**
Single hop	0.49	8.2
Unconstrained	0.32	8.3

region of the city, requested by cars looking for a parking slot), and in which the content object must be delivered as soon as the requesting nodes enter the region of interest.

As we have verified, these choices gave rise to stationary patterns of mobility, vehicle densification, and queuing in the given region. For NDN schemes, we have considered the performance in the optimistic case in which there is no upper bound on the maximum acceptable content delivery delay. We assumed that in NDN schemes, requesters are ready to wait indefinitely before receiving the requested content object. This allows a worst-case estimation of the improvements, in terms of delivery ratio, which our DeepNDN schemes bring with respect to NDN. As the box-plots in Fig. 6.4 and Fig. 6.5 show, differently than in the NDN scheme, in both the unconstrained and single hop configurations, our DeepNDN management strategies are able to satisfy the target delivery ratio within the given constraint on maximum delay. Indeed, even when assuming no limitations to the delay with

which a request can be satisfied in the NDN schemes, and even when caching all contents forwarded, content delivery based only on NDN is not able to achieve the target delivery ratio. In particular, these results suggest that our approach is able to effectively tune content replication and caching, bringing the likelihood of not satisfying the performance constraints to shallow values (in over $5 \cdot 10^3$ simulations, less than 3% violated this constraint), while using only a small portion of system resources.

Table 6.3 shows the mean content availability (i.e. the mean fraction of nodes with content) and the mean number of content exchanges taking place at any time instant for the two considered configurations of the DeepNDN scheme. As Fig. 6.4 shows, restricting IM forwarding to a single hop does not affect the ability of our approach to satisfy the target performance. However, it comes at the cost of a 53% increase in the amount of storage resources used.

Indeed, the fact that our paradigm is effective in our scenario even when limiting the IM forwarding to a single hop might lead to assume that NDN content retrieval alone could be sufficient to achieve the target performance and that the contribution to the performance of the DeepNDN scheme given by opportunistic content replication plays only a marginal role. Instead, Fig. 6.5 clearly indicates that even in the most favourable conditions, all NDN schemes systematically fail to deliver the content within the maximum delay of 5 seconds.

In addition, while no content disappearance has been observed with the DeepNDN schemes, the mean percentage of contents that disappeared from the scenario before the end of the observation interval in the NDN schemes has been of 11% for the unconstrained case, and of 87% in the single hop case.

6.7.3 Highway Scenario

Highways present unique challenges for vehicular networks due to vehicles' high speed and relatively straight movement patterns, leading often to network fragmentation. To evaluate the performance of our algorithms in these settings, we considered a region of interest given by a two-way highway that is 3 km long, with two lanes in each direction, each lane being approximately four meters wide, and a mean speed of 90 km/h, as in [6]. The region is divided into segments that are 200 meters long and a width of 16 m, equal to that of the highway. Each lane accommodates on average 1800 vehicles per hour, and the simulation duration is 1200 s. We maintain

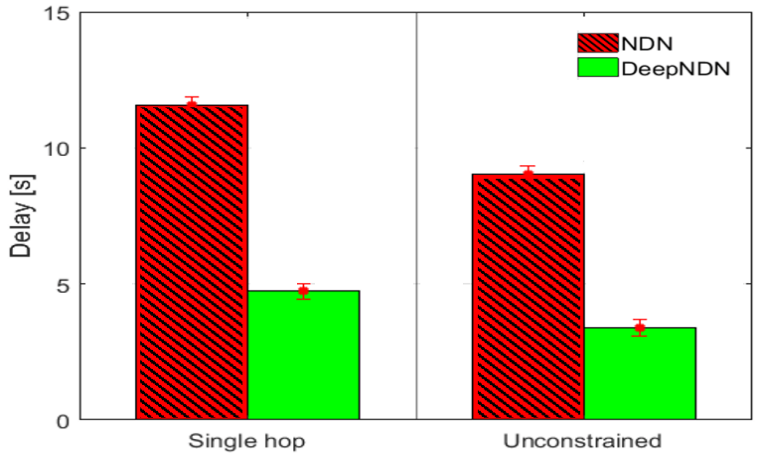


Figure 6.5: Content delivery delay for the DeepNDN and NDN strategies in both the single hop and unconstrained IM forwarding configurations. No upper bound on delivery delay is set for NDN schemes. Baseline scenario, with full D2D configuration.

Table 6.4: Comparative analysis of DeepNDN and SPG-ICN in the highway scenario.

Cost function components	**SPG-ICN**	**DeepNDN**
D2D content transfers	431	778
RSU content transfers	42	8
D2D IM transfers	3458	1796
Content storage [s]	14664	9253

the number of in-network content providers in SPG-ICN as 30 per cent of the total vehicle density to restrict the use of RSU for forwarding the content and prevent unsolicited interest/data flooding [6]. Table 6.4 shows that in this scenario the DeepNDN approach is more efficient than SPG-ICN regarding the content transmitted by RSUs (almost five times less) and transmitted IMs (almost half), and it relies more on D2D content transfer. In SPG-ICN, the system relies heavily on structured names, leading to nodes dismissing any incoming interest messages or content that fails to adhere to the established naming structure. This could potentially lead to a loss of information in sparse and medium-sized networks, an increased reliance on RSUs for content delivery, and a higher number of retransmissions.

6.7.4 Luxembourg Scenario

In order to perform a more realistic assessment of the performance of our DeepNDN approach, we considered a region of interest corresponding to a square area of side

1 km in the centre of Luxembourg City (Fig. 6.6). A single class of nodes has been considered here, too, modelling vehicles. The road grid and its partition in road segments have been derived from OpenStreetMap [40]. Such partition is performed similarly to the one in the Manhattan scenario, with at least two segments for each portion of the road between two crossroads, for a total of 234 segments in the region of interest. The measurement-based vehicular mobility traces have been derived from the Luxembourg SUMO Traffic Scenario (LuST) [9], and they refer to a rush hour time interval (7:00 AM to 7:30 AM), with an average of 82.7 nodes present in the area in the considered interval. Here, too, we have considered the full D2D configuration. We denote such setup as the *Luxembourg rush hour* scenario.

Fig. 6.6 shows the mean number of neighbours per node (i.e. the number of nodes in contact with a given node) in a given instant in the region of interest. The map indicates that even in rush hours, in the considered scenario, a high density of nodes (and thus potentially a high likelihood of clustering) is present only in very limited portions of the considered area. Therefore DTN is likely to play a key role in enabling timely delivery of content objects in the whole area.

To consider a more realistic configuration for the content request arrival process, we have assumed a point-of-interest (POI) based pattern of request arrivals. Namely, we have assumed the presence in the map of a POI (e.g. a cinema, denoted with a light yellow spot in Fig. 6.7), which is likely to be correlated with the need for a specific content object (e.g. a movie trailer) on behalf of users. Thus, we have assumed that the probability for a node entering a given segment to become a requester is equal to one in the segment containing the POI and that for all other segments, it decreases as the inverse of the distance of the segment centre from the point of interest. The resulting average spatial density of requesters is shown in Fig. 6.7. In such a scenario, we have evaluated the performance of our unconstrained DeepNDN strategy in the full D2D configuration, comparing it with that of the unconstrained NDN strategy, in which no opportunistic content replication takes place, as well as with those the CEDO algorithm. Given the higher spatial inhomogeneity of node density and nonstationarity of node trajectories, to perform a conservative estimation of the relative performance of DeepNDN with respect to competing approaches (and NDN in particular) in this scenario, we have considered a maximum request delay of 10 second, i.e. a looser performance requirement concerning the synthetic Manhattan mobility scenario.

As shown in Fig. 6.8, in this scenario, our DeepNDN strategy achieves a delivery

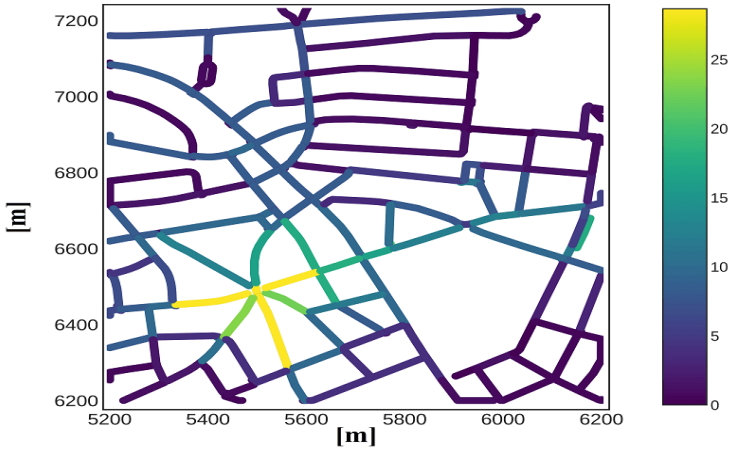


Figure 6.6: Mean number of neighbours per node. Luxembourg rush hour scenario.

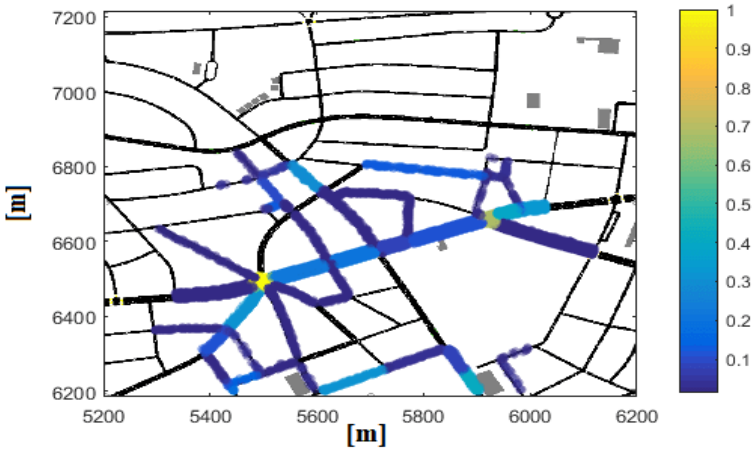


Figure 6.7: Mean number of requesters per road segment in the region of interest, in the Luxembourg rush hour scenario. The point of interest is the light yellow spot.

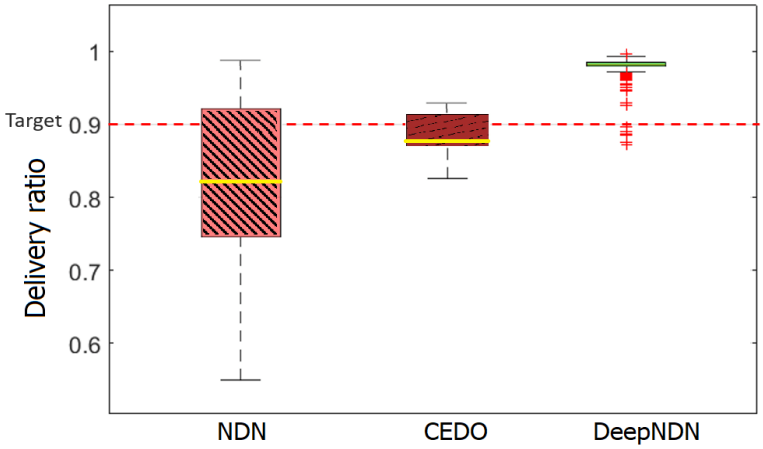


Figure 6.8: The delivery ratio for the DeepNDN, NDN, and CEDO scheme in the unconstrained IM forwarding case for the Luxembourg rush hour scenario. Full D2D configuration.

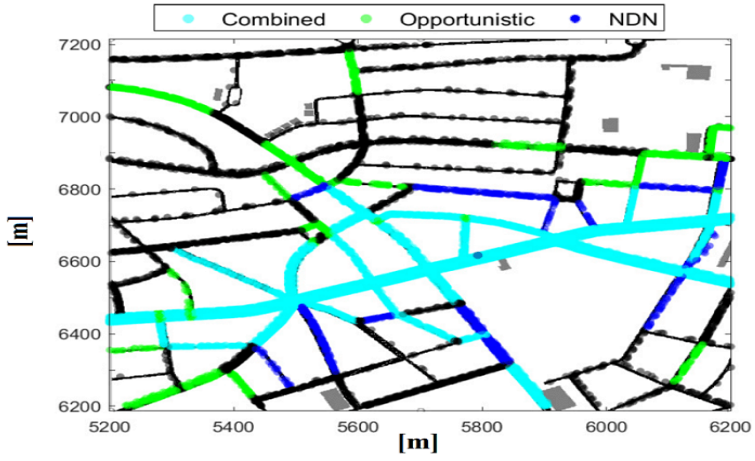


Figure 6.9: Spatial configuration of the DeepNDN caching and replication strategy for the Luxembourg rush hour scenario, with full D2D configuration. Segments with exclusive opportunistic replication are green, and those with exclusive NDN content delivery are blue. Segments in which both mechanisms are active simultaneously are in cyan.

ratio close to 0.99 in all the simulations. All the competing approaches do not meet the target delivery ratio of 0.9, with the CEDO approach achieving it in only 30% of the simulations. In the NDN scheme instead, and despite the highly favorable conditions for NDN performance (in terms of correlation in space between zones of a high user density and zones with a high density of requesters), only 18.3% of requesters receive the content within the maximum delay (Fig. 6.8). Such conservative comparisons suggest that in realistic vehicular scenarios, and even in conditions in which vehicle density and node clustering peak around the point of interest, only the combination of NDN and strategies for opportunistic content replication and caching allows for achieving high delivery ratios. Such a combination is, therefore, key for successfully supporting content delivery with stringent delay requirements when content retrieval can only take place via D2D exchanges. We note that no content disappearance was observed in the scenario for all considered algorithms.

To gain further insights into DeepNDN performance in this scenario, in Fig. 6.9, we have put in evidence those road segments where DeepNDN content exchanges take place only through opportunistic replication (in green), only through NDN (in blue), and with both mechanisms (in cyan). As expected, in those segments with a high density of nodes, particularly around the point of interest, content is delivered through a combination of NDN and opportunistic replications. In these segments, opportunistic replications aim to achieve a sufficiently high amount of content redundancy for the NDN-based content retrieval to be effective within the given maximum delay.

Around these road segments, a few (marked in blue in the figure) are characterized by a lower density of requests. These segments exploit the proximity of high-density segments (and their high content redundancy in particular) to retrieve content objects via NDN only. Indeed, thanks to such proximity, to achieve the target performance they do not need to increase content availability beyond those levels achievable via NDN only. Further away from the point of interest, in areas with even lower densities of users and content requests, the content object is replicated exclusively in an opportunistic manner. These replications are aimed mainly at transferring content to at least some of the vehicles that approach the point of interest, thus increasing content availability and delivery ratio in those segments closer to the POI. Another key performance aspect to consider in our DeepNDN schemes is resource efficiency. To evaluate this, in Table 6.5 we have compared all the considered algorithms in terms of the mean, across all simulations and all the observation

Table 6.5: Resource utilization in the Luxembourg rush hour scenario. Results are with a 98%confidence interval of at most 3%.

Approach	**Mean Availability** (%)	**Transmitting Nodes** (%)
NDN	52.5	43.9
All-on DeepNDN	62.4	44.7
DeepNDN	30.8	46.7
CEDO	18.4	58.3

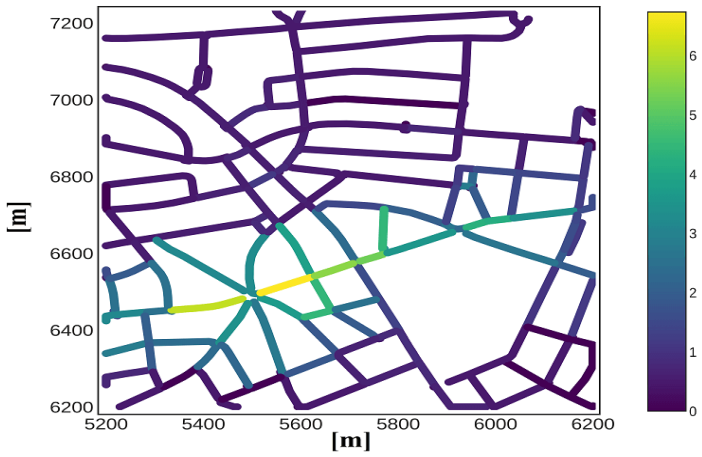


Figure 6.10: Mean number of neighbours per node. Luxembourg off-peak scenario.

intervals, of the percentage of nodes storing in memory the content object (denoted as mean content availability), and of the percentage of nodes transmitting. Despite both the DeepNDN and the all-on DeepNDN approaches satisfying the performance targets in the considered scenario, DeepNDN requires a substantially lower level of content redundancy with respect to the all-on and NDN-only schemes. Only CEDO (which is not able to achieve the target performance) requires lower content availability values. Indeed, in CEDO each producer transfers the content object to all the nodes in contact. If the receiving node is a requester, it stores it; otherwise, the content object is discarded. Such a strategy translates into a relatively high number of transmissions, low content availability, and an insufficient delivery ratio. As for bandwidth utilization, all schemes considered are substantially similar, with only CEDO being less efficient. Remarkably, the utilization of communication resources in the all-on scheme is substantially similar to the DeepNDN case. Indeed, in the all-on case, a higher availability brings to a lower rate of content transfers, seeing that a higher fraction of contacts takes place among nodes that already possess the content object.

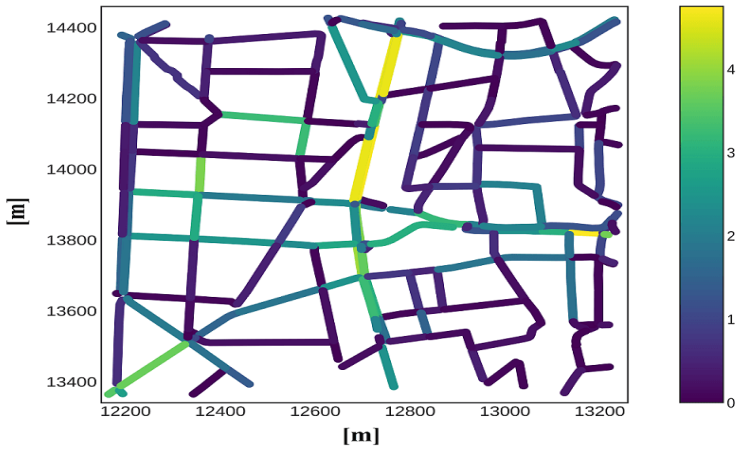
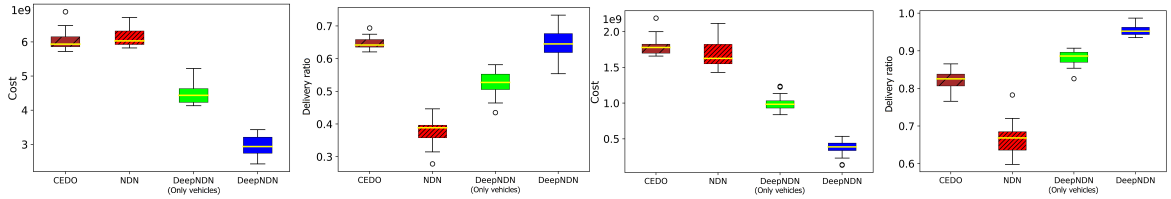


Figure 6.11: Mean number of neighbours per node. Cologne off-peak scenario.

6.7.5 Two node classes scenarios

In another set of experiments, we assessed our DeepNDN scheme in setups closer to present-day and near-future urban settings served by 5G networks (and beyond), in two key aspects. On one side, these settings are characterized by a pervasive deployment of macro and micro base stations. This allows the infrastructure to reliably intervene whenever D2D-based content diffusion fails by delivering the content from the cloud to the requester whenever the maximum retrieval delay has passed (cloud fallback configuration). Thus, unlike the previous scenarios, requesters retrieve the content from the cloud if they do not receive it via D2D communications within a maximum delay. As a result, in these settings and for all content delivery strategies, all requester nodes ultimately become producers. Moreover, differently than in the full D2D case (in which content disappearance prevents requests arriving after the disappearance from being satisfied), in the cloud fallback configuration content disappearance is only temporary, and it only decreases (usually for short time intervals) the possibility to satisfy requests via D2D exchanges.

In addition, we assume these scenarios to be characterized by the presence of different types of nodes with different mobility patterns, resource costs, and roles. Specifically, we consider settings with two classes of nodes, modeling scenarios in which, in addition to vehicles, pedestrians support the content diffusion and delivery process with their smartphones. We assume that content is of interest only to vehicular users so that pedestrians never become requesters. We considered two different settings. In the first (denoted as *Luxembourg off peak*), the region of interest is the same as in the Luxembourg rush hour scenario. However, the considered time



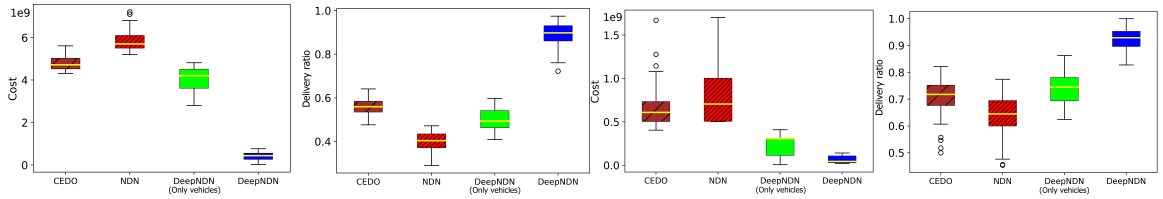
(a) Cost, 5 s maximum content retrieval delay.

(b) D2D delivery ratio, 5 s maximum content retrieval delay.

(c) Cost, 60 s maximum content retrieval delay.

(d) D2D delivery ratio, 60 s maximum content retrieval delay.

Figure 6.12: Cost and D2D delivery ratio of the content distribution process, for the considered algorithms, in the Luxembourg off-peak scenario with pedestrians. Cloud fallback configuration.



(a) Cost, 5 s maximum content retrieval delay.

(b) D2D delivery ratio, 5 s maximum content retrieval delay.

(c) Cost, 60 s maximum content retrieval delay.

(d) D2D delivery ratio, 60 s maximum content retrieval delay.

Figure 6.13: Cost and D2D delivery ratio of the content distribution process, for the considered algorithms, in the Cologne off-peak scenario with pedestrians. Cloud fallback configuration.

interval is 12:00-12:30 PM, with an average of 27.6 nodes present in the region of interest and an average sojourn time of 170.1 s. In the second scenario (denoted as *Cologne off peak*), the region of interest is a square area of side 1 km in the city center of Cologne, Germany. Mobility traces are derived from the TapasCologne dataset [50], over the 2:00 PM - 2:30 PM time interval, with an average of 67.16 nodes and an average sojourn time of 84.31 s in the region of interest. Fig.6.10 and Fig.6.11 show the mean number of neighbors per node in the region of interest for the two settings considered. Through simulations, we have verified that in both scenarios, the density and mobility patterns of vehicles are such that only a small portion of content deliveries would be implementable via any of the considered algorithms and that the resulting delivery ratio would be too small (less than 60%, for all algorithms), forcing the cellular network to provide content objects to satisfy requests directly. In such scenarios, thus, the support of nodes from other classes (such as pedestrians) may play a key role in providing additional service capacity and thus increasing the share of content delivery traffic offloaded from the cellular network.

Thus, in both scenarios, we assumed pedestrians to be located uniformly at random within each segment, to be static, and to arrive and leave the segment. The pedestrian arrival process in each segment has been set to have a mean density of pedestrians equal to $\frac{1}{5}$ of the mean vehicle density in the same segment. This has been done considering the sojourn time of a pedestrian in the segment to be uniformly distributed between 120 s and 180 s. All these choices aim at modeling a population of pedestrian nodes characterized by a high level of churn (e.g. due to the effects of indoor/outdoor pedestrian mobility, as pedestrians move between shops or bars) and whose mean density varies proportionally to that of vehicles, as typical in many urban scenarios. The constant of proportionality between these mean densities has been set to have a population of pedestrians substantially smaller than that of vehicles and thus still have the majority of service capacity implemented through vehicular users' resources.

We have assumed all algorithms include pedestrians in the population of nodes involved in the content delivery process. However, we assumed that pedestrians can be producers but not requesters, as the content is still of interest to vehicular users only. In addition, pedestrians who become producers discard the content when exiting the segment, and in any case, 2 minutes after getting the content object.

To evaluate the resource cost of each algorithm, we have adopted the cost function in 6.3 with $\beta = 0.005$. This value has been chosen to model typical vehicular (and, more generally, ad-hoc) scenarios where storage resources are much more abundant than bandwidth. In addition, in the component of the cost function which accounts for the utilization of communication resources (eq. 6.2), we have assumed the unitary costs $X^{G}_{z,i,c}$ and $X^{D2D}_{z,i,c}$ to be invariant across node classes and road segments. To maximize the use of D2D-based content delivery, and thus the offloading of cellular infrastructure, we have assumed that $\frac{X^{G}}{X^{D2D}} = 20000$, i.e. that the cost of a content retrieval from the cloud is several orders of magnitude larger than that of D2D content delivery.

Fig. 6.12 and Fig. 6.13 show the cost of the delivery process (in the Luxembourg off-peak and in the Cologne off-peak scenarios, respectively), and its associated D2D delivery ratio (i.e. the fraction of requests satisfied via D2D), for our DeepNDN approach as well as for the competing schemes. In addition, it displays the performance of the DeepNDN scheme without pedestrian support. These results show that, in all scenarios and configurations, pedestrian support to DeepNDN is key to achieving the highest levels of delivery ratio and resource efficiency among all considered al-

gorithms, and despite all of DeepNDN's competing approaches also takes advantage of pedestrian support.

As expected, in such a realistic and heterogeneous scenario, none of the approaches is able to reach a 0.9 delivery ratio within a very short max delay (5 s). I.e. with such a short max delay, all schemes require very frequently direct support from infrastructure for content delivery. However, even in these conditions, DeepNDN's resource cost is substantially smaller (55% lower in the Luxembourg scenario and 84% in Cologne) than the best-performing competing approach. This shows how, also in these heterogeneous, two-class scenarios, our DNN-based resource orchestration is able to produce very efficient DeepNDN management strategies.

In both scenarios, we have also evaluated performance with a 60 s maximum content retrieval delay. Unsurprisingly, a larger maximum delay allows lower costs and higher delivery ratios, for all schemes. However, giving more time for content retrieval to all schemes benefits them differently, and the performance gap between DeepNDN and the competing schemes increases when loosening the delay constraint. For the delivery ratio, this is because longer maximum delays increase the solution space for DeepNDN, which, differently than competing approaches, is based on two different and coordinated content delivery strategies. And for the cost, the performance gap widening is due to the effectiveness of our DNN-based orchestration of DeepNDN parameters.

Finally, our results suggest that the performance advantage of DeepNDN with respect to CEDO is larger in the Cologne scenario. Indeed, the node mean sojourn time in the region of interest is substantially lower in the Cologne Scenario than in Luxembourg. And CEDO is mainly based on opportunistic replication, thus requiring more time on average for delivery than schemes based (at least in part) on NDN mechanisms.

Simulation realism. Several aspects of real-world operating conditions for vehicular networks have not been considered in our assessment. These include a realistic simulation of signal propagation in these scenarios, the effects of connection setup latency, and the impact of the size of the content object on the duration of the exchange among vehicles, and thus on the effectiveness of such exchange. However, these effects are likely to impact in an equal manner the performance of our DeepNDN approach as well as that of the baselines, and thus not having included them in the simulations does not affect the validity of the assessment. In addition,

our management strategy is designed to easily and effectively account for them. Indeed, it is based on an ML model trained on measured data that the coordination function regularly collects during the operations of our DeepNDN scheme, and which thus in real deployments can account for all of the above-mentioned factors in the effectiveness of the content delivery scheme.

Scalability. An increase in the number of vehicles in a region could imply, in principle, an increment in the amount of data that the coordination function collects, and thus in the utilization of network and computing resources, as more data implies also a larger training set and a computationally heavier training process. However, the parameters that our approach takes as input for the CNN are averages over time and across all users belonging to the same road segment and node class. Therefore, increasing the number of vehicles in a region does not imply an increment of the computing resources required by the training process. Moreover, as such a process takes place offline, the time it requires does not affect the performance of the DeepNDN scheme. We also observe that the training data does not need to be collected in real-time. Hence, the overall amount of data collected by the coordination function can be greatly reduced through the use of techniques of data compression and aggregation among users, thus saving bandwidth. Thanks to its reliance on averages across time and road segments, our scheme also scales well when increasing the size of the region of interest. During our experiments, we tested several choices for the size of the region, and we have verified that even in the limit case in which the region encompasses the whole city, the computing load of the CNN training process remains acceptable, and the accuracy of our DeepNDN does not decrease with respect to the one square km case.

6.8 Conclusion

In this work, we have proposed DeepNDN, a communication scheme based on the joint application of NDN and of probabilistic spatial content caching, for content retrieval in fragmented and dynamic vehicular ad-hoc networks. We have presented a data-based approach for the dynamic management of DeepNDN, capable of achieving a target delivery ratio in a resource-efficient manner, by locally modulating the content diffusion process. Results suggest that the synergetic application of NDN and probabilistic location-based content caching, supported by a learning-based orchestration of the content delivery process, can satisfy stringent requirements in

terms of maximum delivery delay while minimizing content retrieval from the cloud, in a large variety of settings. As a follow-up, we intend to extend our approach to scenarios with drones on wheels and UAVs, for which optimal routes have to be derived. A further direction of investigation concerns the extension of the DeepNDN approach to those scenarios in which mobility patterns are influenced by the spreading of content (e.g., in the delivery of traffic jams notification or hazard warning messages).

6.9 Acknowledgment

This work has been undertaken under the Chist-Era ABIDI project, and it has been partially supported by the COST INTERACT project.

Bibliography

[1] Kaoutar Ahed, Maria Benamar, Ayoub Ait Lahcen, and Rajae El Ouazzani. Forwarding strategies in vehicular named data networks: A survey. *Journal of King Saud University-Computer and Information Sciences*, 34(5):1819–1835, 2022.

[2] Mays F Al-Naday, Martin J Reed, Dirk Trossen, and Kun Yang. Information resilience: source recovery in an information-centric network. *IEEE network*, 28(3):36–42, 2014.

[3] Kamran Ali, Huan X. Nguyen, Quoc-Tuan Vien, Purav Shah, and Zheng Chu. Disaster management using d2d communication with power transfer and clustering techniques. *IEEE Access*, 6:14643–14654, 2018.

[4] O. Ascigil, V. Sourlas, I. Psaras, and G. Pavlou. Opportunistic off-path content discovery in information-centric networks. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 1–7, 6 2016.

[5] A. A. V. Castro, G. Di Marzo Serugendo, and D. Konstantas. Hovering information - self-organising information that finds its own storage. In *IEEE SUTC*, pages 193–200, 6 2008.

[6] Saqib Rasool Chaudhry, Hadi Tabatabaee Malazi, Sangita Dhara, Aqeel Haider Kazmi, and Siobhán Clarke. Toward context-aware information dissemination in autonomous networks of vehicles. *IEEE Communications Standards Magazine*, 7(2):8–15, 2023.

[7] Nan Cheng, Feng Lyu, Jiayin Chen, Wenchao Xu, Haibo Zhou, Shan Zhang, and Xuemin Shen. Big data driven vehicular networks. *Ieee Network*, 32(6):160–167, 2018.

[8] Francois Chollet. *Deep Learning with Python and Keras: The practical manual from the developer of the Keras library.* MITP-Verlags GmbH & Co., 2018.

[9] L. Codeca, R. Frank, and T. Engel. Luxembourg SUMO Traffic (LuST) Scenario. In *IEEE VNC*, pages 1–8, Dec 2015.

[10] Ramya Daddanala, Vekata Mannava, Lo'ai Tawlbeh, and Mohammad Al-Ramahi. Vehicle to vehicle (v2v) communication protocol: Components, benefits, challenges, safety and machine learning applications, 2021.

[11] Joao M Duarte, Torsten Braun, and Leandro A Villas. Source mobility in vehicular named-data networking: An overview. In *Ad Hoc Networks*, pages 83–93. Springer, 2018.

[12] Joao M Duarte, Torsten Braun, and Leandro A Villas. MobiVNDN: A distributed framework to support mobility in vehicular named-data networking. *Ad Hoc Networks*, 82:77–90, 2019.

[13] Cunha Felipe, Azzedine Boukerche, Leandro Villas, Aline Viana, and Antonio Loureiro. Data communication in vanets: A survey, challenges and applications. *Ad Hoc Networks*, 03 2014.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press Cambridge, March 2016.

[15] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida. Navigo: Interest forwarding by geolocations in vehicular Named Data Networking. In *WoWMoM*, pages 1–10. IEEE, June 2015.

[16] Giulio Grassi, Davide Pesavento, Lucas Wang, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. Vehicular inter-networking via named data. *Proceedings of the IEEE*, 10 2013.

[17] A. Hanggoro and R. F. Sari. Performance evaluation of the Manhattan mobility model in vehicular ad-hoc networks for high mobility vehicle. In *IEEE COMNETSAT*, 12 2013.

[18] Wenxue He, Huafu Li, Xiao Zhi, Xinghua Li, Jingyuan Zhang, Qian Hou, and Yiyao Li. Overview of v2v and v2i wireless communication for cooperative vehicle infrastructure systems. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 127–134, 2019.

[19] Takamasa Higuchi, Gurjashan Singh Pannu, Falko Dressler, and Onur Altintas. Content Replication in Vehicular Micro Cloud-based Data Storage: A Mobility-Aware Approach. In *VNC*, Taipei, Taiwan, 12 2018. IEEE.

[20] Esa Hyytiä, Jorma Virtamo, Pasi Lassila, Jussi Kangasharju, and Jörg Ott. When does content float? Characterizing availability of anchored information in opportunistic content sharing. In *INFOCOM*, pages 3137–3145. IEEE, April 2011.

[21] Hasan MA Islam, Andrey Lukyanenko, Sasu Tarkoma, and Antti Yla-Jaaski. Towards disruption tolerant icn. In *ISCC*, pages 212–219. IEEE, 2015.

[22] Steffi Jayakumar. A review on resource allocation techniques in d2d communication for 5g and b5g technology. *Peer-to-Peer Networking and Applications*, 14:243–269, 2021.

[23] Daniel Jiang and Luca Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*, 06 2008.

[24] Caihong Kai, Hui Li, Lei Xu, Yuzhou Li, and Tao Jiang. Energy-efficient device-to-device communications for green smart cities. *IEEE Transactions on Industrial Informatics*, 14(4):1542–1551, 2018.

[25] Eirini Kalogeiton and Torsten Braun. Infrastructure-assisted communication for ndn-vanets. In *WoWMoM*, pages 1–10. IEEE, 2018.

[26] SM Kazmi, Latif U Khan, Nguyen H Tran, and Choong Seon Hong. 5g networks. In *Network Slicing for 5G and Beyond Networks*, pages 1–12. Springer, 2019.

[27] Hakima Khelifi, Senlin Luo, Boubakr Nour, Hassine Moungla, Yasir Faheem, Rasheed Hussain, and Adlen Ksentini. Named data networking in vehicular ad hoc networks: State-of-the-art and challenges. *IEEE Communications Surveys & Tutorials*, 22(1):320–351, 2020.

[28] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. SUMO (Simulation of Urban MObility), an open-source traffic simulation. In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM)*. MESM, 2002.

[29] Ge Ma, Zhen Chen, Junwei Cao, Zhenhua Guo, Yixin Jiang, and Xiaobin Guo. A tentative comparison on cdn and ndn. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2893–2898, 2014.

[30] G. Manzo, S. Otálora, T. Braun, M. Ajmone Marsan, G. Rizzo, and H. Nguyen. Deepfloat: Resource-efficient dynamic management of vehicular floating content. In *ITC 31*, 8 2019.

[31] Gaetano Manzo, Marco Ajmone Marsan, and Gianluca Rizzo. Performance modeling of vehicular floating content in urban settings. In *29th International Teletraffic Congress (ITC 29)*, volume 1, pages 99–107. IEEE, September 2017.

[32] Gaetano Manzo, Marco Ajmone Marsan, and Gianluca A Rizzo. Analytical models of floating content in a vehicular urban environment. *Ad Hoc Networks*, 88:65–80, 2019.

[33] Gaetano Manzo, Juan Sebastian Otalora, Marco Ajmone Marsan, and Gianluca Rizzo. A Deep Learning Strategy for Vehicular Floating Content Management. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):159–162, January 2019.

[34] Gaetano Manzo, Ridha Soua, Antonio Di Maio, Thomas Engel, Maria Rita Palattella, and Gianluca Rizzo. Coordination mechanisms for floating content in realistic vehicular scenarios. In *IEEE MobiWorld*, 4 2017.

[35] Alex McMahon and Stephen Farrell. Delay-and disruption-tolerant networking. *IEEE Internet Computing*, 13(6):82–87, 2009.

[36] Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8, 2010.

[37] Edo Monticelli, Benno M Schubert, Mayutan Arumaithurai, Xiaoming Fu, and KK Ramakrishnan. An information centric approach for communications in disaster situations. In *LANMAN*, pages 1–6. IEEE, 2014.

[38] Francisco Neves dos Santos, Benjamin Ertl, Chadi Barakat, Thrasyvoulos Spyropoulos, and Thierry Turletti. Cedo: Content-centric dissemination algorithm for delay-tolerant networks. In *ACM MSWIM*, pages 377–386. ACM, 2013.

[39] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *Communications Surveys & Tutorials, IEEE*, PP(99):1–18, 2014.

[40] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[41] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju. Floating content: Information sharing in urban areas. In *PerCom 2011*, pages 136 –146, 3 2011.

[42] Neil Scott, Gladys Mcpherson, Craig Ramsay, and Marion Campbell. The method of minimization for allocation to clinical trials. a review. *Controlled clinical trials*, 01 2003.

[43] Mohammad Shahrokh Esfahani and Edward R. Dougherty. Effect of separate sampling on classification accuracy. *Bioinformatics*, 11 2013.

[44] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 2011.

[45] Vasilis Sourlas, Leandros Tassiulas, Ioannis Psaras, and George Pavlou. Information resilience through user-assisted caching in disruptive content-centric networks. In *IFIP Networking*, pages 1–9. IEEE, 2015.

[46] Terry M. Therneau. How many stratification factors are "too many" to use in a randomization plan? *Controlled Clinical Trials*, 14(2):98–108, 1993.

[47] M. Torrent-Moreno, J. Mittag, P. Santi, and H. Hartenstein. Vehicle-to-vehicle communication: Fair transmit power control for safety-critical information. *IEEE Transactions on Vehicular Technology*, 9 2009.

[48] Ion Turcanu, Thomas Engel, and Christoph Sommer. Fog Seeding Strategies for Information-Centric Heterogeneous Vehicular Networks. In *IEEE VNC*, pages 282–289, Los Angeles, CA, 12 2019. IEEE.

[49] Gareth Tyson, John Bigham, and Eliane Bodanese. Towards an information-centric delay-tolerant network. In *2013 IEEE Conference on Computer Communications Workshops*, pages 387–392. IEEE, 2013.

[50] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 99(PrePrints):1, 2013.

[51] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *International ICST Conference on Simulation Tools and Techniques*, 2008.

[52] Siyang Wang, Jian Deng, Weigang Wu, and Jieying Zhou. RSU Controlled Named Data Networking for Traffic Information Dissemination in Vehicular Networks. In *IEEE SmartWorld*. IEEE, October 2018.

[53] Xiaonan Wang and Xingwei Wang. Vehicular Content-Centric Networking Framework. *IEEE Systems Journal*, 13(1):519–529, March 2019.

[54] Yu-Ting Yu, Joshua Joy, Ruolin Fan, You Lu, Mario Gerla, and MY Sanadidi. Dt-ican: A disruption-tolerant information-centric ad-hoc network. In *2014 IEEE Military Communications Conference*, pages 1021–1026. IEEE, 2014.

[55] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

CHAPTER 7

Context-Aware Orchestration of Energy-Efficient Gossip Learning Schemes

Published in IEEE World AI IoT Congress (AIIoT 2024), Pages: 192-198.
Date: May 2024.
Publisher: IEEE.
ISBN: 979-8-3503-8780-3.

Abstract– Fully distributed learning schemes such as Gossip Learning (GL) are gaining momentum due to their scalability and effectiveness even in dynamic settings. However, they often imply a high utilization of communication and computing resources, whose energy footprint may jeopardize the learning process, particularly on battery-operated IoT devices. To address this issue, we present Optimized Gossip Learning (OGL), a distributed training approach based on the combination of GL with adaptive optimization of the learning process, which allows for achieving a target accuracy while minimizing the energy consumption of the learning process. We propose a data-driven approach to OGL management that relies on optimizing in real-time for each node the number of training epochs and the choice of which model to exchange with neighbors based on patterns of node contacts, models' quality, and available resources at each node. Our approach employs a DNN model for dynamic tuning of the aforementioned parameters, trained by an infrastructure-based orchestrator function. We performed our assessments on two different datasets, leveraging time-varying random graphs and a measurement-based dynamic urban scenario. Results suggest that our approach is highly efficient and effective in a broad spectrum of network scenarios.

7.1 Introduction

Distributed learning schemes are poised to become one of the key enablers of future 6G networks, as they allow fast and efficient training of complex and large-scale models while delivering better reliability and fault-tolerance than traditional, centralized approaches. Among these, Gossip Learning (GL) schemes are of special interest, as they do not require uploading models to a parameter server, thus offering better robustness and scalability.

Originally introduced in [17], GL schemes train ML models over decentralized data via direct model gossiping among nodes. Several versions of GL have been proposed for dynamic settings [6, 7]. In these works, the changing topology is a result of varying patterns of network connectivity, changes in node availability (e.g. due to node duty cycling or battery depletion), and churn, among others, as it is often the case in realistic mobile edge and vehicular scenarios and use cases. GL is based on a combination of iterative local training and model exchange over wireless channels. Both tasks on battery-operated, resource-constrained IoT and edge devices might imply rapid energy budget depletion, potentially slowing down and jeopardizing the whole learning process.

Recently, several works have focused on decreasing the energy footprint of distributed learning, albeit in server-based architectures such as Federated Learning (FL). [22] reduces the amount of exchanged models in FL by decreasing the number of communication rounds. Other approaches focus instead on reducing the number of exchanged models at each round. These methods consider factors such as the size and quality of a node's local dataset [2, 14, 13, 21], the rate of network evolution [22, 20], and the node's trustworthiness [9]. All these works, however, consider a static network. [5] proposes a Gossip Learning scheme, evaluating the effect of the number of models merged at each round in a vehicular network. It shows that the resource-optimal value for these parameters is highly context-specific. However, it considers measurement-based mobility patterns, making it hard to untangle dependencies between mobility features and the performance of the training scheme. All these works leave unanswered the critical question of when and which nodes should exchange models in a fully distributed learning scheme to achieve a target performance (in terms of accuracy and convergence speed) in a dynamic network in an energy-optimal manner.

In this paper, we consider a scenario of reference in which cellular connectivity is

pervasive, and it allows taking advantage of an orchestration function that monitors the gossip-based learning process without requiring infrastructure-based exchanges of training data or ML models. The primary contributions of this paper are:

- We propose OGL, a gossip-based training strategy for dynamic networks capable of adapting to a wide range of network topologies and dynamic settings.
- We present a data-driven approach for the dynamic management of OGL, which achieves a target performance in a resource-efficient manner by proactively adapting the models' distribution and training parameters to local conditions. The approach employs a Deep Neural Network (DNN) model, which is trained offline and distributed to all nodes at the start of the learning process, enabling each node to tune the main parameters of the OGL scheme adaptively.
- We assess the effectiveness and efficiency of our approach by leveraging time-varying random graphs and a measurement-based mobility trace. These results suggest that it substantially outperforms a set of baseline approaches while achieving the target minimum accuracy in all of the considered scenarios.

7.2 System model

We consider a set V of nodes, with cardinality $|V|$ (modelling, e.g., mobile devices, UAVs, or connected vehicles) moving within a specific region according to an arbitrary mobility model and during a predefined time interval T (the *observation interval*). Let $v \in \mathbb{N}$ denote the unique identifier of a node. We assume nodes can communicate directly with each other through wireless peer-to-peer (P2P) communications (e.g., using DSRC or Bluetooth Low Energy [8]). Furthermore, each node is equipped with a cellular network interface. Two nodes can exchange information whenever they are in *contact*, that is, within each other's transmission range. We assume these exchanges are always unicast (one-to-one). However, the proposed scheme can be easily extended to incorporate the effects of multicasting and broadcasting. We assume there is a coordination function (possibly implemented by a Software Defined Network Controller(SDNC) [16]) in the region, and it resides within the cellular access network. The coordination function comprises an auxiliary ML model M_{tune}. Through its cellular network interface, the coordination function transmits M_{tune} to the nodes entering the region. Hence, the architecture

of this model is equal for all nodes. Note that all communications are P2P, except for the initial dissemination of the M_{tune} model from the coordination function to nodes. Every node independently employs M_{tune} to fine-tune and adjust its learning parameters. Moreover, we assume each node entering the region possesses a local model w_v and a *local dataset*, partitioned in the training set $\mathcal{D}_v$, and validation set $\mathcal{S}_v$, which generally differ in size and composition for each node. The choice of the validation and training set size is context-specific. We also assume that the observation interval is segmented into I *slots* of equal size, short enough that node mobility patterns can be considered not to vary substantially within each slot. Let $t \in 1, .., I$ be the label of slots.

7.3 The OGL Approach to Energy-Efficient Gossip Learning

7.3.1 A Gossip-Based Collaborative Training Algorithm

We assume all nodes in the region train their local model through a gossip-based cooperative learning algorithm, denoted as OGL, and based on P2P model exchanges among nodes. Such an approach is orchestrated by a cellular-based coordination function, without requiring the exchange of the trained models or training data (which would potentially expose it to privacy breaches) between each node and the coordinator. At the beginning of the scheme, all nodes present in the region randomly generate an initial local model w_0. We assume the generation procedure to produce the same random initial model for all nodes. Similarly, after the beginning of the scheme, whenever a node enters the region, it generates w_0 following the same procedure. Starting from w_0, every node elaborates an ML model (which we assume will be used by the node itself. e.g. to carry out the same inference task, e.g. trajectory prediction or image recognition) by alternating local training on each node's local training set, with model aggregation with models received from neighbours. Moreover, to all nodes joining the scheme, the coordination function delivers an *auxiliary ML model* M_{tune}. Each node employs M_{tune} for the adaptive tuning of some key parameters of the learning process. Such dynamic management of the learning process at each node is based on each node's available hardware resources and power budget. In addition, it also accounts for each node's context in terms of the number of neighbours, the speed at which they vary over time, and

the quality and quantity of their local model (i.e., in terms of mean accuracy or loss), among others. Each node then uses the M_{tune} to modulate the number of local training epochs and to choose, among its neighbours, those whose local model should be requested and used for improving the node's local model, as we will explain later.

Then, at every time slot, the OGL algorithm proceeds through three *phases*. The duration of each phase can be tuned and adapted to the specific training task and setup, and it does not need to be synchronized across nodes.

In the *training* phase, each node in the region applies M_{tune} to get the number of epochs $Z_{v,t}$ that it has to train its local model over its local dataset and train its model accordingly. Subsequently, it assesses its local model over its validation set $\mathcal{S}_v$ to derive the loss value l_v used in the next phase. The choice of the loss function is context-specific.

In the *communication* phase, nodes exchange the loss value of their local model with their neighbours. Each node then employs M_{tune} to identify the neighbouring nodes from which it should request the transfer of their local models. The node then initiates a request directed towards the selected nodes, soliciting their respective models. In response, these requested nodes transmit their models if they are still within range of each other. During this phase, a node may not request any model, e.g. because it has no neighbours or when the M_{tune} indicates that no model is worth requesting among the available neighbours' models. We assume that the connectivity between nodes is relatively stable while exchanging models.

Finally, in the *merging* phase, each node combines the models received from the chosen neighbours and its local model to produce a new version of its local model. The merging procedure consists of a weighted averaging method. The weights associated with each merged model are computed via the DFed Pow strategy [7]. In DFed Pow, the weight of each model to be merged is a function of the inverse of loss calculated on the node's validation set. Note, however, that our approach is more general and does not rely on a specific algorithm for calculating the weights for merging.

The three phases are repeated at each time slot until a stopping condition is met (e.g., after a maximum number of iterations, when the average local models' accuracy surpasses a certain threshold or when there is no significant improvement in the model's accuracy over several rounds). Regardless of the reason, the final

round at which the algorithm stops is called the cut-off round.

7.3.2 Formulation of the energy optimization problem

The main goal of our OGL approach is to enable the energy-efficient training of an ML model in a distributed manner. As mentioned, this is enabled by an orchestrator function that elaborates and distributes the M_{tune} model among all the nodes entering the region. Given the node context, as well as some key parameters of the system and the training task, such a model enables each node to tune the number of training epochs and the set of ML models to merge, which allows for achieving a given target accuracy while minimizing a cost function which models the overall energy cost of the training process.

In what follows, we formalize the energy optimization problem that the orchestration function tries to solve. The cost function we consider is the sum of two components. The first one accounts for the computing costs. Generally, the energy consumption associated with a computation task is determined by CPU(or GPU) usage and memory resources [19]. Those, in turn, depend upon the architecture implemented, the quantity of data it processes, and the node's characteristics. In this work, we assume all nodes have the same computing power. Then, the energy required to run the (local) training process is:

$$S(Z) = \sum_{t \in T} \sum_{v \in V} Z_{v,t} d_v (e_g + e_s) \tag{7.1}$$

- $Z_{v,t}$ depicts the number of epochs required by node v to train its local model using the local dataset at time slot t;
- e_g is the energy consumed by the CPU or GPU to perform one training epoch on one sample;
- e_s is the energy required to provide storage and memory resources for the training of one epoch on a sample;
- d_v denotes the number of samples of the local training set of node v;
- T is the label of the slot at which convergence happens.

The energy consumed for computing the loss of the local model on the validation

set is modelled as follows:

$$\Gamma = \sum_{t \in T} \sum_{v \in V} s_v (e_e + e_{es}) \tag{7.2}$$

e_e and e_{es} are the energy consumed by the CPU (or GPU) and energy required to provide storage and memory to evaluate the local model on one dataset sample. s_v indicates the size of the validation set at node v.

The second component of the cost function accounts for the communication costs. The communication costs consider the exchanges between nodes:

$$C(k) = C^{d2d} \sum_{t \in T} \sum_{v \in V} h_{v,t} L + k_{v,t} (M + R) \tag{7.3}$$

- C^{d2d} is the cost per byte of a d2d (peer to peer) transfer
- $\mathcal{H}_{v,t}$ is the set of neighbors of node v at time slot t, of cardinality $h_{v,t}$;
- $\mathcal{K}_{v,t} \subseteq H_{v,t}$ is the set of chosen neighbours of node v at time slot t from which models to be merged are retrieved, of cardinality $k_{v,t}$.
- L, R and M are the message size containing loss value, request and a local model, respectively.

Note that such a cost function neglects the cost of model merging, as it is usually negligible [7, 5]. Let $\mathcal{Z} = \{Z_{v,t}\}$, and $\mathcal{K} = \{\mathcal{K}_{v,t}\}$. Thus, an optimal OGL orchestration scheme is a solution to the following optimization problem:

Problem 1.

$$\underset{\mathcal{Z},\mathcal{K}}{minimize}\ C(k) + \beta(S(Z) + \Gamma) \tag{7.4}$$

Subject to:

$$r \geq r^0 \tag{7.5}$$

Where r denotes the mean accuracy of the trained model across all nodes achieved at convergence, and r^0 is its target minimum value. By varying β, it is possible to adapt the cost function to settings with different resource availability on user devices and at the cellular network and to different incentive schemes for resource sharing and cooperation.

7.3.3 OGL architecture, components and functions

The OGL approach aims at solving Problem 1 by training a DNN-based auxiliary ML model, which enables nodes to adapt in real-time the learning process to available contributions by neighbours and, more generally, to each node's context. The auxiliary model is trained by the orchestrator on a training dataset whose data points are labelled by simulating the system.

We assume the orchestrator regularly collects data from every node, which is used as the feature set of the auxiliary model that it has to train. The data collected are those that are well known from the state of the art to be relevant to the training process and its efficiency. These include computing and communication costs, the number of neighbours for each node at each time slot, the size of the local dataset, the available computing power, and the initial power budget of each node. Such a choice of features as input parameters for the auxiliary model is, however, one of many possible, and our approach is independent of it. From each set of input parameters, the orchestrator derives a set of full system configurations by associating to the input parameters a random value for each of the parameters in $\mathcal{Z}$ and a random subset $\mathcal{K}$ of each node's neighbours. These inputs are fed to a simulator, which labels them with the outputs and performance metrics of the distributed training scheme. Specifically, for every node and every time slot in a given time interval during which the orchestrator has collected data, the simulation derives the local model accuracy, the loss value, and the energy budget of each node. In such a way, a training set is produced, which is then used to train a DNN model.

This model is trained and evaluated using a k-fold cross-validation approach [18], with 10 folds. The architecture of the model is a multi-layer perceptron composed of four layers. The multiple layers allow models to be more efficient at learning complex features [11]. The initial layer is a dense layer with 64 neurons and a rectified linear unit (ReLU) activation function. The second layer is a flattening layer, which reshapes the input to a one-dimensional array. The third and fourth layers are dense layers with 32 and 16 neurons, respectively, and ReLU activation functions. The final layer is a dense layer with two neurons. The model is compiled with a mean squared error loss function and the Adam optimizer. Early stopping and model checkpoint callbacks are used to prevent overfitting and save the best model. This approach ensures a robust evaluation of the model performance, as it assesses the model's ability to generalize to unseen data. Note that the selection of

CNN parametrs	MNIST	CIFAR-10
Input shape	(28,28,1)	(32,32,3)
Batch size	32	64
Learning rate	0.0001	0.001
Number of neurons	100	100
Momentum	0.9	0.60
Kernel dimension	3	3
Number of filters	32	32
Number of outputs	10	10

Table 7.1: Parameter values used to train the CNN model on the CIFAR-10 and MNIST datasets.

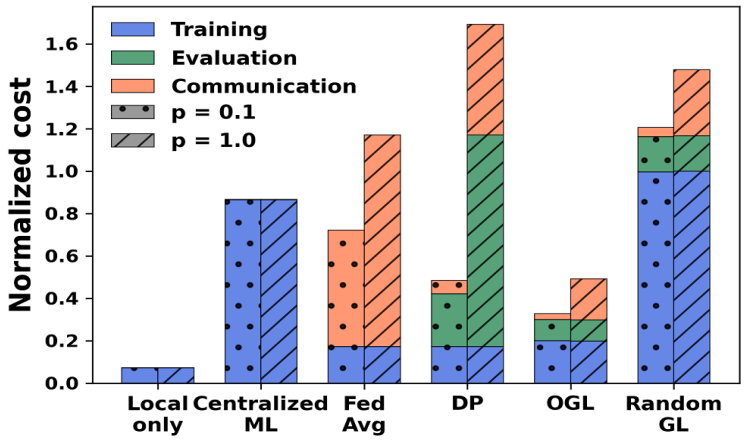


Figure 7.1: Comparative analysis of cost function on the MNIST dataset with $|V| = 6$ and various values for p. Results are presented with a confidence interval of 95% and an error margin of 2%.

parameters in the model is either empirical or based on extensive usage in the state-of-the-art models. After the training and evaluation process, the best-performing model, referred to as M_{tune}, is saved for future use. This model encapsulates the optimal parameters learned during the training process. At runtime, the orchestrator function disseminates the M_{tune} model to all nodes entering the region. Then, at each time slot, each node feeds the M_{tune} model with its own data to determine the optimal number of local training iterations (number of epochs) and the optimal set of models to merge to achieve the given target accuracy while minimizing the energy cost of the whole process.

Algorithm	Acc	F1	Loss	Precision	Recall	Cut-off round
Centralized ML	0.87	0.87	0.45	0.88	0.87	300
Fed Avg	0.85	0.84	0.57	0.87	0.85	500
Local only	0.36	0.42	3.2	0.42	0.56	200
DP	0.58	0.60	1.42	0.68	0.67	500
OGL	**0.88**	**0.88**	**0.44**	**0.88**	**0.89**	320
Random GL	0.51	0.56	1.73	0.7	0.6	500

Table 7.2: Performance metrics of OGL at the cut-off round, compared to baselines on the MNIST dataset, with $|V| = 6$ and $p = 1$. Results are presented with a 98% confidence interval and a maximum error margin of 1%.

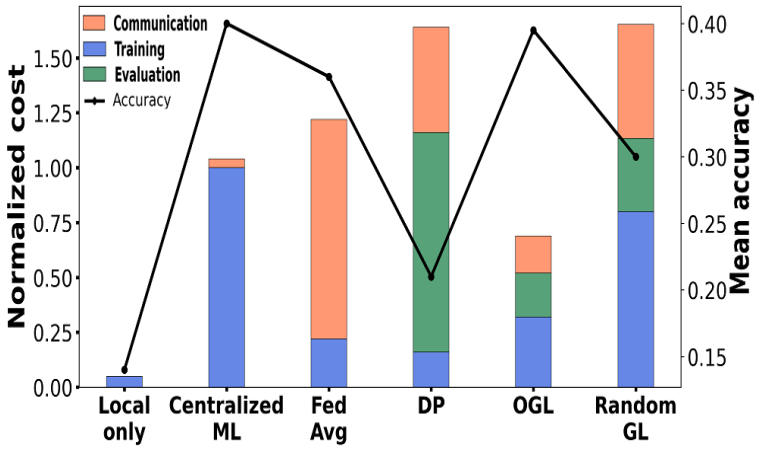


Figure 7.2: Comparative analysis of cost function and mean accuracy at convergence on the CIFAR-10 dataset, with $|V| = 6$ and $p = 1$. Results are presented with a confidence interval of 95% and an error margin of 2%.

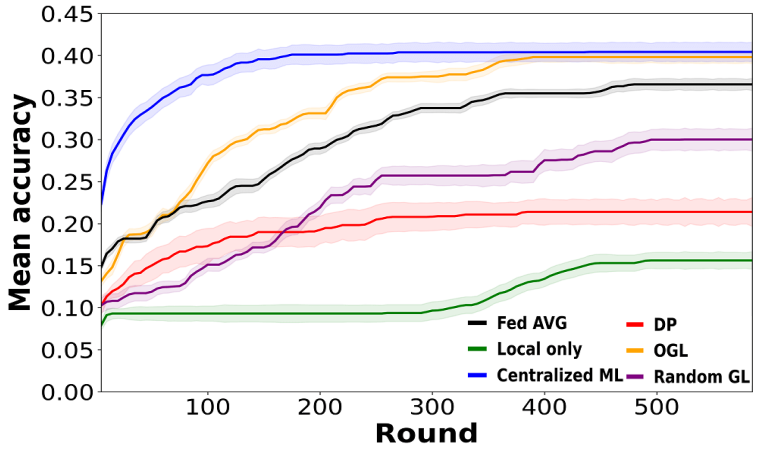
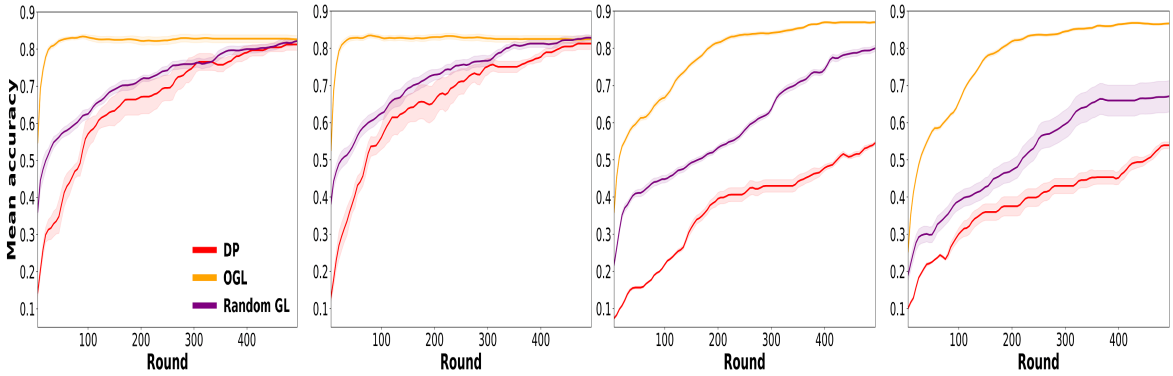
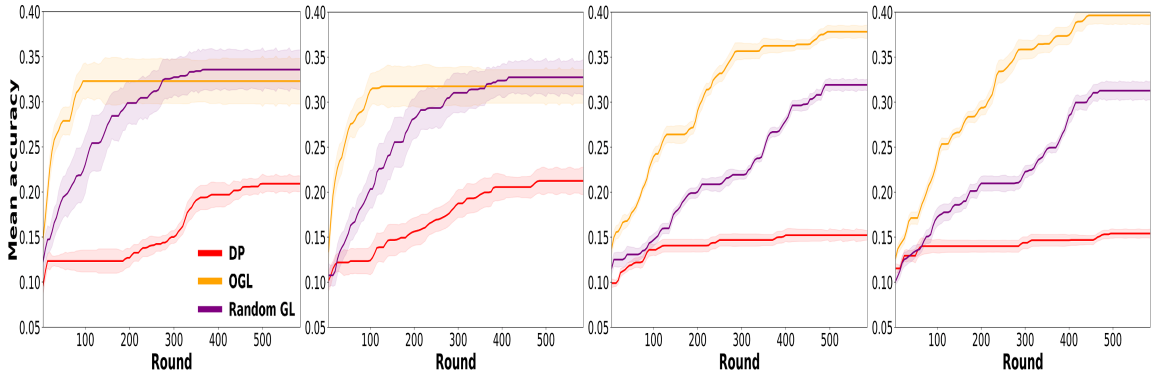


Figure 7.3: Mean accuracy versus round of our OGL algorithm compared to baselines using the CIFAR-10 dataset, with $|V| = 6$ and $p = 1$. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.



(a) $|V| = 3, p = 0.1$ (b) $|V| = 3, p = 0.7$ (c) $|V| = 12, p = 0.1$ (d) $|V| = 12, p = 0.7$

Figure 7.4: Mean accuracy versus round for OGL, random GL, and DP algorithm using the MNIST dataset, for different values of number of nodes in the system and edge probability. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.



(a) $|V| = 3, p = 0.1$ (b) $|V| = 3, p = 0.7$ (c) $|V| = 12, p = 0.1$ (d) $|V| = 12, p = 0.7$

Figure 7.5: Mean accuracy versus round for the OGL, Random GL, and DP algorithms, for the CIFAR-10 dataset, for different values of number of nodes in the system and edge probability. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.

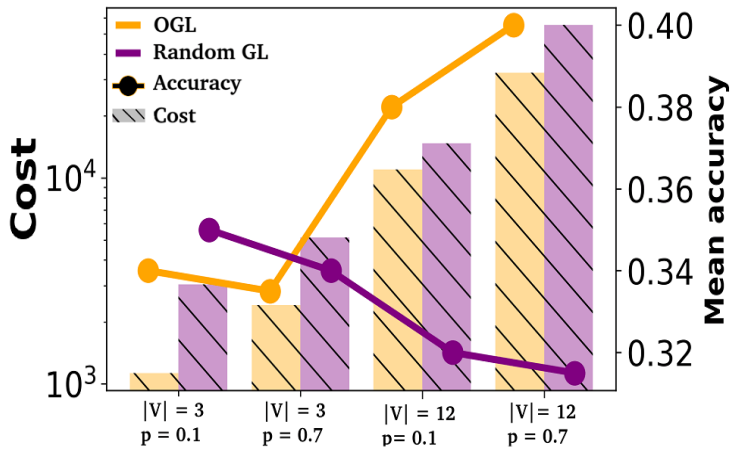


Figure 7.6: Comparison of mean accuracy at convergence time versus cost for the OGL and random GL algorithms over different network configurations using the CIFAR-10 dataset. Results are presented with a 98% confidence interval and a maximum error margin of 2%.

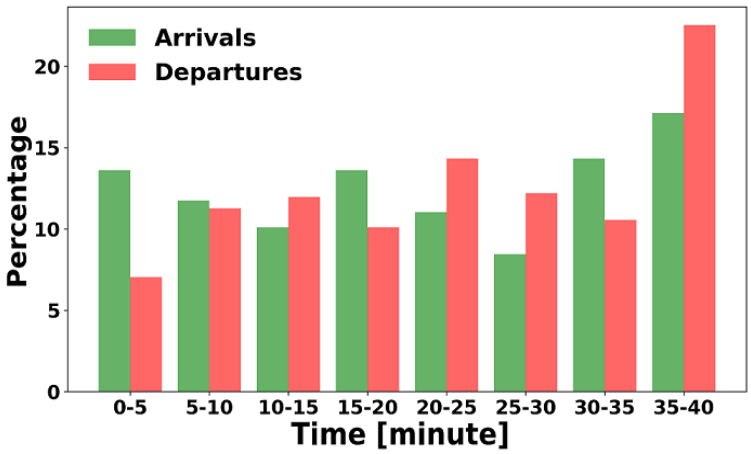


Figure 7.7: Histogram representing the percentage of vehicle arrivals and departures across different time slots in the Luxembourg City off-peak scenario (12:00-12:40 PM). Time on the x-axis shows the time from the beginning of the scheme.

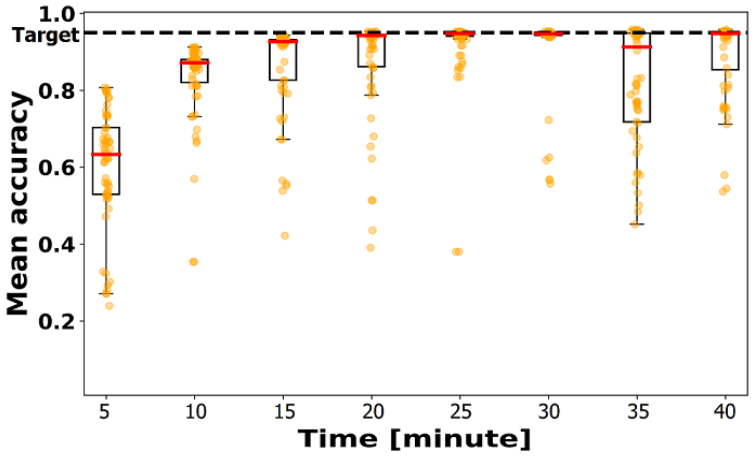


Figure 7.8: Distribution of mean accuracy at every five-minute interval for the OGL algorithms in the Luxembourg City off-peak scenario (12:00-12:40 PM) using the MNIST dataset. Each point is the mean accuracy of a single vehicle, averaged in the time interval. The target line shows the target accuracy (95%) obtained using the centralized ML training on the union of the local dataset of all vehicles. Time on the x-axis corresponds to the time from the beginning of the scheme.

7.4 Numerical assessment

To assess the effectiveness of our OGL approach in dynamic settings, we consider a set of V nodes which need to perform a handwritten digit recognition task (MNIST dataset [12]) or object recognition (CIFAR-10 dataset [10]). We assume that each node in the system is endowed with a *local dataset* of different sizes and randomly selected without replacement from the original MNIST and CIFAR-10 datasets. The resulting dataset size, denoted by d_v, falls within the range of 50-350 samples for each node. This implied a training set size ranging from 600 kB to 3.2 MB when utilizing the CIFAR-10 dataset and between 224 KB and 645 KB when employing the MNIST dataset. Let us denote the aggregate local dataset across all nodes in the system as the *global dataset*. The size of the global dataset is 700 samples in all scenarios unless stated differently. We aim to ensure that the results remain consistent and are not subject to significant variation due to differences in global information in different scenarios.

In Problem 1, the coefficient β has been set to 1, to ensure both computing and communication costs are equally weighted. The target accuracies have been set to 0.8 and 0.4 for the MNIST and CIFAR-10 datasets, respectively. These targets were set based on the convergence accuracy of a centralized ML model trained on the global dataset. Furthermore, we assume the cost per byte of d2d transfer to be four times less costly than device-to-server transfers. This is because d2d transfers

bypass the need for data routing or server maintenance, making them a more cost-effective solution. We associate a *global test set* obtained by random sampling 20% of the source datasets and ensuring that the local datasets and test set are disjoint. We assume that nodes use a CNN model to perform both inference tasks.

The first layer of the CNN model is a Conv2D layer, which applies a number of convolutional operations to the input image and uses the activation function ReLU. This layer is followed by a MaxPooling2D layer with a 2x2 pool size, which reduces the spatial dimensions of the input. The Flatten layer then transforms the 2D matrix data into a 1D vector. Subsequently, a Dense layer using ReLU activation and He-uniform weight initialization is added. The final layer is another Dense layer, with the number of neurons equal to the number of output classes, and the softmax activation function is used for multi-class classification. The model is compiled with the Stochastic Gradient Descent (SGD) optimizer and categorical cross-entropy loss function. The values of the parameters of each layer are mentioned in Table 7.1. We choose this architecture and parameter values based on two key considerations. Firstly, some choices are widely recognized as effective in extracting shape features from images for the considered datasets [3]. Secondly, we tune some other parameters empirically by conducting a series of experiments.

With the specified parameters, the resulting model size is approximately 320 KB when trained on the MNIST dataset and 1.2 MB when trained on the CIFAR-10 dataset. Note that better energy performance might be achieved by tuning all of the CNN hyperparameters, as they impact the size of the model to be exchanged. However, this is out of the scope of the present work and is left for future developments. We end our simulations after 600 rounds or when the average accuracy across all nodes does not improve by more than 0.5% for 20 consecutive rounds.

In addition to our scheme, we have considered the following baseline approaches:

- *Centralized ML.* In this approach, a central server possesses a dataset identical to the scenario's global dataset, over which it trains the CNN model.
- *Federated Averaging (Fed AVG)* [15]. In this training scheme, a parameter server collects the CNN models trained locally by each node at every round, merges them, and sends the resulting CNN to each node for a new round of local training. For fairness of comparison, we assumed random client subsampling, with an average number of selected clients coinciding with the average number of nodes each node comes in contact with during a round.

- *Decentralized Powerloss (DP)* [5] is a decentralized learning approach. In this approach, all the nodes set the number of local training epochs to 1 and merge the models from all neighbours at a given time slot without exception. In this approach, the weights associated with each model to be merged are derived from a measure of the received models' performance over the node's validation set.

- *Random GL* is derived from OGL algorithm by setting uniformly at random (and independently for each node and time slot) parameters $\mathcal{K}_{v,t}$ and $\mathcal{Z}_{v,t}$, i.e. the number of training epochs and the set of neighbour nodes whose models have to be merged.

- *Local only*, in which each node trains the local model only on its local dataset, with no data or models exchanged with neighbouring nodes or a server.

The size of local datasets varies across nodes, leading to an uneven distribution of classes. It requires using various performance metrics to compare the effectiveness of our algorithm with baseline methods.

In the first set of experiments, we considered scenarios with different numbers of nodes in the network, specifically $|V| = [3, 6, 12]$. In addition, we model the connectivity graph resulting from node mobility via an Erdős-Rényi dynamic random graph [4]. It is thus a sequence of graphs, each associated with a time slot. In this type of graph, an edge is established between two nodes with probability p, independent from other edges. Then, at each time slot, the connectivity graph stays constant, but possibly the set of edges in the graph (connection among nodes) varies. The degree of connectivity in the network is determined by the parameter p. A mesh network is formed when $p = 1$, while $p = 0.1$ leads to a sparse network. This type of graph enables a controlled and systematic modification of node numbers, connection patterns, and node interaction frequency and duration [4].

Figure 7.1 shows the mean total computing and communication costs at convergence for OGL and the baselines in different network configurations utilizing the MNIST dataset. Figure 7.2 illustrates the mean total amount of computing and communication costs at convergence for OGL as well as for the baselines using the CIFAR-10 dataset. These results suggest that our OGL scheme is by far the most energy-efficient among the gossip learning schemes, particularly concerning communication costs, achieving a level of efficiency comparable to that of Federated

Learning. This confirms that context-aware tuning of the local training and merging phases of GL schemes may have a high impact on the efficiency and effectiveness of the training process. Another key aspect resulting from our experiments is the relative mean training performance of our OGL scheme in terms of model accuracy at convergence. As Table 7.2 shows, using MNIST dataset OGL outperforms all baseline distributed approaches in all performance metrics, achieving performance comparable to centralized training. Critically, though being significantly more energy efficient, at convergence, OGL improves by more than 40% both mean accuracy and mean loss with respect to DP, i.e. to the best performing gossip-learning approach in the state-of-the-art. Indeed, the two other distributed learning models, DP and random GL, as well as the local-only approach, fail to achieve the target mean accuracy. Figure 7.3 depicts the mean accuracy versus round (learning process) of the OGL and other baseline algorithms using the CIFAR-10 dataset. The outcomes derived from the CIFAR-10 dataset largely mirror those obtained from the MNIST dataset. It reinforces the effectiveness and consistency of the OGL algorithm across different datasets.

Figure 7.4, and 7.5 show the impact of network connectivity and the number of nodes in the system on the evolution of mean accuracy over the learning round for MNIST and CIFAR-10 datasets. In a system with very few nodes, the impact of optimally choosing the neighbours' contributions is relatively modest, with the mean accuracy of Random GL and DP eventually matching that of OGL. In larger systems, our OGL tuning approach is key to achieving faster convergence and higher accuracy in sparse and dense networks. Figure 7.6 illustrates that OGL maintains superior energy efficiency, notwithstanding an accuracy comparable to Random GL. Note that all schemes perform sensibly worse in the CIFAR-10 dataset, as for the same average local dataset size, its samples are more complex (i.e. larger pictures with more pixels).

To evaluate the effectiveness of our OGL approach in a realistic scenario, we consider a scenario where moving nodes are vehicles traversing a region of interest. We focused on a specific area in the city centre of Luxembourg City. This area, a square with sides measuring 1 km, was observed during a low-traffic period (off-peak) from 12:00 PM to 12:40 PM. During this time interval, there are 492 vehicles in the region, with an average sojourn time of 2.9 minutes. On average, there are about 27.3 vehicles in the region at any given time. In this scenario, vehicles are in contact if they are within each other transmission radius. The transmission radius

has been set to 150 m (e.g. typical of DSRC in urban environments [1]). In this case, on average, each vehicle is in contact with 6.7 vehicles at any time. Note that, unlike previous scenarios, the set of nodes in the region may change at different time intervals. Figure 7.7 depicts the percentage of arrivals and departures at every 5-minute interval. To ensure a dynamic neighbourhood pool and give each vehicle enough time to train its local model, vehicles interchange both the loss values and the trained models at regular intervals of twenty seconds. Figure 7.8 shows the mean accuracy of the vehicles approach the target accuracy, obtained by training a centralized ML model on the union of all vehicles' datasets, after ten minutes despite having churn in the network. In addition, we observe the adaptive learning capability of new arrivals. New arrivals are characterized by their initial impact on reducing the mean accuracy, as they are identified as outliers within the given time shown in Figure 7.8. Despite the initial disruption, these outliers demonstrate a capacity to learn from the existing vehicles, thereby gradually aligning with the overall trend. This is evidenced by the subsequent decrease in the number of outliers from time interval 15-20 to 20-25 minutes. This adaptive learning capability of new arrivals contributes to the robustness and resilience of the system, enabling it to maintain overall accuracy over time. It indicates our OGL model is able to maintain high accuracy even in the face of network instability.

7.5 Conclusions

This work presents a novel approach to an energy-efficient gossip learning scheme for dynamic settings. We employ an auxiliary DNN model trained by an orchestrator to adaptively tune some of the key parameters of the learning process in a decentralized manner. Results indicate that our approach efficiently achieves accuracy comparable with a centralized ML method across various network conditions, utilizing time-varying random graphs and a measurement-based dynamic urban scenario across two distinct datasets.

For future work, we plan to enhance the scalability and adaptability of our optimization system by developing a fully distributed optimization system, eliminating the need for an orchestrator, where nodes can self-optimize in response to drastic environmental changes. Furthermore, we plan to investigate the impact of different types of DNN models on the optimization and learning process, as DNN models vary in their computational requirements. This could be particularly important in

a distributed learning context where computational resources are limited.

Bibliography

[1] Khadige Abboud, Hassan Aboubakr Omar, and Weihua Zhuang. Interworking of DSRC and cellular network technologies for V2X communications: A survey. *IEEE transactions on vehicular technology*, 65(12):9457–9470, 2016.

[2] Ahmed M. Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A. Fahmy. REFL: Resource-efficient federated learning. In *Proceedings of the Eighteenth European Conference on Computer Systems*. ACM, may 2023.

[3] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[4] Antonio Di Maio, Mina Aghaei Dinani, and Gianluca Rizzo. The upsides of turbulence: Baselining gossip learning in dynamic settings. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '23, page 376–381, New York, NY, USA, 2023. ACM.

[5] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. Vehicle position nowcasting with gossip learning. In *2022 IEEE WCNC*, pages 728–733, 2022.

[6] Mina Aghaei Dinani, Adrian Holzer, Hung Nguyen, Marco Ajmone Marsan, and Gianluca Rizzo. A gossip learning approach to urban trajectory nowcasting for anticipatory ran management. *IEEE Transactions on Mobile Computing*, pages 1–17, 2023.

[7] Dinani, Mina Aghaei and Holzer, Adrian and Nguyen, Hung and Marsan, Marco

Ajmone and Rizzo, Gianluca. Gossip learning of personalized models for vehicle trajectory prediction. In *2021 IEEE WCNC*, pages 1–7, 2021.

[8] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *sensors*, 12(9):11734–11753, 2012.

[9] Ahmed Imteaj and M. Hadi Amini. Fedar: Activity and resource-aware federated learning model for distributed mobile robots, 2021.

[10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *Learning Multiple Layers of Features from Tiny Images*, 2009.

[11] Rudolf Kruse, Christian Borgelt, Frank Klawonn, Christian Moewes, Matthias Steinbrecher, and Pascal Held. *Multi-Layer Perceptrons*, pages 47–81. Springer London, London, 2013.

[12] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[13] Francesco Malandrino and Carla Fabiana Chiasserini. Federated learning at the network edge: When not all nodes are created equal, 2021.

[14] Othmane Marfoq, Giovanni Neglia, Laetitia Kameni, and Richard Vidal. Personalized federated learning through local memorization, 2022.

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[16] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *Communications Surveys & Tutorials, IEEE*, PP(99):1–18, 2014.

[17] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[18] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, 2009.

[19] Xiaoyong Tang, Jianhua Li, Keqin Li, and Albert Y. Zomaya. Cpu–gpu utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems. *IEEE Access*, 8:58948–58958, 2020.

[20] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems, 2019.

[21] Hongda Wu and Ping Wang. Fast-convergent federated learning with adaptive weighting, 2021.

[22] Yuhao Zhou, Ye Qing, and Jiancheng Lv. Communication-efficient federated learning with compensated overlap-fedavg, 2021.

CHAPTER 8

Conclusion and future work

Conclusion

This thesis advances the state of the art in GL by addressing its deployment under dynamic network conditions and constrained environments. While GL naturally offers scalability and privacy benefits, its practical application in real-world systems, such as vehicular networks and IoT, has remained limited due to challenges such as high mobility, node churn, and resource limitations.

To overcome these barriers, this thesis proposes and evaluates a set of novel GL algorithms and frameworks that address three core challenges: enabling GL in highly dynamic networks, analyzing the impact of network-level dynamics on learning, and ensuring energy-efficient operation in resource-constrained settings.

In Part I, we designed two novel personalized GL algorithms tailored for environments like VANETs, where nodes maintain local models and integrate peer updates through context-aware merging strategies, presented in Chapters 2 and 3. The first algorithm, based on training delegation and marginal utility, improved accuracy but struggled under sparse connectivity. To address this, the second algorithm introduced an opportunistic exchange strategy with location-based merging, which proved effective in enabling robust learning despite mobility, data heterogeneity, and limited peer contact.

In Part II, we introduced a principled modeling framework based on TVGs to evaluate GL under dynamic network conditions beyond specific mobility traces, detailed in Chapters 4 and 5. We first introduced a baseline model for performance under connection rewiring, then expanded it with three new TVG models that incorporate edge persistence and node churn. This allowed us to isolate the effects

of network structure and volatility on learning performance. Our findings revealed a critical exploration–exploitation trade-off, with preferential attachment topologies offering greater robustness to churn.

Finally, in Part III, we addressed the challenge of energy efficiency in decentralized learning. As a supporting effort, we extended prior work on hybrid content delivery in vehicular networks by integrating a learning-assisted mechanism that balances D2D communication with cloud fallback under delay constraints (Chapter 6). Next, we developed OGL in Chapter 7, a decentralized and adaptive framework where each node adjusts its local learning behavior using a pre-trained DNN. Evaluations across synthetic TVGs and vehicular traces demonstrated that OGL significantly reduces energy consumption while maintaining accuracy, enabling practical deployment in constrained edge networks.

Together, these contributions establish a foundation for applying GL in dynamic, resource-constrained systems.

8.1 Future work

While this thesis addresses core feasibility challenges for GL, several promising research directions remain:

- **Hybrid Merging for Context-Aware Trajectory Prediction:** Our current approach delivers robust, low-latency trajectory prediction in vehicular networks. However, the limited scope of local data may constrain its accuracy. A promising direction is to explore hybrid merging strategies that combine decentralized learning with selectively integrated, non-sensitive global data sources (e.g., weather or road conditions). This would allow nodes to balance localized model updates with broader contextual awareness. Developing mechanisms to fuse these heterogeneous data streams while maintaining privacy and minimizing communication overhead remains an open challenge.
- **Multi-Criteria and Attention-Based Model Merging:** Another direction for improving merging strategies lies in moving beyond static, single-criterion approaches. Future work could design adaptive strategies that dynamically combine multiple signals, such as training data volume, distributional similarity, and model reliability, when weighting peer updates. Incorpo-

rating attention-based mechanisms could further enhance responsiveness, allowing nodes to prioritize models most relevant to their own context. The key challenges in this direction involve designing mechanisms that are lightweight, robust to noise, and capable of generalizing across highly heterogeneous and non-IID data environments, without introducing prohibitive computational or communication overhead.

- **Fully Decentralizing Energy-Aware Control:** In our current framework, energy-efficient adaptation relies on a centrally trained DNN-based controller. A critical next step could be to develop fully decentralized, self-adaptive learning strategies—e.g., using reinforcement learning—enabling each node to independently learn optimal policies based on its local energy profile, network conditions, and performance targets. However, realizing this vision requires overcoming factors such as limited on-device computation, sparse local observations, and the difficulty of ensuring stable convergence without centralized oversight in constantly changing network conditions.

- **Expanding to New Application Domains:** While this thesis focused on vehicular networks, our algorithms and models are applicable to other domains with dynamic topology and decentralized structure. Future work could investigate the applicability of our findings to swarm robotics, drone fleets, or wearable health-monitoring systems, where mobility, energy constraints, and privacy are also critical.

- **Exploring Real-World Deployment and Validation:** A key next step is moving from simulation to real-world deployment. This includes deploying the system on wireless edge devices or vehicular testbeds to assess performance under real-time constraints, hardware variability, and communication stack limitations. The challenge lies in achieving robust system-level performance while addressing practical issues like fault tolerance, message loss, and integration with communication stacks and hardware constraints.