Distributed Learning and Adaptive Algorithms for Edge Networks

by

Mehmet Dedeoglu

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved September 2021 by the
Graduate Supervisory Committee:

Junshan Zhang, Chair
Deliang Fan
Oliver Kosut
Yanchao Zhang

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Edge networks pose unique challenges for machine learning and network management. The primary objective of this dissertation is to study deep learning and adaptive control aspects of edge networks and to address some of the unique challenges therein. This dissertation explores four particular problems of interest at the intersection of edge intelligence, deep learning and network management.

The first problem explores the learning of generative models in edge learning setting. Since the learning tasks in similar environments share model similarity, it is plausible to leverage pre-trained generative models from other edge nodes. Appealing to optimal transport theory tailored towards Wasserstein-1 generative adversarial networks, this part aims to develop a framework which systematically optimizes the generative model learning performance using local data at the edge node while exploiting the adaptive coalescence of pre-trained generative models from other nodes.

In the second part, a many-to-one wireless architecture for federated learning at the network edge, where multiple edge devices collaboratively train a model using local data, is considered. The unreliable nature of wireless connectivity, together with the constraints in computing resources at edge devices, dictates that the local updates at edge devices should be carefully crafted and compressed to match the wireless communication resources available and should work in concert with the receiver. Therefore, a Stochastic Gradient Descent based bandlimited coordinate descent algorithm is designed for such settings.

The third part explores the adaptive traffic engineering algorithms in a dynamic network environment. The ages of traffic measurements exhibit significant variation due to asynchronization and random communication delays between routers and controllers. Inspired by the software defined networking architecture, a controller-assisted distributed routing scheme with recursive link weight reconfigurations, accounting for the impact of measurement ages and routing instability, is devised.

The final part focuses on developing a federated learning based framework for traffic reshaping of electric vehicle (EV) charging. The absence of private EV owner information and scattered EV charging data among charging stations motivates the utilization of a federated learning approach. Federated learning algorithms are devised to minimize peak EV charging demand both spatially and temporarily, while maximizing the charging station profit.

To my father and my grandfather who never saw this adventure...

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to my advisor, Professor Junshan Zhang, for his constant support and encouragement. His sincere enthusiasm, strong dedication, and serious attitude towards research always motivated me and drive me to do the best of mine. Moreover, he is always ready to provide help to me in both work and life. He has been an excellent mentor for me during my PhD study. I am very fortunate to have him as my advisor.

I am very grateful to Professor Oliver Kosut, Professor Yanchao Zhang, Professor Deliang Fan, Professor Lei Ying and Prof. Lalitha Sankar for serving as my committee members and for their helpful suggestions. I would like to sincerely thank them for their time and their constant support.

Finally, I would like to thank my mother Aysen Dedeoglu and my father Metin Dedeoglu for their endless love and support, without whom I cannot have come this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview

The recent advancements in edge networks, e.g., content delivery networks and femtocell networks, provoked the exponential growth of the network traffic. To manage the negative impacts of this growth, internet service providers (ISPs) has to implement denser coverage by deploying more small cells at the expense of operational and capital costs. Recent projections on the number of edge devices indicate that the total number may reach 125 billion by 2030 ([70]), which may lead to further revenue losses. Therefore, recently there have been growing concerns towards the efficiency of cloud based schemes. Consequently, there has been a paradigm shift towards edge networks and edge computing, because of their benefits in mitigating the ever-growing traffic and because of the increasing availability of the local computing resources in edge devices. A key advantage of edge networks is their integrity with the cloud-based systems as well as their individual operation ability, which qualifies edge networks as a robust framework.

However, the edge networks have its unique challenges along with its prospects. The data on the edge is often distributed across edge devices. Therefore, the accuracy of the edge learning and network management algorithms heavily hinges upon the collection of these data or the expensive local processing of the data. These approaches either increase the communication cost or require expensive computational resources. To this end, contemporary edge networking and learning algorithms must optimize the trade-off between the communication and computational costs by efficiently utilizing the local computational resources to mitigate the amount of traffic between edge devices. This technical dissertation studies and develops various machine learning and traffic engineering algorithms for the edge networks [39, 40, 168, 167, 89, 166, 37].

1.1.1 Deep Learning for Edge Networks

In accordance with the reports forecasting the hundreds of billions of edge devices [70, 69], the number of learning tasks is going to grow during the next decade. To match this unprecedented growth, both the network capacity and computational resources in the cloud networks must be enhanced. Edge intelligence envisions the efficient utilization of both the data and computational resources on edge networks to mitigate the burden on the cloud, which arises due to the overwhelming number of learning tasks to be completed under tight timing constraints. To this end, the efficient data/model transfer and distributed computation techniques are need to be engineered, which pose unique challenges associated with edge networks.

One particular problem arises in training generative adversarial networks (GANs) in the absence of enough number of data samples. On the grounds that the cloud has abundant computing resources, the conventional method for AI at the edge is that the cloud trains the AI models with the data uploaded from edge devices, and then pushes the models back to the edge for on-device inference (e.g., Google Edge TPU). However, an emerging view is that this approach suffers from overwhelming communication overhead incurred by the data transmission from the edge devices to the cloud, as well as potential privacy leakage. GANs have proven that they can address the problem related to the communication costs and privacy leakage. Recent studies further illustrated that GANs are an effective way of transferring data samples for continual learning as well.

To address these fundamental challenges, this dissertation discusses a new framework, in which the edge network fuses a single GAN model by making use of the previously trained local GAN models and maintains it. Hence, the edge network benefits from less traffic, since a GAN model is much smaller in size than the local datasets at edge devices. Subsequently, new edge devices can utilize this model to quickly train a new local GAN model of high quality by only using small amounts of data samples available to them. To enable the process of incorporating the edge network knowledge and local data samples, a novel adaptive deep learning technique is developed in this dissertation [39].

A second well-established approach to mitigating the excessive traffic in edge learning applications is to locally perform computations up to the capability of edge devices rather than directly transmitting data samples to the cloud. To this end, locally computing gradients and only transmitting the important dimensions of gradients can substantially mitigate communication cost in exchange of the acceptable usage of local computational resources. In many edge networks, mobile and IoT devices collecting huge amounts of data are often connected to each other or a central node wirelessly. The unreliable nature of wireless connectivity, together with constraints in computing resources at edge devices, puts forth a significant challenge for the computation, communication and coordination required to learn an accurate model at the network edge [166].

1.1.2 Adaptive Algorithms for Edge Networks

With the explosive growth of video streaming services and the Internet of Things (IoT) devices at the network edge, the data traffic between cloud services and edge devices thrived to record highs in both directions. According to a recent report [31], the amount of IP traffic has enormously increased over the last two decades and is expected to grow threefold in the next 5 years. In order to sustain the current quality of service requirements of their users and to prevent the deterioration of network robustness, ISPs will need to invest more on their network infrastructures by deploying more routers, base stations and small cells in the near future. Alternatively, less expensive approaches, including cellular traffic offloading onto Device-to-Device (D2D) communications [15, 41, 66] and traffic engineering techniques [53, 146, 19, 160, 152], have been proposed in the literature to sustain the growth in the IP traffic. The techniques deploying D2D provide additional bandwidth and capacity for the traffic growth, but the extra capacity is limited by the amount of short distance communications. On the other hand, the traffic engineering based techniques grant high flexibility by making the use of the network resources to control the link load utilization, but are complex to implement. In particular, the better utilization of the network resources could provide the considerable share of the required capacity for the staggeringly growing IP traffic.

Traffic engineering techniques have been developed for efficient use of network resources in both conventional OSPF-based networks and software-defined networks (SDNs) operating under tight quality of service constraints [53, 19, 152, 160]. Traffic engineering policies can be determined based on the traffic measurements collected throughout the network by a centralized controller. Unfortunately, out-of-date traffic measurements and rapid fluctuations in traffic demand constitute an inaccurate global view of the network at the controller, diminishing the effectiveness of traffic engineering techniques. For better utilization of network resources, contemporary networks need to quickly adapt to changing network environment and satisfy tight timing constraints. To address the above problems, this technical dissertation studies adaptive traffic engineering on edge and backbone networks while taking into consideration the threat of denial of service (DoS) attacks. Accordingly, a local information-based, yet controller-aided, adaptive traffic routing algorithm is engineered [37].

1.2 Summary of Main Contributions

The main body of this dissertation is organized into four chapters. Chapter 2 studies the adaptive coalescence of generative models over edge networks by utilizing the powerful Optimal Transport (OT) theory. A systematic framework to enable fast edge learning of generative models via the adaptive coalescence of pre-trained generative models from other edge nodes and local samples at a new node is proposed. In particular, by treating the knowledge transferred from each node as a Wasserstein ball centered around its local pre-trained generative model, the problem is cast as a constrained optimization problem, which optimizes the edge learning of generative models. Further, a two-stage approach is proposed to efficiently solve the constrained optimization problem, i.e., barycenter problem, where a barycenter for the K pre-trained generative models is found recursively offline in the cloud. Subsequently, the barycenter between the empirical distribution at the target edge node and the coalesced generative model is computed via fast adaptation. The algorithm is supported with analytical results and it is shown that the final barycenter is indeed an interpolation of the given two generative models. Finally, the experimental results illustrate the efficacy of the proposed recursive algorithm for edge networks.

Chapter 3 studies the impact of wireless medium and bandlimited communications on learning a deep neural network model across an edge network. An integrated learning and communication scheme is considered where multiple edge devices send their local gradient updates over multi-carrier communications to the receiver for learning. Bandlimited communication imposes sparse communication of gradient information between edge devices and edge server, facilitating a bias on true gradient information. The devices are subject to power constraints, giving rise to a key question on how to allocate transmission power across dimension, at each edge device, based on the gradient update values and channel conditions. Thus, joint optimization of the power allocation and the learning rate is explored to obtain the best estimate of the gradient updates and minimize the impact of the communication error. A centralized solution to this problem is investigated as a benchmark, and then a sub-optimal distributed solutions amenable to practical implementation is devised. Further, the impact of synchronization errors across devices in this setting is studied.

Chapter 4 constructs a framework for efficient optimization electric vehicle (EV) charging pricing. In the absence of EV owner information, charging stations must adopt a learning based approach to maximize their profit while taking into consideration the charging queues during peak charging hours. Optimal EV charging prices should minimize the waiting duration for charging and also maximize the profit of charging stations. To this end, charging stations train a deep neural network model to predict EV demand for charging at every charging station at any time. Subsequently, the optimal prices are numerically computed by using the trained neural network model. Through developed framework, peak demand for EV charging is spread both spatially and temporarily for improved quality of service via monetary incentives. Consequently, EV owners benefit from decreased charging duration and charging stations gains additional profit from increased quality of service.

Chapter 5 proposes an adaptive algorithm for routing IP traffic amenable to edge networks. In consideration of timing constraints and robustness, chapter 4 adopts centralized load-sensitive routing, and employs a flexible controller to assist distributed routing for improving network utilization. The employed controller iteratively updates OSPF weights and deploys these weights on legacy routers. The legacy routers then determine their own forwarding tables using the OSPF weight set announced by the controller and forward traffic accordingly. In consideration of tight timing constraints, the controller employs a lightweight, load-sensitive OSPF weight update algorithm, which cannot minimize, but only mitigates network congestion. Consequently, network benefits from robustness and flexibility at the cost of optimality, while satisfying tight timing constraints.

Finally, Chapter 6 concludes the dissertation.

Chapter 2

CONTINUAL LEARNING OF GENERATIVE MODELS WITH LIMITED DATA:FROM WASSERSTEIN-1 BARYCENTER TO ADAPTIVE COALESCENCE

2.1 Introduction

The past few years have witnessed an explosive growth of Artificial Intelligence of Things (AIoT) devices at the network edge. On the grounds that the cloud has abundant computing resources, the conventional method for AI at the network edge is that the cloud trains the AI models with the data uploaded from edge devices, and then pushes the models back to the edge for on-device inference (e.g., Google Edge TPU). However, an emerging view is that this approach suffers from overwhelming communication overhead incurred by the data transmission from edge devices to the cloud, as well as potential privacy leakage. It is therefore of great interest to obtain generative models for the edge data, because they require a smaller number of parameters than the data volume and it is much more parsimonious compared to sending the edge data to the cloud, and further they can also help to preserve data privacy. Clearly, continual learning fits naturally in edge applications. Taking a forward-looking view, this chapter focuses on continual learning of generative models for edge intelligence.

There are a variety of edge devices and edge servers, ranging from self-driving cars to robots, from 5G base station servers to mobile phones. Many edge AI applications (e.g., autonomous driving, smart robots, safety-critical health applications, and augmented/virtual reality) require edge intelligence and continual learning capability via fast adaptation with local data samples to adapt to dynamic application environments. Although deep generative models can parametrize high dimensional data samples at edge nodes effectively, it is often not feasible for a single edge server to train a deep generative model from scratch, which would otherwise require humongous training data and high computational power [159, 147, 141].

A general consensus is that learning tasks across different edge nodes often share some model similarity. For instance, different robots may perform similar coordination behaviors according to the environment changes. With this sight, we advocate that the pre-trained generative models from other edge nodes are utilized to speed up the learning at a given edge node, and seek to answer the following critical questions: (1) *"What is the right abstraction of knowledge from multiple pre-trained models for continual learning?"* and (2) *"How can an edge server leverage this knowledge for continual learning of a generative model?"*

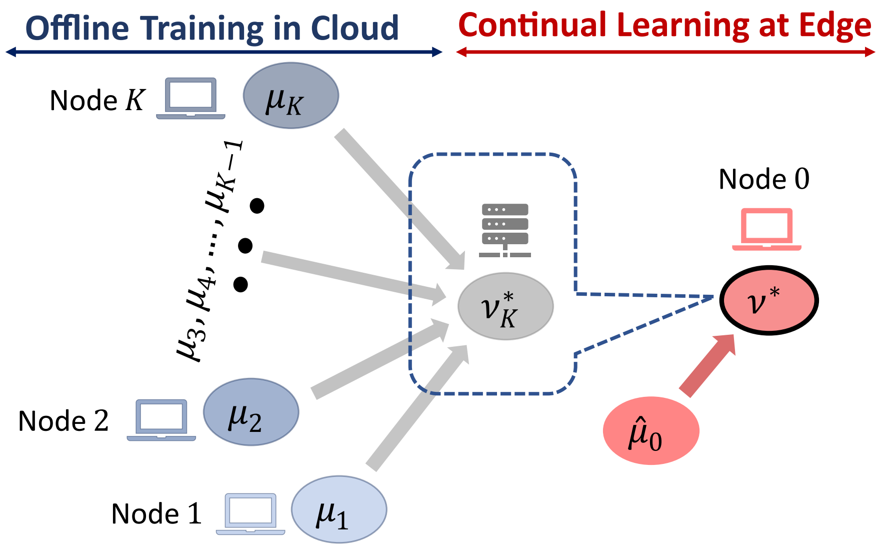
The key to answering the first question lies in efficient model fusion of multiple pre-trained generative models. A common approach is the *ensemble method* [23, 109] where the outputs of different models are aggregated to improve the prediction performance. However, this requires the edge server to maintain all the pre-trained models and run each of them, which would outweigh the resources available at edge servers. Another way for model fusion is direct *weight averaging* [121, 86]. Because the weights in neural networks are highly redundant and no one-to-one correspondence exists between the weights of two different neural networks, this method is known to yield poor performance even if the networks represent the same function of the input. As for the second question, Transfer Learning is a promising learning paradigm where an edge node incorporates the knowledge from the cloud or another node with its local training samples. [147, 141, 159]. Recent work on Transferring GANs [147] proposed several transfer configurations to leverage pre-trained GANs to accelerate the learning process. However, since the transferred GAN is used only as initialization, Transferring GANs suffers from catastrophic forgetting.

To tackle these challenges, this work aims to develop a framework which explicitly optimizes the continual learning of generative models for the edge, based on the adaptive coalescence of pre-trained generative models from other edge nodes, using optimal transport theory tailored towards GANs. To mitigate the mode collapse problem due to the vanishing gradients, multiple GAN configurations have been proposed based on the Wasserstein-p metric W_p, including Wasserstein-1 distance [16] and Wasserstein-2 distance [87, 91]. Despite Wasserstein-2 GANs are analytically tractable, the corresponding implementation often requires regularization and is often outperformed by the Wasserstein-1 GAN (W1GAN). With this insight, in this chapter we focus on the W1GAN (WGAN refers to W1GAN throughout).

2.1.1 Basic Setting

Specifically, we consider a setting where an edge node, denoted Node 0, aims to learn a generative model. It has been shown that training a WGAN is intimately related to finding a distribution minimizing the Wasserstein distance from the underlying distribution μ_0 [17]. In practice, an edge node has a limited number of samples with empirical distribution $\hat{\mu}_0$, which is distant from μ_0. A naive approach is to train a WGAN based on the limited local samples only, which can be captured via the optimization problem given by $\min_{\nu \in \mathcal{P}} W_1(\nu, \hat{\mu}_0)$, with $W_1(\cdot, \cdot)$ being the Wasserstein-1 distance between two distributions. The best possible outcome of solving this optimization problem can generate a distribution very close to $\hat{\mu}_0$, which however could still be far away from the true distribution μ_0. Clearly, training a WGAN simply based on the limited local samples at an edge node would not work well.

As alluded to earlier, learning tasks across different edge nodes may share model



Offline Training in Cloud
Continual Learning at Edge
Node K
μ_K
$\mu_3, \mu_4, \ldots, \mu_{K-1}$
Node 0
ν_K^*
ν^*
Node 2
μ_2
$\hat{\mu}_0$
Node 1
μ_1

Figure 2.1: Continual Learning of Generative Models Based on Coalescence of Pre-trained Generative Models $\{\mu_k, k = 1, \ldots, K\}$ and Local Dataset at Edge Node 0 (Denoted by $\hat{\mu}_0$).

similarity. To facilitate the continual learning at Node 0, pre-trained generative models from other related edge nodes can be leveraged via knowledge transfer. Without loss of generality, we assume that there are a set $\mathcal{K}$ of K edge nodes with pre-trained generative models. Since one of the most appealing benefits of WGANs is the ability to continuously estimate the Wasserstein distance during training [16], we assume that the knowledge transfer from Node k to Node 0 is in the form of a Wasserstein ball with radius η_k centered around its pre-trained generative model μ_k at Node k, for $k = 1, \ldots, K$. Intuitively, radius η_k represents the relevance (hence utility) of the knowledge transfer, and the smaller it is, the more informative the corresponding Wasserstein ball is. Building on this knowledge transfer model, we treat the continual learning problem at Node 0 as the coalescence of K generative models and empirical distribution $\hat{\mu}_0$ (Figure 2.1), and cast it as the constrained optimization problem:

$$\min_{\nu \in \mathcal{P}} W_1(\nu, \hat{\mu}_0), \text{ s.t. } W_1(\nu, \mu_k) \leq \eta_k, \forall k \in \mathcal{K}. \tag{2.1}$$

Observe that the constraints in problem (2.1) dictate that the optimal coalesced generative model, denoted by ν^*, lies within the intersection of K Wasserstein balls (centered around $\{\mu_k\}$), exploiting the knowledge transfer systematically. It is worth noting that the optimization problem (2.1) can be extended to other distance functionals, e.g., Jensen-Shannon divergence.

2.1.2 Main Contributions

The contributions of this work are summarized as follows.

i) We propose a systematic framework to enable continual learning of generative models via adaptive coalescence of pre-trained generative models from other edge nodes and local samples at Node 0. In particular, by treating the knowledge transferred from each node as a Wasserstein ball centered around its local pre-trained generative model, we cast the problem as a constrained optimization problem which optimizes the continual learning of generative models.

ii) Applying Lagrangian relaxation to (2.1), we reduce the optimization problem to finding a Wasserstein-1 barycenter of $K+1$ probability measures, among which K of them are pre-trained generative models and the last one is the empirical distribution (not a generative model though) corresponding to local data samples at Node 0. We propose a *barycentric fast adaptation approach* to efficiently solve the barycenter problem, where the barycenter ν_K^* for the K pre-trained generative models is found recursively offline in the cloud, and then the barycenter between the empirical distribution $\hat{\mu}_0$ of Node 0 and ν_K^* is solved via fast adaptation at Node 0. A salient feature in this barycentric approach is that generative replay, enabled by pre-trained GANs, is used to annihilate catastrophic forgetting.

iii) It is known that the Wasserstein-1 barycenter is notoriously difficult to analyze, partly because of the existence of infinitely many minimizers of the Monge Problem. Appealing to optimal transport theory, we use displacement interpolation as the theoretic foundation to devise recursive algorithms for finding adaptive barycenters, which ensures the resulting barycenters lie in the baryregion.

iv) From the implementation perspective, we introduce a "recursive" WGAN configuration, where a 2-discriminator WGAN is used per recursive step to find adaptive barycenters sequentially. Then the resulting barycenter in offline training is treated as the meta-model initialization and fast adaptation is carried out to find the generative model using the local samples at Node 0. A weight ternarization method, based on joint optimization of weights and threshold for quantization, is developed to compress the generative model and enable efficient edge learning. Extensive experiments corroborate the efficacy of the proposed framework for fast edge learning of generative models. Further, our experimental studies corroborate that W_1-based recursive WGAN configuration performs better than the W_2^2-based one.

2.2 Related Work

Optimal transport theory has recently been studied for deep learning applications (see, e.g., [24, 10, 133]). [2] has developed an analytical solution to the Wasserstein barycenter problem. Aiming to numerically solve the Wasserstein barycenter problem, [32, 33, 34] proposed smoothing through entropy regularization for the discrete setting, based on linear programming. [122] employed posterior sampling algorithms in studying Wasserstein barycenters, and [14] characterized Wasserstein barycenters for the discrete setting (cf. [123, 157, 119]). It is worth mentioning that [119] presented a layer-wise model fusion algorithm for DNNs that utilizes optimal transport to align neurons in this discrete setting; in contrast, our proposed framework enables adaptive coalescence of pre-trained generative models for both continuous and discrete cases.

GANs [55] have recently emerged as a powerful deep learning tool for obtaining generative models. [16] has introduced Wasserstein metric in GANs, which can help mitigate the vanishing gradient issue to avoid mode collapse. Though gradient clipping is applied to ensure 1-Lipschitz conditions, it may still lead to non-convergence. [59] proposed to use gradient penalty to overcome the shortcomings due to weight clipping. Using optimal transport theory, recent advances of Wasserstein GANs have shed light on understanding generative models. Recent works [87, 91, 79] proposed multiple transport theory based GAN configurations using quadratic Wasserstein-2 cost. Furthermore, [85] devised a computationally efficient method for computing the generator when the cost function is convex. In contrast, for the Wasserstein-1 GAN, the discriminator may utilize one of infinitely many transport maps from underlying empirical data distribution to the generative model [10, 133], and it remains open to decipher the relation between the model training and the transport maps. Along a different line, a variety of techniques have been proposed for more robust training of GANs [101, 159, 46, 118]. [46] proposed a multi-discriminator GAN configuration for robust training of GANs. Since samples from the same underlying distribution are fed to every discriminator, this technique does not attempt to find a barycenter. [159] extends [46] by proposing a method to train a generator leveraging pre-trained discriminators on distinct empirical distributions, which resembles the ensemble technique since each discriminator is trained on a single mode of the dataset and the generator directly learns an ensemble distribution by using these discriminators. Herein, we develop a WGAN configuration for computing Wasserstein-1 barycenter models, and the proposed framework is capable of learning novel generative models in addition to ensemble distributions.

Pushing the AI frontier to the network edge for achieving edge intelligence has recently emerged as the marriage of AI and edge computing [170]. There are significant challenges since AI model training generally requires tremendous resources that greatly outweigh the capability of resource-limited edge nodes. To address this, various approaches have been proposed in the literature, including model compression [113, 156, 139], knowledge transfer learning [98, 140], hardware acceleration [131, 142], collaboration-based methods [90, 169]. Different from these existing studies, *this work focuses on continual learning of generative models at the edge node.* Rather than learning the new model from scratch, continual learning aims to design algorithms leveraging knowledge transfer from pre-trained models to the new learning task [128], assuming that the training data of previous tasks are unavailable for the newly coming task. Generative replay is gaining more attention where synthetic samples corresponding to earlier tasks are obtained with a generative model and replayed in model training for the new task to mitigate forgetting [108, 104]. In this work, by learning generative models via the adaptive coalescence of pre-trained generative models from other nodes, the proposed "recursive" WGAN configuration facilitates fast edge learning in a continual manner, which can be viewed as an innovative integration of a few key ideas in continual learning, including the replay method [116, 150, 99, 107] which generates pseudo-samples using generative models, and the regularization-based methods [77, 83, 110, 44] which sets the regularization for the model learning based on the learned knowledge from previous tasks, in continual learning [35].

2.3 Adaptive Coalescence of Wasserstein-1 Generative Models

In what follows, we first recast problem (2.1) as a variant of the Wasserstein barycenter problem. Then, we propose a two-stage recursive algorithm, characterize the geometric properties of geodesic curves therein and use displacement interpolation as the foundation to devise recursive algorithms for finding adaptive barycenters.

2.3.1 A Wasserstein-1 Barycenter Formulation via Lagrangian Relaxation

Observe that the Lagrangian for (2.1) is given as follows:

$$\mathcal{L}(\{\lambda_k\}, \nu) = W_1(\nu, \hat{\mu}_0) + \sum_{k=1}^{K} \lambda_k W_1(\nu, \mu_k) - \sum_{k=1}^{K} \lambda_k \eta_k, \tag{2.2}$$

where $\{\lambda_k \geq 0\}_{1:K}$ are the Lagrange multipliers. Based on [136], problem (2.1) can be solved by using the following Lagrangian relaxation with $\lambda_k = \frac{1}{\eta_k}, \forall k \in \mathcal{K}$, and $\lambda_0 = 1$:

$$\min_{\nu \in \mathcal{P}} \sum_{k=1}^{K} \frac{1}{\eta_k} W_1(\nu, \mu_k) + W_1(\nu, \hat{\mu}_0). \tag{2.3}$$

It is shown in [120] that the selection $\lambda_k = \frac{1}{\eta_k}, \forall k \in \mathcal{K}$, ensures the same levels of robustness for (2.3) and (2.1). Intuitively, such a selection of $\{\lambda_k\}_{0:K}$ strikes a right balance, in the sense that larger weights are assigned to the knowledge transfer models (based on the pre-trained generative models $\{\mu_k\}$) from the nodes with higher relevance, captured by smaller Wasserstein-1 ball radii. For given $\{\lambda_k \geq 0\}$, (2.3) turns out to be a Wasserstein-1 barycenter problem (cf. [2, 122]), with the new complication that $\hat{\mu}_0$ is an empirical distribution corresponding to local samples at Node 0. Since $\hat{\mu}_0$ is not a generative model per se, its coalescence with other K general models is challenging.

2.3.2 A Two-Stage Adaptive Coalescence Approach for Wasserstein-1 Barycenter Problem

Based on (2.3), we take a two-stage approach to enable efficient learning of the generative model at edge Node 0. The primary objective of Stage I is to find the barycenter for K pre-trained generative models $\{\mu_1, \ldots, \mu_K\}$. Clearly, the ensemble method would not work well due to required memory and computational resources. With this insight, we develop a recursive algorithm for adaptive coalescence of pre-trained generative models. In Stage II, the resulting barycenter solution in Stage I is treated as the model initialization, and is further trained using the local samples at Node 0. We propose that the offline training in Stage I is performed in the cloud, and the fast adaptation in Stage II is carried out at the edge server (in the same spirit as the model update of Google EDGE TPU), as outlined below:

Stage I: Find the barycenter of K pre-trained generative models across K edge nodes offline. Mathematically, this entails the solution of the following problem:

$$\min_{\nu \in \mathcal{P}} \sum_{k=1}^{K} \frac{1}{\eta_k} W_1(\nu, \mu_k). \tag{2.4}$$

To reduce computational complexity, we propose the following recursive algorithm: Take μ_1 as an initial point, i.e., $\nu_1^* = \mu_1$, and let ν_{k-1}^* denote the barycenter of $\{\mu_i\}_{1:k-1}$ obtained at iteration $k-1$ for $k = 2, \ldots, K$. Then, at each iteration k, a new barycenter ν_k^* is solved between the barycenter ν_{k-1}^* and the pre-trained generative model μ_k.

Stage II: Fast adaptation to find the barycenter between ν_K^* and the local dataset at Node 0. Given the solution ν_K^* obtained in Stage I, we subsequently solve the following problem: $\min_{\nu \in \mathcal{P}} \ W_1(\nu, \hat{\mu}_0) + W_1(\nu, \nu_K^*)$. By taking ν_K^* as the model initialization, fast adaptation based on local samples is used to learn the generative model at Node 0.

2.3.3 A Preliminary Review on Optimal Transport Theory

This section provides a brief overview of optimal transport theory, which serves as the theoretic foundation for the proposed two-stage adaptive coalescence algorithm for fast edge learning of generative models. In particular, it is known that the Wasserstein-1 barycenter is difficult to analyze, because of the existence of infinitely many minimizers of the Monge Problem. We will review related geometric properties of geodesic curves therein and introduce displacement interpolation.

Optimal transport theory has been extensively utilized in economics for decades, and has recently garnered much interest in deep learning applications (see, e.g., [24, 10, 133]). Simply put, optimal transport theory aims to find the most efficient transport map from one probability distribution to another with respect to a predefined cost function $c(x, y)$. The optimal distribution preserving the transport map can be obtained by solving the Monge problem.

Definition 1. *(**Monge Problem**) Let $(\mathcal{X}, d)$ and $\mathcal{P}(\mathcal{X})$ denote a complete and separable metric space, i.e., a Polish space, and the set of probability distributions on $\mathcal{X}$, respectively. Given $\mu \in \mathcal{P}(\mathcal{X})$ and $\nu \in \mathcal{P}(\mathcal{Y})$ defined on two Polish spaces which are connected with a Borel map T, the Monge problem is defined as:*

$$\inf_{T:T\#\mu=\nu} \int_{\mathcal{X}} c(x, T(x)) d\mu(x). \tag{2.5}$$

In Definition 1, T is referred as the distribution preserving *transport map* and $\#$ denotes the push-forward operator. In lieu of the strict constraint, there may not exist an optimal transport map for the Monge problem. A relaxation of the Monge problem leads to Kantorovich's optimal transport problem.

Definition 2. *(**Kantorovich Problem**) Given $\mu \in \mathcal{P}(\mathcal{X})$ and $\nu \in \mathcal{P}(\mathcal{Y})$ are two probability distributions defined on two Polish spaces, the Kantorovich problem is defined as:*

$$\inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\gamma(x, y), \tag{2.6}$$

where $\Pi(\mu, \nu)$ is the admissible set with its elements satisfying:

$$\pi_\mu\#\gamma = \mu, \qquad\qquad \pi_\nu\#\gamma = \nu, \tag{2.7}$$

where π_μ and π_ν are two projector transport maps.

In Definition 2, γ is referred as the *transference plan* and the admissible set Π is a relaxation to $T\#\mu = \nu$. A transference plan can leverage *mass splitting* in contrast to transport maps, and hence can result in a solution under the semi-continuity assumptions. Mass splitting further enables the reputed Kantorovich duality, as shown in the following lemma, facilitating an alternative and convenient representation of the Kantorovich problem.

Lemma 1. ***(Kantorovich Duality,*** *[132]) Let $\mu \in \mathcal{P}(\mathcal{X})$ and $\nu \in \mathcal{P}(\mathcal{Y})$ be two probability distributions defined on Polish spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively, and let $c(x, y)$ be a lower semi-continuous cost function. Further, define Φ_c as the set of all measurable functions $(\varphi, \psi) \in L^1(d\mu) \times L^1(d\nu)$ satisfying:*

$$\varphi(x) + \psi(y) \leq c(x, y), \tag{2.8}$$

for $d\mu$-almost all $x \in \mathcal{X}$ and $d\nu$-almost all $y \in \mathcal{Y}$. Then, the following strong duality holds for c-concave function φ:

$$\inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\gamma(x, y) = \sup_{(\varphi,\psi)\in\Phi_c} \int_{\mathcal{X}} \varphi(x) d\mu(x) + \int_{\mathcal{Y}} \psi(y) d\nu(y). \tag{2.9}$$

As the right hand side of (2.9) is an optimization over two functions, efficient gradient algorithms can be employed to learn the optimal solution. (2.9) can be further simplified using c-transform [133], in which $\psi(y)$ can be replaced by the c-transform $\varphi^c(y) = \inf_{x\in\mathcal{X}} c(x, y) - \varphi(x)$, and φ is referred as the Kantorovich potential. The following lemma establishes the existence of a Kantorovich potential that can also represent the Monge problem.

Lemma 2. ***(Existence of Optimal Transport Plan,*** *[9]) For a lower semi-continuous cost function $c(x, y)$ defined on $\mathcal{X} \times \mathcal{Y}$, there exists at least one $\gamma \in \Pi(\mu, \nu)$ solving the Kantorovich problem. Furthermore, if $c(x, y)$ is continuous and* *real-valued, and μ has no atoms, then the minimums to both Monge and Kantorovich problems are equivalent, i.e.,*

$$\inf_{T:T\#\mu=\nu}\int_{\mathcal{X}} c(x,T(x))d\mu(x)=\inf_{\gamma\in\Pi(\mu,\nu)}\int_{\mathcal{X}\times\mathcal{Y}} c(x,y)d\gamma(x,y). \tag{2.10}$$

Lemma 2 indicates that there exists at least one transport map which are solutions to the Kantorovich problem. We here remark that not all transference plans are necessarily transport maps. Lemma 2 further facilitates a connection between dataset interpolation and the proposed Wasserstein GAN configuration in this chapter, along with the McCann's celebrated displacement interpolation result [94].

2.3.4 From Displacement Interpolation to Adaptive Barycenters

As noted above, in practical implementation, the W1-GAN often outperforms Wasserstein-p GANs ($p > 1$). However, the Wasserstein-1 barycenter is notoriously difficult to analyze due to the non-uniqueness of the minimizer to the Monge Problem [133]. Appealing to optimal transport theory, we next characterize the performance of the proposed two-stage recursive algorithm for finding the Wasserstein-1 barycenter of pre-trained generative models $\{\mu_k, k=1,\dots,K\}$ and the local dataset at Node 0, by examining the existence of the barycenter and characterizing its geometric properties based on geodesic curves.

The seminal work [94] has established the existence of geodesic curves between any two distribution functions σ_0 and σ_1 in the p-Wasserstein space, $\mathcal{P}_p$, for $p \geq 1$. It is shown in [133] that there are infinitely many minimal geodesic curves between σ_0 and σ_1, when $p=1$. This is best illustrated in N dimensional Cartesian space, where the minimal geodesic curves between $\varsigma_0 \in \mathbb{R}^N$ and $\varsigma_1 \in \mathbb{R}^N$ can be parametrized as follows: $\varsigma_t = \varsigma_0 + s(t)(\varsigma_1 - \varsigma_0)$, where $s(t)$ is an arbitrary function of t, indicating that there are infinitely many minimal geodesic curves between ς_0 and ς_1. This is in stark contrast to the case $p > 1$ where there is a unique geodesic between ς_0 and ς_1. In a similar fashion, there exists infinitely many transport maps, T_0^1, from σ_0 to σ_1 when $p = 1$. For convenience, let $C(\sigma_0, \sigma_1)$ denote an appropriate transport cost function quantifying the minimum cost to move a unit mass from σ_0 to σ_1. It has been shown in [133] that when $p = 1$, two interpolated distribution functions on two distinct minimal curves may have a non-zero distance, i.e., $C(\hat{T}_0^1 \# \sigma_0, \tilde{T}_0^1 \# \sigma_0) \geq 0$, where $\#$ denotes the push-forward operator, thus yielding multiple minimizers to (2.4). For convenience, define $\mathcal{F} := \hat{\mu}_0 \cup \{\mu_k\}_{1:K}$.

Definition 3. ***(Baryregion)*** *Let $g_t(\mu_k, \mu_\ell)_{0 \leq t \leq 1}$ be any minimal geodesic curve between any pair $\mu_k, \mu_\ell \in \mathcal{F}$, and define the union $\mathcal{R} := \bigcup_{k=1}^{K} \bigcup_{\ell=k+1}^{K+1} g_t(\mu_k, \mu_\ell)_{0 \leq t \leq 1}$. The baryregion $\mathcal{B}_{\mathcal{R}}$ is given by $\mathcal{B}_{\mathcal{R}} = \bigcup_{\sigma \in \mathcal{R}} \bigcup_{\varpi \in \mathcal{R}, \varpi \neq \sigma} g_t(\sigma, \varpi)_{0 \leq t \leq 1}$.*

Intuitively, $\mathcal{B}_{\mathcal{R}}$ encapsulates all possible interpolations through distinct geodesics between any two distributions in $\mathcal{F}$. Since each geodesic has finite length, $\mathcal{B}_{\mathcal{R}}$ defines a bounded set in $\mathcal{P}_1$. Next we restate in Lemma 3 the renowned *Displacement Interpolation* result [94], which sets the foundation for each recursive step in finding a barycenter in our proposed two-stage algorithm. In particular, Lemma 3 leads to the fact that the barycenter ν^* resides in $\mathcal{B}_{\mathcal{R}}$.

Lemma 3. ***(Displacement Interpolation, [132])*** *Let $C(\sigma_0, \sigma_1)$ denote the minimum transport cost between σ_0 and σ_1, and suppose $C(\sigma_0, \sigma_1)$ is finite for $\sigma_0, \sigma_1 \in \mathcal{P}(\mathcal{X})$. Assume that $C(\sigma_s, \sigma_t)$, the minimum transport cost between σ_s and σ_t for any $0 \leq s \leq t \leq 1$, is continuous. Then, the following holds true for any given continuous path $g_t(\sigma_0, \sigma_1)_{0 \leq t \leq 1}$:*

$$C(\sigma_{t_1}, \sigma_{t_2}) + C(\sigma_{t_2}, \sigma_{t_3}) = C(\sigma_{t_1}, \sigma_{t_3}), \ 0 \leq t_1 \leq t_2 \leq t_3 \leq 1.$$

In the adaptive coalescence algorithm, the kth recursion defines a baryregion, $\mathcal{B}_{\{\nu_{k_1}^*, \mu_k\}}$, consisting of geodesics between the barycenter ν_{k-1}^* found in $(k-1)$th recursion and generative model μ_k. Clearly, $\mathcal{B}_{\{\nu_k^*,\mu_k\}} \subset \mathcal{B}_{\mathcal{R}}$. Viewing each recursive step in the above two-stage algorithm as adaptive displacement interpolation, we have the following main result on the geodesics and the geometric properties regarding ν^* and $\{\nu_k^*\}_{1:K}$.

Proposition 1. ***(Displacement Interpolation for Adaptive Barycenters)*** *The adaptive barycenter, ν_k^*, obtained at the output of kth recursive step in Stage I, is a displacement interpolation between ν_{k-1}^* and μ_k and resides inside $\mathcal{B}_{\mathcal{R}}$. Further, the final barycenter ν^* resulting from Stage II of the recursive algorithm resides inside $\mathcal{B}_{\mathcal{R}}$.*

Remark 1. *It is worth pointing out that different orders for adaptive coalescence may lead to different final barycentric W1GAN models, although the resulting ν^* resides in $\mathcal{B}_{\mathcal{R}}$ always. Had the quadratic Wasserstein cost W_2^2 been used, the final barycenter ν^* would be unique in $\mathcal{B}_{\mathcal{R}}$. However, the corresponding implementation using W_2^2 poses significant challenges [91, 79] and is often outperformed by W1GAN, which we will elaborate further in Section 2.5 and in Appendix 2.9.2 and 2.9.3.*

2.4 A Preliminary Review on Wasserstein GANs

2.4.1 From Vanilla Generative Adversarial Networks (GAN) to Wasserstein-1 GAN

A generative adversarial network is comprised of a generator and discriminator neural networks. Random noise samples are fed into the generator to generate data samples of certain structure at the output of the generator. The generated (or fake) samples are then fed into the discriminator along with real-world samples taken from the dataset. The discriminator acts as a classifier and incurs a loss when mislabeling takes place. From a game theoretic point of view, the generator and the discriminator play a zero-sum game, in which the generator seeks to manipulate the discriminator to classify fake samples as real by generating samples similar to the real-world dataset. In principle, GAN training is equivalent to solving for the following optimization problem:

$$\begin{aligned}\min_G \max_D V(D,G) &= \min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mu}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\vartheta}[\log(1-D(G(\mathbf{z})))] \\ &= \min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mu}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{y}\sim\nu}[\log(1-D(\mathbf{y}))], \qquad (2.11)\end{aligned}$$

where D and G represent the discriminator and generator networks, respectively. Let μ, ν and ϑ denote the distributions from empirical data, at generator output and at generator input, respectively. The latent distribution ϑ is often selected to be uniform or Gaussian. The output of the generator, denoted $\mathbf{y} = G(\mathbf{z}, \theta_G) \sim \nu$, is composed by propagating $\mathbf{z}$ through a nonlinear transformation, represented by neural network parameter θ_G. Model parameter θ_G entails ν to reside in a parametric probability distribution space $\mathcal{Q}_G$, constructed by passing ϑ through G. It has been shown in [55] that the solution to (2.11) can be expressed as an optimization problem over ν as:

$$\min_{\nu\in\mathcal{Q}_G} -\log(4) + 2\cdot \mathrm{JSD}(\mu||\nu), \qquad (2.12)$$

where JSD denotes Jensen-Shannon divergence. Clearly, the solution to (2.12) can be achieved at $\nu^* = \mu$, and the corresponding θ_G^* is the optimal generator model parameter.

The vanilla GAN training process suffers from the mode collapse issue that is often caused by vanishing gradients during the training process of GANs [16]. In contrast to JSD, under mild conditions the Wasserstein distance does not incur vanishing gradients, and hence exhibits more useful gradient properties for preventing mode collapse. The training process of Wasserstein-1 distance based GAN can be expressed as solving an optimization problem $\min_{\nu\in\mathcal{Q}_G} W_1(\nu,\mu)$. Since the c-transform of the

Kantorovich potential admits a simpler and more convenient form for W_1, i.e., $\varphi^c = -\varphi$, the Wasserstein-1 GAN cost function can be rewritten as:

$$W_1(\nu, \mu) = \sup_{||\varphi||_L \leq 1} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}[\varphi(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\nu}[\varphi(\mathbf{y})] \right\}, \tag{2.13}$$

where φ is constrained to be a 1-Lipschitz function. Following the same line as in the vanilla GAN, φ in (2.13) can be characterized by a neural network, which is parametrized by model parameters θ_D. Consequently, training a Wasserstein-1 GAN is equivalent to solve the following non-convex optimization problem through training the generator and discriminator neural networks:

$$\min_G \max_{||\varphi||_L \leq 1} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}[\varphi(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim\nu}[\varphi(\mathbf{y})] \right\} = \min_G \max_{||\varphi||_L \leq 1} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}[\varphi(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\vartheta}[\varphi(G(\mathbf{z}))] \right\}. \tag{2.14}$$

We here note that φ must be selected from a family of 1-Lipschitz functions. To this end, various training schemes have been proposed in the literature.

2.4.2 *From Wasserstein-1 Barycenter to Multi-Discriminator GAN Cost*

Problem (2.3) can be expressed in terms of Kantorovich potentials by applying Kantorovich's Duality as:

$$\begin{aligned}\min_{\nu\in\mathcal{P}} \sum_{k=1}^{K} \frac{1}{\eta_k} W_1(\nu, \mu_k) + W_1(\nu, \hat{\mu}_0) = \min_{\nu\in\mathcal{P}} \sum_{k=1}^{K} \frac{1}{\eta_k} \sup_{(\varphi_k, \psi_k)\in\Phi_c} \left\{ \int_{\mathcal{X}} \varphi_k(x) d\mu_k(x) + \int_{\mathcal{Y}} \psi_k(y) d\nu(y) \right\} \\ + \sup_{(\varphi_0, \psi_0)\in\Phi_c} \left\{ \int_{\mathcal{X}} \varphi_0(x) d\hat{\mu}_0(x) + \int_{\mathcal{Y}} \psi_0(y) d\nu(y) \right\}. \end{aligned} \tag{2.15}$$

By using c-transformation, we have $\psi_k(y) = \varphi_k^c(y)$. In particular, for the Wasserstein-1 distance, we have that $\varphi_k^c(y) = -\varphi_k(y)$, and hence (2.15) is further simplified as:

$$\begin{aligned}
\min_{\nu\in\mathcal{P}} \sum_{k=1}^{K} \frac{1}{\eta_k} W_1(\nu,\mu_k) + W_1(\nu,\hat{\mu}_0) &= \min_{\nu\in\mathcal{P}} \ \sum_{k=1}^{K} \frac{1}{\eta_k} \max_{\|\varphi_k\|_L\le 1} \left\{\mathbb{E}_{\mathbf{x}\sim\mu_k}[\varphi_k(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim\nu}[\varphi_k(\mathbf{y})]\right\} \\
&\quad + \max_{\|\varphi_0\|_L\le 1} \left\{\mathbb{E}_{\mathbf{x}\sim\hat{\mu}_0}[\varphi_0(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim\nu}[\varphi_0(\mathbf{y})]\right\} \\
&= \min_{G} \max_{\{\|\varphi_k\|_L\le 1\}_{0:K}} \left\{\mathbb{E}_{\mathbf{x}\sim\hat{\mu}_0}[\varphi_0(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\vartheta}[\varphi_0(G(\mathbf{z}))]\right\} \\
&\quad + \sum_{k=1}^{K} \frac{1}{\eta_k} \left\{\mathbb{E}_{\mathbf{x}\sim\mu_k}[\varphi_k(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\vartheta}[\varphi_k(G(\mathbf{z}))]\right\}. \qquad (2.16)
\end{aligned}$$

Therefore, a barycenter of K distributions can be obtained by minimizing the cost in (2.16) through a specially designed GAN configuration.

2.5 Recursive WGAN Configuration for Continual Learning

Based on the above theoretic results on adaptive coalescence via Wasserstein-1 barycenters, we next turn attention to the implementation of computing adaptive barycenters. Notably, assuming the knowledge of accurate empirical distribution models on discrete support, [33] introduces a powerful linear program (LP) to compute Wasserstein-p barycenters, but the computational complexity of this approach is excessive. In light of this, we propose a WGAN-based configuration for finding the Wasserstein-1 barycenter, which in turn enables fast learning of generative models based on the coalescence of pre-trained models. Specifically, (2.3) can be rewritten as:

$$\min_{G} \max_{\{\varphi_k\}_{0:K}} \mathbb{E}_{x\sim\hat{\mu}_0}[\varphi_0(x)] - \mathbb{E}_{z\sim\vartheta}[\varphi_0(G(z))] + \sum_{k=1}^{K} \frac{1}{\eta_k} \left\{\mathbb{E}_{x\sim\mu_k}[\varphi_k(x)] - \mathbb{E}_{z\sim\vartheta}[\varphi_k(G(z))]\right\}, \qquad (2.17)$$

where G represents the generator and $\{\varphi_k\}_{0:K}$ are $1-$Lipschitz functions for discriminator models, respectively. Observe that the optimal generator DNN G^* facilitates

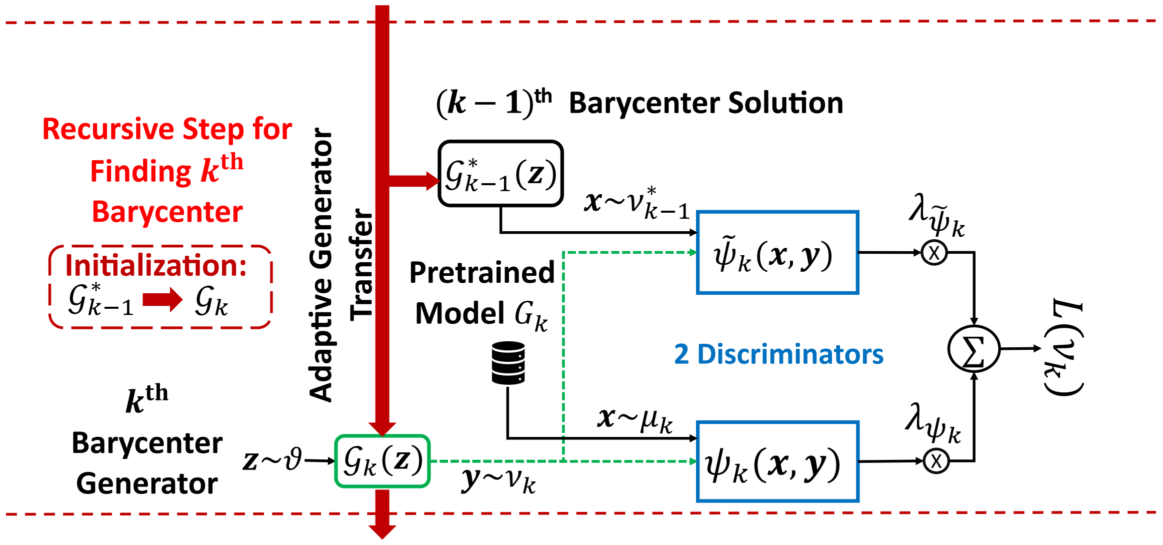

Recursive Step for Finding k^{th} Barycenter
Initialization: $\mathcal{G}_{k-1}^* \Rightarrow \mathcal{G}_k$
Adaptive Generator Transfer
$(k-1)^{\text{th}}$ Barycenter Solution
$\mathcal{G}_{k-1}^*(z)$
$x \sim \nu_{k-1}^*$
$\tilde{\psi}_k(x,y)$
$\lambda_{\tilde{\psi}_k}$
Pretrained Model G_k
2 Discriminators
$L(\nu_k)$
$x \sim \mu_k$
$\psi_k(x,y)$
λ_{ψ_k}
k^{th} Barycenter Generator
$z \sim \vartheta$
$\mathcal{G}_k(z)$
$y \sim \nu_k$

Figure 2.2: A 2-discriminator WGAN for Efficient Learning of the kth Barycenter Generator in Offline Training, Where x Denotes the Synthetic Data Generated from Pretrained Models.

the barycenter distribution ν^* at its output. We note that the multi-discriminator WGAN configuration have recently been developed [46, 62, 97], by using a common latent space to train multiple discriminators so as to improve stability. In stark contrast, in this work distinct generative models from multiple nodes are exploited to train different discriminators, aiming to learn distinct transport plans among generative models.

A naive approach is to implement the above multi-discriminator WGAN in a one-shot manner where the generator and $K+1$ discriminators are trained simultaneously, which however would require overwhelming computation power and memory. To enable efficient training, we use the proposed two-stage algorithm and develop a *recursive* WGAN configuration to sequentially compute 1) the barycenter ν_K^* for the offline training in the cloud, as shown in Figure 2.2; and 2) the barycenter ν^* for the fast adaptation at the target edge node, as shown in Figure 2.3. The analytical relation between *one-shot* and *recursive* barycenters has been studied for Wasserstein-2 distance, and sufficient conditions for their equivalence is presented in [22], which,

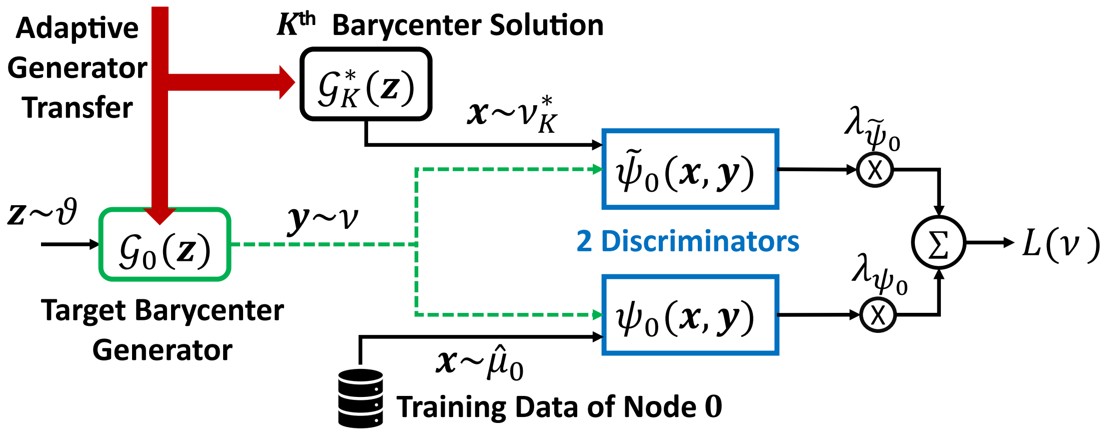


Figure 2.3: Fast Adaptation for Learning a Generative Model at Node 0.

would not suffice for Wasserstein-1 distance, because of the existence of multiple Wasserstein-1 barycenters. Proposition 1 shows that any barycenter solution to recursive algorithm resides inside a baryregion, which can be viewed as the counterpart for the *one-shot* solution. We have also developed the bound on the gap between *one-shot* and *recursive* algorithms, which can be found in Appendix 2.9.3. We next highlight a few important advantages of the "recursive" WGAN configuration for the Barycentric Fast Adaptation algorithm.

2.5.1 A 2-discriminator WGAN Implementation per Recursive Step to Enable Efficient Training

At each recursive step k, we aim to find the barycenter ν_k^* between pre-trained model μ_k and the barycenter ν_{k-1}^* from last round, which is achieved by training a 2-discriminator WGAN as follows:

$$\min_{\mathcal{G}_k} \max_{\psi_k, \tilde{\psi}_k} \lambda_{\psi_k} \{\mathbb{E}_{x\sim\mu_k}[\psi_k(x)] - \mathbb{E}_{z\sim\vartheta}[\psi_k(\mathcal{G}_k(z))]\} \\ + \lambda_{\tilde{\psi}_k} \{\mathbb{E}_{x\sim\nu_{k-1}^*}[\tilde{\psi}_k(x)] - \mathbb{E}_{z\sim\vartheta}[\tilde{\psi}_k(\mathcal{G}_k(z))]\}, \qquad (2.18)$$

Algorithm 1 Offline Training to Solve the Barycenter of K Pre-trained Generative Models

```
1: Inputs: $K$ pre-trained generator-discriminator pairs $\{(G_k, D_k)\}_{1:K}$ of corresponding source
   nodes $k \in \mathcal{K}$, noise prior $\vartheta(z)$, the batch size $m$, learning rate $\alpha$
2: Set $\mathcal{G}_1^* \leftarrow G_1$, $\tilde{\psi}_1^* \leftarrow$ rand() or $\tilde{\psi}_1^* \leftarrow D_1$; //Barycenter initialization
3: for iteration $k = 2, ..., K$ do
4:    Set $\mathcal{G}_k \leftarrow \mathcal{G}_{k-1}^*$, $\tilde{\psi}_k \leftarrow$ rand(), $\psi_k \leftarrow$ rand() (or $\tilde{\psi}_k \leftarrow \psi \in \{\tilde{\psi}_{k-1}^*, \psi_{k-1}^*\}$, $\psi_k \leftarrow D_k$) and
      choose $\lambda_{\tilde{\psi}_k}$, $\lambda_{\psi_k}$;
5:    while generator $\mathcal{G}_k$ has not converged do
6:       Sample batches of prior samples $\{z^{(i)}\}_{i=1}^m$, $\{z_{\tilde{\psi}_k}^{(i)}\}_{i=1}^m$, $\{z_{\psi_k}^{(i)}\}_{i=1}^m$ independently from prior
         $\vartheta(z)$;
7:       Generate synthetic data batches $\{x_{\tilde{\psi}_k}^{(i)}\}_{i=1}^m \sim \nu_{k-1}^*$ and $\{x_{\psi_k}^{(i)}\}_{i=1}^m \sim \mu_k$ by passing $\{z_{\tilde{\psi}_k}^{(i)}\}_{i=1}^m$
         and $\{z_{\psi_k}^{(i)}\}_{i=1}^m$ through $\mathcal{G}_{k-1}^*$ and $G_k$, respectively;
8:       Compute gradients $g_{\tilde{\psi}_k}$ and $g_{\psi_k}$:
9:       $\left\{g_\omega \leftarrow \lambda_\omega \nabla_\omega \frac{1}{m}\sum_{i=1}^m \left[\omega(x_\omega^{(i)}) - \omega(\mathcal{G}_k(z^{(i)})\right]\right\}_{\omega=\tilde{\psi}_k,\psi_k}$;
10:      Update both discriminators $\psi_k$ and $\tilde{\psi}_k$: $\{\omega \leftarrow \omega + \alpha \cdot \text{Adam}(\omega, g_\omega)\}_{\omega=\psi_k,\tilde{\psi}_k}$;
11:      Compute gradient $g_{\mathcal{G}_k} \leftarrow -\nabla_{\mathcal{G}_k} \frac{1}{m}\sum_{i=1}^m \left[\lambda_{\psi_k}\psi_k(\mathcal{G}_k(z^{(i)})) + \lambda_{\tilde{\psi}_k}\tilde{\psi}_k(\mathcal{G}_k(z^{(i)}))\right]$;
12:      Update generator $\mathcal{G}_k$: $\mathcal{G}_k \leftarrow \mathcal{G}_k - \alpha \cdot \text{Adam}(\mathcal{G}_k, g_{\mathcal{G}_k})$;
13:   end while
14:   Assign $\mathcal{G}_k^* \leftarrow \mathcal{G}_k$
15: end for
16: return generator $\mathcal{G}_K^*$ for barycenter $\nu_K^*$, discriminators $\tilde{\psi}_K^*$, $\psi_K^*$.
```

where ψ and $\tilde{\psi}$ denote the corresponding discriminators for pre-trained model G_k and barycenter model $\mathcal{G}_{k-1}^*$ from the previous recursive step, respectively *(See Algorithm 1)*.

2.5.2 Model Initialization in Each Recursive Step

For the initialization of the generator $\mathcal{G}_k$, we use the trained generator $\mathcal{G}_{k-1}^*$ in last step. $\mathcal{G}_{k-1}^*$ corresponds to the barycenter ν_{k-1}^*, and using it as the initialization

Algorithm 2 Fast Adaptation Algorithm to Solve for Learning the Generative Model at Node 0.

```
1: Inputs: Final generator $\mathcal{G}_K^*$, final discriminators $\tilde{\psi}_K^*$, $\psi_K^*$, noise prior $\vartheta(z)$, the batch size $m$, learning rate $\alpha$
2: Set $\mathcal{G}_0^* \leftarrow \mathcal{G}_K^*$, $\tilde{\psi}_0^* \leftarrow$ rand() or $\tilde{\psi}_0^* \leftarrow \psi \in \{\tilde{\psi}_K^*, \psi_K^*\}$;
3: while generator $\mathcal{G}_0$ has not converged do
4:     Sample batches of prior samples $\{z^{(i)}\}_{i=1}^m$ and $\{z_{\tilde{\psi}_0}^{(i)}\}_{i=1}^m$ independently from prior $\vartheta(z)$;
5:     Get real data batch $\{x_{\psi_0}^{(i)}\}_{i=1}^m \sim \hat{\mu}_0$ and generate synthetic data batch $\{x_{\tilde{\psi}_0}^{(i)}\}_{i=1}^m \sim \nu_K^*$ by passing $\{z_{\tilde{\psi}_0}^{(i)}\}_{i=1}^m$ through $\mathcal{G}_K^*$;
6:     Compute gradients $g_{\tilde{\psi}_0}$ and $g_{\psi_0}$: $\left\{g_\omega \leftarrow \lambda_\omega \nabla_\omega \frac{1}{m}\sum_{i=1}^m \left[\omega(x_\omega^{(i)}) - \omega(\mathcal{G}_0(z^{(i)})\right]\right\}_{\omega=\tilde{\psi}_0,\psi_0}$;
7:     Update both discriminators $\psi_0$ and $\tilde{\psi}_0$: $\{\omega \leftarrow \omega + \alpha \cdot \mathrm{Adam}(\omega, g_\omega)\}_{\omega=\psi_0,\tilde{\psi}_0}$;
8:     Compute gradient $g_{\mathcal{G}_0} \leftarrow -\nabla_{\mathcal{G}_0} \frac{1}{m}\sum_{i=1}^m \left[\lambda_{\psi_0}\psi_0(\mathcal{G}_0(z^{(i)})) + \lambda_{\tilde{\psi}_0}\tilde{\psi}_0(\mathcal{G}_0(z^{(i)}))\right]$;
9:     Update generator $\mathcal{G}_0$: $\mathcal{G}_0 \leftarrow \mathcal{G}_0 - \alpha \cdot \mathrm{Adam}(\mathcal{G}_0, g_{\mathcal{G}_0})$;
10: end while
11: Assign $\mathcal{G}_0^* \leftarrow \mathcal{G}_0$
12: return Generator $\mathcal{G}_0^*$ for barycenter $\nu_0^*$.
```

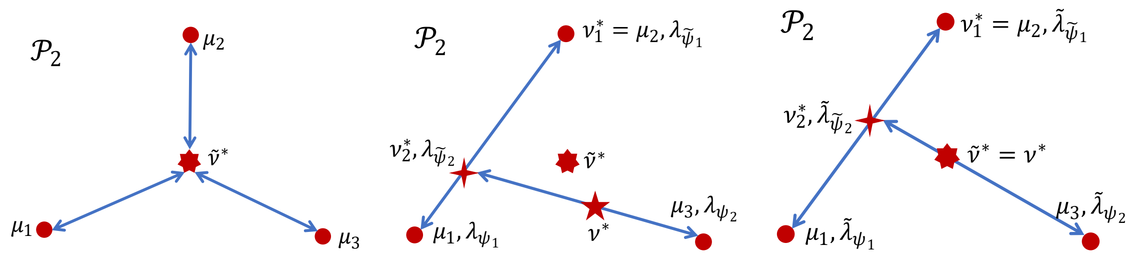
the displacement interpolation would move along the geodesic curve from ν_{k-1}^* to μ_k [87]. It has been shown that training GANs with such initializations would accelerate the convergence compared with training from scratch [147]. Finally, ν_K^* is adopted as initialization to enable fast adaptation at the target node. As the barycenter ν_K^* solved via offline training, a new barycenter ν^* between local data (represented by $\hat{\mu}_0$) and ν_K^*, can be obtained by solving the problem:

$$\min_{\mathcal{G}_0}\max_{\psi_0,\tilde{\psi}_0} \lambda_{\psi_0}\{\mathbb{E}_{x\sim\hat{\mu}_0}[\psi_0(x)] - \mathbb{E}_{z\sim\vartheta}[\psi_0(\mathcal{G}_0(z))]\} + \lambda_{\tilde{\psi}_0}\{\mathbb{E}_{x\sim\nu_K^*}[\tilde{\psi}_0(x)] - \mathbb{E}_{z\sim\vartheta}[\tilde{\psi}_0(\mathcal{G}_0(z))]\}, \tag{2.19}$$

i.e., by training a 2-discriminator WGAN, and fine-tuning the generator $\mathcal{G}_0$ from $\mathcal{G}_K^*$ would be notably *faster and more accurate* than learning the generative model from local data only *(See Algorithm 2 and Fig. 2.4)*.

Algorithm 3 Fast Adaptive Learning of the Ternary Generative Model for Edge Node 0

```
1: Inputs: Training dataset $\mathcal{S}_0$, generator $\mathcal{G}_K^*$ for the barycenter $\nu_K^*$, offline barycenter discriminators $\psi_K^*$, $\tilde{\psi}_K^*$, noise prior $\vartheta(z)$, the batch size $m$, learning rate $\alpha$, the number of layers $L_{\mathcal{G}} = L_\psi = L_{\tilde{\psi}} = L$;
2: Set $\mathcal{G}_0 \leftarrow \mathcal{G}_K^*$, $\tilde{\psi}_0 \leftarrow$ rand() and $\psi_0 \leftarrow$ rand() (or $\tilde{\psi}_0 \leftarrow \tilde{\psi}_K^*$ and $\psi_0 \leftarrow \psi_K^*$);
3: while generator $\mathcal{G}_0$ has not converged do
4:     for $l := 1$ to $L$ do
5:         $\left\{\mathbf{w}'_{l_\omega} \leftarrow Tern(\mathbf{w}_{l_\omega}, S_{l_\omega}, \Delta^\pm_{l_\omega})\right\}_{\omega=\tilde{\psi}_k,\psi_k}$;
6:         $\mathbf{w}'_{l_\mathcal{G}} \leftarrow Tern(\mathbf{w}_{l_\mathcal{G}}, S_{l_\mathcal{G}}, \Delta^\pm_{l_\mathcal{G}})$;
7:     end for
8:     Sample batches of prior samples $\{z^{(i)}\}_{i=1}^m$ from prior $\vartheta(z)$;
9:     Sample batches of training samples $\{x_0^i\}_{i=1}^m$ from local dataset $\mathcal{S}_0$;
10:    for $l := L$ to $1$ do
11:        Compute gradients $\left\{g_{\Delta^\pm_{l_\omega}}\right\}_{\omega=\tilde{\psi}_k,\psi_k}$: $\left\{g_{\Delta^\pm_{l_\omega}} \leftarrow \nabla_{\Delta^\pm_{l_\omega}} \frac{1}{m}\sum_{i=1}^m \Big[\omega_0(x_0^{(i)}) - \omega_0(\mathcal{G}_0(z^{(i)})\Big]\right\}_{\omega=\tilde{\psi}_k,\psi_k}$;
12:        Update $\left\{\Delta^\pm_{l_\omega}\right\}_{\omega=\tilde{\psi}_k,\psi_k}$: $\left\{\Delta^\pm_{l_\omega} \leftarrow \Delta^\pm_{l_\omega} + \alpha \cdot g_{\Delta^\pm_{l_\omega}}\right\}_{\omega=\tilde{\psi}_k,\psi_k}$;
13:        Compute gradient $g_{\Delta^\pm_{l_\mathcal{G}}}$: $g_{\Delta^\pm_{l_\mathcal{G}}} \leftarrow -\nabla_{\Delta^\pm_{l_\mathcal{G}}} \frac{1}{m}\sum_{i=1}^m \Big[\psi_0(\mathcal{G}_0(z^{(i)})) + \tilde{\psi}_0(\mathcal{G}_0(z^{(i)}))\Big]$;
14:        Update $\Delta^\pm_{l_\mathcal{G}}$: $\Delta^\pm_{l_\mathcal{G}} \leftarrow \Delta^\pm_{l_\mathcal{G}} - \alpha \cdot g_{\Delta^\pm_{l_\mathcal{G}}}$;
15:    end for
16:    Repeat steps 3-5 using updated thresholds;
17:    for $l := L$ to $1$ do
18:        Compute gradients $\{g_{\mathbf{w}_{l_\omega}}\}_{\omega=\tilde{\psi}_k,\psi_k}$: $\left\{g_{\mathbf{w}_{l_\omega}} \leftarrow \nabla_{\mathbf{w}_{l_\omega}} \frac{1}{m}\sum_{i=1}^m \Big[\omega_0(x_0^{(i)}) - \omega_0(\mathcal{G}_0(z^{(i)})\Big]\right\}_{\omega=\tilde{\psi}_k,\psi_k}$;
19:        Update $\{\mathbf{w}_{l_\omega}\}_{\omega=\tilde{\psi}_k,\psi_k}$: $\{\mathbf{w}_{l_\omega} \leftarrow \mathbf{w}_{l_\omega} + \alpha \cdot \mathrm{Adam}(\mathbf{w}_{l_\omega}, g_{\mathbf{w}_{l_\omega}})\}_{\omega=\tilde{\psi}_k,\psi_k}$;
20:        Compute gradient $g_{\mathbf{w}_{l_\mathcal{G}}}$: $g_{\mathbf{w}_{l_\mathcal{G}}} \leftarrow -\nabla_{\mathbf{w}_{l_\mathcal{G}}} \frac{1}{m}\sum_{i=1}^m \Big[\psi_0(\mathcal{G}_0(z^{(i)})) + \tilde{\psi}_0(\mathcal{G}_0(z^{(i)}))\Big]$;
21:        Update $\mathbf{w}_{l_\mathcal{G}}$: $\mathbf{w}_{l_\mathcal{G}} \leftarrow \mathbf{w}_{l_\mathcal{G}} - \alpha \cdot \mathrm{Adam}(\mathbf{w}_{l_\mathcal{G}}, g_{\mathbf{w}_{l_\mathcal{G}}})$;
22:    end for
23:    Repeat step 3-5 using updated full-precision weights;
24: end while
25: return Ternary generator $\mathcal{G}_0$.
```


$\mathcal{P}_2$
μ_2
$\tilde{\nu}^*$
μ_1
μ_3
$\mathcal{P}_2$
$\nu_1^* = \mu_2, \lambda_{\tilde{\psi}_1}$
$\nu_2^*, \lambda_{\tilde{\psi}_2}$
$\tilde{\nu}^*$
μ_3, λ_{ψ_2}
μ_1, λ_{ψ_1}
ν^*
$\mathcal{P}_2$
$\nu_1^* = \mu_2, \tilde{\lambda}_{\tilde{\psi}_1}$
$\nu_2^*, \tilde{\lambda}_{\tilde{\psi}_2}$
$\tilde{\nu}^* = \nu^*$
$\mu_3, \tilde{\lambda}_{\psi_2}$
$\mu_1, \tilde{\lambda}_{\psi_1}$

(a) The Illustration of the Solution to the Wasserstein-1 Barycenter Problem.

(b) The Illustration of a Barycenter Obtained via the Recursive Algorithm for a Set of Given Weights.

(c) The Illustration of the Barycenter Obtained via the Recursive Algorithm with the Set of Optimal Weights.

Figure 2.4: The Illustrations of *Barycenters* in $\mathcal{P}_2$ Space. Blue Lines Represent the Displacement Interpolation Between Any Pair of Distributions. 7-edge Star Is One of the Analytical Barycenter Solutions to Wasserstein-1 Barycenter Problem. 4-edge and 5-edge Stars Show the Barycenter Solutions Obtained in the First and Second Recursions, Respectively, of the Proposed Recursive Algorithm.

2.5.3 Fast Adaptation for Training Ternary WGAN at Node 0

As outlined in Algorithm 2, fast adaptation is used to find the barycenter between ν_K^* and the local dataset at Node 0. To further enhance edge learning, we adopt the weight ternarization method to compress the WGAN model during training. The weight ternarization method not only replaces computationally-expensive multiplication operations with efficient addition/subtraction operations, but also enables the sparsity in model parameters [61]. Specifically, the ternarization process is formulated as:

$$w_l' = S_l \cdot Tern\left(w_l, \Delta_l^{\pm}\right) = S_l \cdot \begin{cases} +1, & w_l > \Delta_l^+ \\ 0, & \Delta_l^- \leq w_l \leq \Delta_l^+ \\ -1, & w_l < \Delta_l^- \end{cases} \tag{2.20}$$

where $\{w_l\}$ are the full precision weights for lth layer, $\{w_l'\}$ are the weights after ternarization, $\{S_l\}$ is the layer-wise weight scaling coefficient and $\Delta_l^{\pm}$ are the layer-wise thresholds. Since the fixed weight thresholds may lead to accuracy degradation, S_l is approximated as a differentiable closed-form function of $\Delta_l^{\pm}$ so that both weights and thresholds can be optimized simultaneously through back-propagation [63]. Let the generator and the discriminators of WGAN at Node 0 be denoted by $\mathcal{G}_0$, $\tilde{\psi}_0$ and ψ_0, which can be parametrized by the ternarized weights $\{\mathbf{w}'_{l_{\mathcal{G}}}\}_{l_{\mathcal{G}}=1}^{L_{\mathcal{G}}}$, $\{\mathbf{w}'_{l_{\tilde{\psi}}}\}_{l_{\tilde{\psi}}=1}^{L_{\tilde{\psi}}}$ and $\{\mathbf{w}'_{l_{\psi}}\}_{l_{\psi}=1}^{L_{\psi}}$, respectively. The barycenter ν^* at Node 0, captured by $\mathcal{G}_0^*$, can be obtained by training the ternary WGAN via iterative updates of both weights and thresholds, which takes three steps in each iteration: 1) calculating the scaling coefficients and the ternary weights for $\mathcal{G}_0$, $\tilde{\psi}_0$ and ψ_0, 2) calculating the loss function using the ternary weights via forward-propagation and 3) updating the full precision weights and the thresholds via back-propagation *(See Algorithm 3)*.

2.5.4 Recursive Configuration and The Impact of for Recursive Coalescence Order

Even though the multi-discriminator GAN can lead to a Wasserstein-1 barycenter in principle, training a many-discriminator GAN in a one-shot manner is overwhelming for memory-limited edge nodes. The proposed 2-stage recursive configuration is designed to address the memory problem by 'converting' the one-shot formulation to a nested Wasserstein barycenter problem. In a nutshell, a 2-discriminator GAN configuration suffices to obtain a shape-preserving interpolation of all distributions. As discussed above, the Wasserstein-1 barycenter problem not necessarily constitutes a unique solution due to the non-uniqueness of geodesic curves between distributions in the probability space. Proposition 1 asserts that any solution to each pairwise Wasserstein-1 barycenter problem, referred as a barycenter in this chapter, resides inside the baryregion formed by $\{\mu_k\}_{1:K}$. Consequently, the final barycenter ν^*, obtained at the end of all recursions, also resides inside the baryregion. However, the 2-stage recursive configuration may not obtain the same barycenter solution to Wasserstein-1 barycenter problem. Through the intuition that the Wasserstein ball radius $\eta_k = \frac{1}{\lambda_{\psi_k}}$ for pre-trained model k represents the relevance (and hence utility) of the distribution k, larger weights $\lambda_k = 1/\eta_k$ would be assigned to the nodes with higher relevance. Since we introduce the recursive WGAN configuration, the order of coalescence (each corresponding to a geodesic curve) may impact the final barycentric WGAN model, and hence the performance of Barycentric Fast Adaptation. To this end, we compute the coalescence of models of nodes with higher relevance at latter recursions to ensure that the final barycentric model is closer to the models of nodes with higher relevance. For fairness, we have used $\eta_k = \eta_m, \forall k \neq m$ (except otherwise stated) and the same order of training across different baselines.

2.5.5 From Optimal Transport Theory to Wasserstein-2 GAN

Similar to the Wasserstein-1 GAN, the Wasserstein-2 GAN aims to minimize $W_2^2(\nu, \mu)$, which reduces to the following problem:

$$\arg\min_{\nu} \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X},\mathcal{Y}} \frac{\|x-y\|^2}{2} d\gamma(x,y). \tag{2.21}$$

By applying the Kantorovich duality to (2.21), we can obtain [132]:

$$\arg\min_{\nu} W_2^2(\nu,\mu) = \arg\min_{\nu} \sup_{(\varphi,\psi)\in\Phi_c} \int_{\mathcal{X}} \tilde{\varphi}(x) d\mu(x) + \int_{\mathcal{Y}} \tilde{\psi}(y) d\nu(y) \tag{2.22}$$

$$= \arg\min_{\nu} \sup_{(\varphi)\in\text{Convex}} \left[\int_{\mathcal{X}} \left(\frac{\|x\|^2}{2} - \varphi(x) \right) d\mu(x) + \int_{\mathcal{Y}} \left(\frac{\|y\|^2}{2} - \varphi^*(y) \right) d\nu(y) \right] \tag{2.23}$$

$$= \arg\min_{\nu} \int_{\mathcal{X}} \frac{\|x\|^2}{2} d\mu(x) + \int_{\mathcal{Y}} \frac{\|y\|^2}{2} d\nu(y) - \min_{(\varphi)\in\text{Convex}} \left[\int_{\mathcal{X}} \varphi(x) d\mu(x) + \int_{\mathcal{Y}} \varphi^*(y) d\nu(y) \right], \tag{2.24}$$

where φ^* is the convex (Fenchel) conjugate of φ and is stated as:

$$\varphi^*(y) = \max_{x\in\mathcal{X}} \left(\langle x, y \rangle \right) - \varphi(x). \tag{2.25}$$

Different from the Wasserstein-1 distance, the quadratic Wasserstein-2 function requires φ and ψ to be convex conjugates, which is a much harder constraint to enforce on neural networks than $\varphi = -\psi$ in the W1GAN case. To overcome these challenges, recent studies utilized the property $(\nabla\varphi)^{-1}(y) = \nabla\varphi^*(y)$ of the unique solution [79, 111, 127, 93] and the cycle consistency [79, 174]. For the sake of comparison, we use the W2GAN architecture developed very recently in [79] to compare the practical performances of both W1GAN and W2GAN on the Barycentric Fast Adaptation technique developed herein.

The W2GAN technique developed in [79] numerically solves the following problem:

$$\min_{\nu} \min_{\varphi,\psi\in\text{Convex}} \mathbb{E}_{\mathbf{x}\sim\mu}[\varphi(\mathbf{x})] + \mathbb{E}_{\mathbf{y}\sim\nu}[\langle\nabla\psi(\mathbf{y}),\mathbf{y}\rangle - \varphi(\nabla\psi(\mathbf{y}))] + \frac{\varrho}{2}\mathbb{E}_{\mathbf{y}\sim\nu}[\|\nabla\varphi\circ\nabla\psi(\mathbf{y})-\mathbf{y}\|_2^2] \tag{2.26}$$

$$\begin{aligned} =\min_{G} \min_{\varphi,\psi\in\text{Convex}} & \ \mathbb{E}_{\mathbf{x}\sim\mu}[\varphi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle\nabla\psi(G(\mathbf{z})), G(\mathbf{z})\rangle - \varphi(\nabla\psi(G(\mathbf{z})))] \\ & + \frac{\varrho}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\varphi\circ\nabla\psi(G(\mathbf{z})) - G(\mathbf{z})\|_2^2], \end{aligned} \tag{2.27}$$

where the third term is the regularization, enforcing the convex conjugate constraint. Each expectation above can be approximated by a Monte Carlo estimate to apply a stochastic gradient descent algorithm for numerical computation as in the W1GAN counterpart. It is worth pointing out that (2.27) is an approximation to the W_2^2 function. Indeed, W_2^2 function does not qualify to be a metric as it does not satisfy the triangle inequality. Note that although W_2^2 does not induce a metric space, it constitutes geodesics, but the geodesics induced by W_2^2 function and W_2 metric are different.

2.5.6 From Wasserstein-2 to Multi-Discriminator GAN Architecture

Instead of (2.3), we could target the Wasserstein-2 barycenter problem:

$$\nu^\dagger = \arg\min_{\nu\in\mathcal{P}} W_2^2(\nu,\hat{\mu}_0) + \sum_{k=1}^{K}\frac{1}{\eta_k}W_2^2(\nu,\mu_k), \tag{2.28}$$

which can be expressed in terms of the Kantorovich potentials as:

$$\nu^\dagger = \arg\min_{\nu\in\mathcal{P}} \sup_{(\tilde{\varphi}_0,\tilde{\psi}_0)\in\Phi_c} \left\{ \int_{\mathcal{X}} \tilde{\varphi}_0(x)d\hat{\mu}_0(x) + \int_{\mathcal{Y}} \tilde{\psi}_0(y)d\nu(y) \right\} + \sum_{k=1}^{K} \frac{1}{\eta_k} \sup_{(\tilde{\varphi}_k,\tilde{\psi}_k)\in\Phi_c} \left\{ \int_{\mathcal{X}} \tilde{\varphi}_k(x)d\mu_k(x) + \int_{\mathcal{Y}} \tilde{\psi}_k(y)d\nu(y) \right\} \tag{2.29}$$

By using the c-transformation and the property $\tilde{\psi}_k(y) = \tilde{\varphi}^c(y) = \tilde{\varphi}^*(y)$, we obtain:

$$\nu^\dagger = \arg\min_{\nu\in\mathcal{P}} \Bigg\{ \int_{\mathcal{X}} \frac{\|x\|^2}{2} d\hat{\mu}_0(x) + \int_{\mathcal{Y}} \frac{\|y\|^2}{2} d\nu(x) - \min_{\varphi_0\in\text{Convex}} \left[\int_{\mathcal{X}} \varphi_0(x)d\hat{\mu}_0(x) + \int_{\mathcal{Y}} \varphi_0^*(y)d\nu(y) \right] + \sum_{k=1}^{K} \frac{1}{\eta_k} \left[\int_{\mathcal{X}} \frac{\|x\|^2}{2} d\mu_k(x) + \int_{\mathcal{Y}} \frac{\|y\|^2}{2} d\nu(x) - \min_{\{\varphi_k\}\in\text{Convex}} \left[\int_{\mathcal{X}} \varphi_k(x)d\mu_k(x) + \int_{\mathcal{Y}} \varphi_k^*(y)d\nu(y) \right] \right] \Bigg\} \tag{2.30}$$

Finally, the numerical problem for finding a W_2^2 based barycenter can be expressed as follows by fusing the estimate cost given in [79] and (2.30):

$$\min_G \Bigg\{ \min_{\varphi_0,\psi_0\in\text{Convex}} \Big(\mathbb{E}_{\mathbf{x}\sim\hat{\mu}_0}[\varphi_0(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle \nabla\psi_0(G(\mathbf{z})), G(\mathbf{z})\rangle - \varphi_0(\nabla\psi_0(G(\mathbf{z})))] + \frac{\varrho_0}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\varphi_0 \circ \nabla\psi_0(G(\mathbf{z})) - G(\mathbf{z})\|_2^2] \Big) + \sum_{k=1}^{K} \frac{1}{\eta_k} \left[\min_{\{\varphi_k,\psi_k\}\in\text{Convex}} \Big(\mathbb{E}_{\mathbf{x}\sim\mu_k}[\varphi_k(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle \nabla\psi_k(G(\mathbf{z})), G(\mathbf{z})\rangle - \varphi_k(\nabla\psi_k(G(\mathbf{z})))] + \frac{\varrho_k}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\varphi_k \circ \nabla\psi_k(G(\mathbf{z})) - G(\mathbf{z})\|_2^2] \Big) \right] \Bigg\}. \tag{2.31}$$

The numerical optimization problem defined in (2.31) can be implemented using a generative adversarial network architecture. In particular, each of $G, \{\varphi_k\}, \{\psi_k\}$ can be represented with $2K + 3$ DNN models in total. As discussed in Section 2.5, a recursive WGAN configuration constitutes a better memory efficiency for edge devices in comparison to a single-shot WGAN configuration. A recursive W2GAN configuration attempts to solve the following problem at each recursion:

$$\min_{\mathcal{G}_k}\Bigg\{\min_{\{\varphi_k,\tilde{\varphi}_k,\psi_k,\tilde{\psi}_k\}\in\text{Con.}}\Bigg[\frac{1}{\eta_k}\Big(\mathbb{E}_{\mathbf{x}\sim\mu_k}[\varphi_k(\mathbf{x})]+\mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle\nabla\psi_k(\mathcal{G}_k(\mathbf{z})),\mathcal{G}_k(\mathbf{z})\rangle-\varphi_k(\nabla\psi_k(\mathcal{G}_k(\mathbf{z})))]$$
$$+\frac{\varrho_k}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\varphi_k\circ\nabla\psi_k(\mathcal{G}_k(\mathbf{z}))-\mathcal{G}_k(\mathbf{z})\|_2^2]\Big)$$
$$+\frac{1}{\tilde{\eta}_k}\Big(\mathbb{E}_{\mathbf{x}\sim\nu_k^*}[\tilde{\varphi}_k(\mathbf{x})]+\mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle\nabla\tilde{\psi}_k(\mathcal{G}_k(\mathbf{z})),\mathcal{G}_k(\mathbf{z})\rangle-\tilde{\varphi}_k(\nabla\tilde{\psi}_k(\mathcal{G}_k(\mathbf{z})))]$$
$$+\frac{\tilde{\varrho}_k}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\tilde{\varphi}_k\circ\nabla\tilde{\psi}_k(\mathcal{G}_k(\mathbf{z}))-\mathcal{G}_k(\mathbf{z})\|_2^2]\Big)\Bigg]\Bigg\}. \tag{2.32}$$

We note that the recursive W2GAN configuration consists of 5 DNNs rather than the 3 DNNs in the case of the recursive W1GAN configuration. The additional 2 discriminator DNN models are needed to mimic the convex conjugate of ψ_k. In a similar fashion, the fast adaptation stage numerically solves the following problem:

$$\min_{\mathcal{G}_0}\Bigg\{\min_{\{\varphi_0,\tilde{\varphi}_0,\psi_0,\tilde{\psi}_0\}\in\text{Conv.}}\Bigg[\frac{1}{\eta_0}\Big(\mathbb{E}_{\mathbf{x}\sim\hat{\mu}_0}[\varphi_0(\mathbf{x})]+\mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle\nabla\psi_0(\mathcal{G}_0(\mathbf{z})),\mathcal{G}_0(\mathbf{z})\rangle-\varphi_0(\nabla\psi_0(\mathcal{G}_0(\mathbf{z})))]$$
$$+\frac{\varrho_0}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\varphi_0\circ\nabla\psi_0(\mathcal{G}_0(\mathbf{z}))-\mathcal{G}_0(\mathbf{z})\|_2^2]\Big)$$
$$+\frac{1}{\tilde{\eta}_0}\Big(\mathbb{E}_{\mathbf{x}\sim\nu_K^*}[\tilde{\varphi}_0(\mathbf{x})]+\mathbb{E}_{\mathbf{z}\sim\vartheta}[\langle\nabla\tilde{\psi}_0(\mathcal{G}_0(\mathbf{z})),\mathcal{G}_0(\mathbf{z})\rangle-\tilde{\varphi}_0(\nabla\tilde{\psi}_0(\mathcal{G}_0(\mathbf{z})))]$$
$$+\frac{\tilde{\varrho}_0}{2}\mathbb{E}_{\mathbf{z}\sim\vartheta}[\|\nabla\tilde{\varphi}_0\circ\nabla\tilde{\psi}_0(\mathcal{G}_0(\mathbf{z}))-\mathcal{G}_0(\mathbf{z})\|_2^2]\Big)\Bigg]\Bigg\}. \tag{2.33}$$

The recursive W2GAN configuration can, in theory, obtain a unique barycenter at each recursion. However, its practical implementation is challenging due to the convex conjugate constraint arising from the Kantorovich duality for W_2^2 function.

2.5.7 Implementation Challenges in W_2^2-based GAN

The practical success of W1GAN can be largely attributed to the elegant structural relation between Kantorovich potentials, i.e., $\varphi = -\psi$. Unfortunately, when the quadratic cost W_2^2 is used, Kantorovich potentials translate to $\varphi = \psi^*$, where * denotes the convex conjugate. Note that W_2 is a metric but W_2^2 is not. When implementing the W_2^2-based GAN, both Kantorovich potentials are estimated by 2 distinct DNNs that must satisfy the convex conjugate constraint, which is practically challenging. Very recent studies [87, 91, 79] attempt to enforce the convex conjugate constraint between these DNNs through approximations or regularization under certain assumptions, but it remains not well understood. In this chapter, we have carried out experimental studies to compare the performance of W_1 and W_2^2 based recursive WGAN configurations, and our findings corroborate that W1GAN performs better.

2.6 Experiments

2.6.1 Datasets, Models and Evaluation

We extensively examine the performance of learning a generative model, using the Barycentric Fast Adaptation algorithm, on a variety of widely adapted datasets in the GAN literature, including CIFAR10, CIFAR100, LSUN and MNIST [42, 80, 161]. In experiments, we used various DCGAN-based architectures [103] depending on the dataset as different datasets vary in image size, feature diversity and in sample size, e.g., image samples in MNIST has less diversity compared to the rest of the datasets, while LSUN contains the largest number of samples with larger image sizes. Further, we used the weight ternarization method [63] to jointly optimize weights and quantizers of the generative model at the target edge node, reducing the memory burden of generative models in memory-limited edge devices. Details on the characteristics of datasets and network architectures used in experiments are relegated to the supplementary materials. The implementation details for all the results are provided in the supplementary materials.

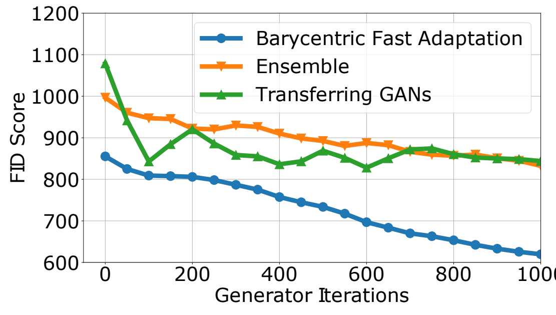
The Frechet-Inception Distance (FID) score [65] is widely adopted for evaluating the performance of GAN models in the literature [29, 147, 58], since it provides a quantitative assessment of the similarity of a dataset to another reference dataset. Therefore, we use the FID score to evaluate the performance evolution of the two-stage adaptive coalescence algorithm and all baseline algorithms during training. We here emphasize that a smaller FID score of a GAN indicates that it has better performance. Note that to avoid one-sided scores and make a fair comparison, other evaluation metrics, in addition to the FID score, are also leveraged to quantify the performance of all algorithms. A more comprehensive discussion of FID and other metrics is relegated to the supplementary materials. In all experiments, we use the entire dataset as the reference dataset.

To demonstrate the improvements by using the proposed framework based on *barycentric fast adaptation*, we conduct extensive experiments and compare performance with 3 distinct baselines: 1) *Transferring GANs* [147]: a pre-trained GAN model is used as initialization at Node 0 for training a new WGAN model by using local data samples. 2) *Ensemble method*: The model initialization, obtained by using pre-trained GANs at other edge nodes, is further trained using both local data from Node 0 and synthetic data samples. 3) *Edge-Only*: only local dataset at Node 0 is used in WGAN training. Due to the lack of sample diversity at the target edge node, the WGAN model trained using local data only is not expected to attain good performance. In stark contrast, the WGAN model trained using the proposed two-stage adaptive coalescence algorithm, inherits the diversity from the pre-trained models at other edge nodes, and results in better performance compared to its counterparts. Needless to say, if the entire dataset were available at Node 0, the best performance would be achieved.

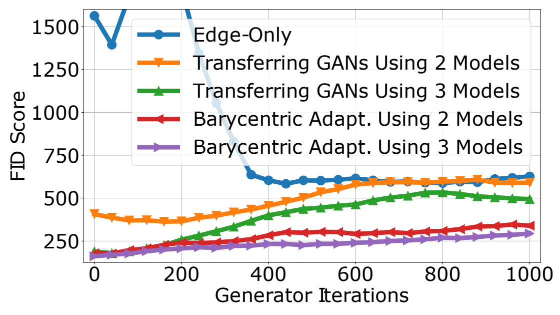
2.6.2 Experiment Setup

We consider the following two scenarios: 1) *The overlapping case*: the classes of the data samples at other edge nodes and at Node 0 overlap; 2) *The non-overlapping case*: the classes of the data samples at other edge nodes and at Node 0 are mutually exclusive. In overlapping experiments, the corresponding dataset is equally split into 2 sub-datasets and sub-datasets are used to pre-train 2 WGAN models independently. Subsequently, Algorithm 1 and Algorithm 2 are used consecutively to find the barycenter and the final WGAN model at Node 0, respectively. The few data samples to be used in the fast adaptation stage are randomly selected from all classes in the dataset. For the Transferring GANs method, the first pre-trained model is further trained on the second sub-dataset using transfer learning to compute a fused model. In the final stage, the few data samples from all classes in the dataset are used to train a WGAN model at Node 0 by leveraging the fused model via transfer learning again. In the Ensemble method, the pre-trained models are used to generate a synthetic dataset. The synthetic dataset is combined with the few data samples from all the classes in the dataset and the combined dataset is leveraged to train a final WGAN model at Node 0. Lastly, the Edge-only method only leverages the few data samples from all the classes in the dataset to train a WGAN model at Node 0.

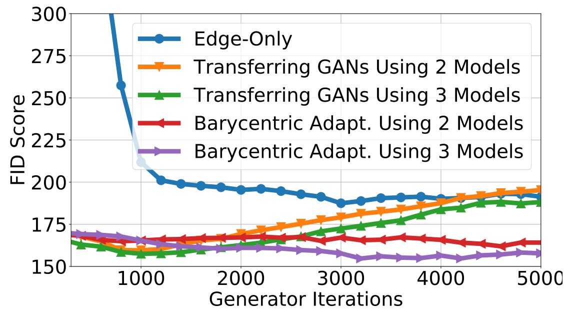
In non-overlapping experiments, randomly drawn samples from the first 40% of the classes in the dataset are allocated into the first node and randomly drawn samples from the second 40% of the classes are allocated into the second node. The same steps as in the overlapping case are followed by using these two sub-datasets until the final stage. In the final stage, a few data samples are randomly selected from the remaining 20% of the classes and are placed in Node 0.



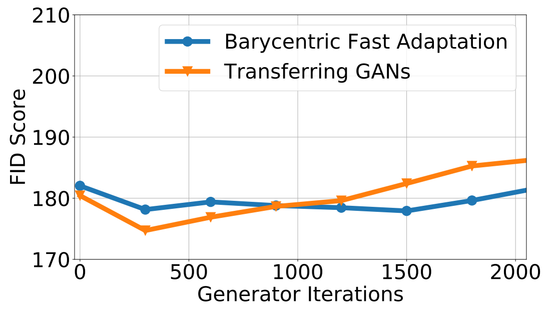
(a) MNIST: Non-overlapping Case.



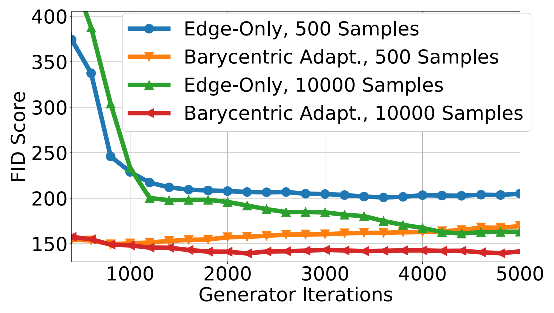
(b) MNIST: Overlapping Case.



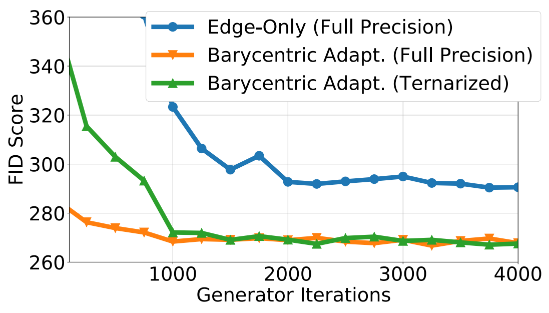
(c) CIFAR10: Overlapping Case.



(d) CIFAR100: Overlapping Case.



(e) CIFAR10: Overlapping Case.



(f) LSUN: Overlapping Case.

Figure 2.5: Comparison of Convergence of Barycentric Fast Adaptation with Various Baselines.

2.6.3 Continual Learning Against Catastrophic Forgetting

We investigate the convergence and the generated image quality of various training scenarios on CIFAR100 and MNIST datasets. As illustrated in Figure 2.5 and 2.6,

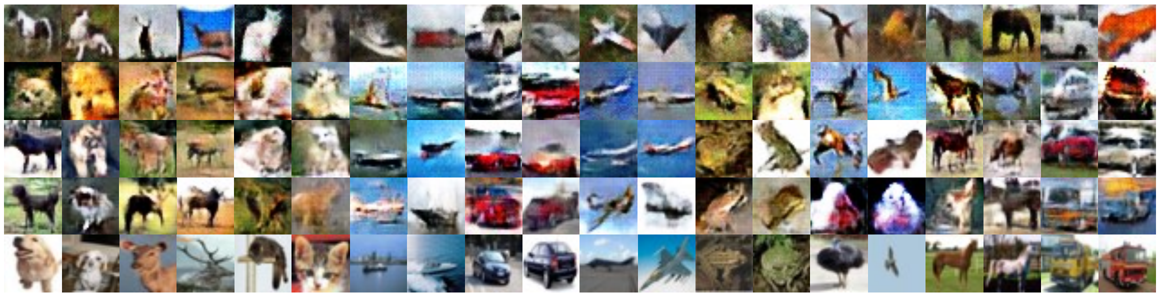


Figure 2.6: Image Samples from 4 Different Approaches for CIFAR10: From Top Row to Bottom Row; the Images Generated by Edge-Only (at 90000 Iterations), Ternary Barycentric Fast Adaptation (at 5000 Iterations), Transferring GANs (at 1000 Iterations) and Barycentric Fast Adaptation (at 1000 Iterations) Algorithms. Last Row Illustrates Some Real Images.

Barycentric Fast Adaptation clearly outperforms all baselines. Transferring GANs suffers from catastrophic forgetting, because the continual learning is performed over local data samples at Node 0 only. On the contrary, the Barycentric Fast Adaptation and the ensemble method leverage generative replay, which mitigates the negative effects of catastrophic forgetting. Further, observe that the ensemble method suffers because of the limited data samples at Node 0, which are significantly outnumbered by synthetic data samples from pre-trained GANs, and this imbalance degrades the applicability of the ensemble method for continual learning. On the other hand, the Barycentric Fast Adaptation can obtain the barycenter between the local data samples at Node 0 and the barycenter model trained offline, and hence can effectively leverage the abundance of data samples from edge nodes and the accuracy of local data samples at Node 0 for better continual learning.

2.6.4 Impact of Number of Pre-trained Generative Models

To quantify the impact of cumulative model knowledge from pre-trained generative models on the learning performance at the target node, we consider the scenario where 10 classes in CIFAR10/MNIST are split into 3 subsets, e.g., the first pre-train model has classes $\{0, 1, 2\}$, the second pre-trained model has classes $\{2, 3, 4\}$

and the third pre-trained model has the remaining classes. One barycenter model is trained offline by using the first two pre-trained models and the second barycenter model is trained using all 3 pre-trained models, respectively, based on which we evaluate the performance of Barycentric Fast Adaptation with 1000 data samples at the target node. Figure 2.5(b) and 2.5(c) showcase that the more model knowledge is accumulated in the barycenter computed offline, the higher image quality is achieved at Node 0. As expected, more model knowledge can help new edge nodes in training higher-quality generative models. In both figures, the Barycentric Fast Adaptation outperforms Transferring GANs.

2.6.5 Impact of the Number of Data Samples at Node 0

Figure 2.5(e) further illustrates the convergence across different number of data samples at the target node on CIFAR10 dataset. As expected, the FID score gap between Barycentric Fast Adaptation and *Edge-Only* method decreases as the number of data samples at the target node increases, simply because the empirical distribution becomes more ‘accurate’. In particular, the significant gap of FID scores between *Edge-Only* and the *Barycentric Fast Adaptation* approaches in the initial stages indicates that the barycenter found via offline training and adopted as the model initialization for fast adaptation, is indeed close to the underlying model at the target node, hence enabling faster and more accurate edge learning than *Edge-Only*.

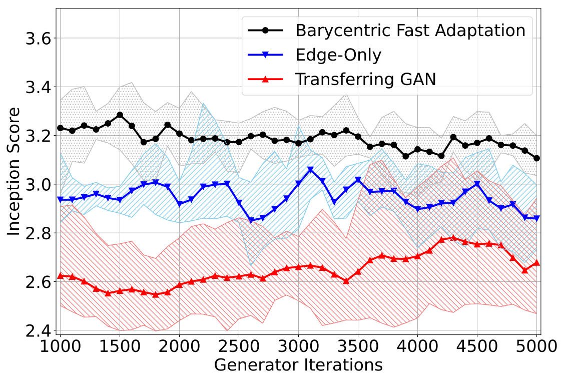
2.6.6 Impact of Wasserstein Ball Radii

The Wasserstein ball radius η_k for pre-trained model k represents the relevance (hence utility) of the knowledge transfer which is intimately related to the capability to generalize beyond the pre-trained generative models, and the smaller it is, the more informative the corresponding Wasserstein ball is. Hence, larger weights $\lambda_k = 1/\eta_k$

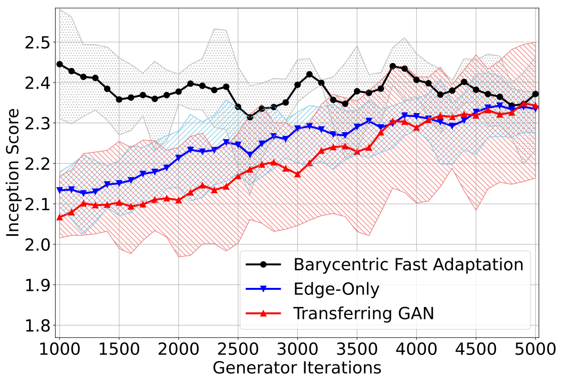
would be assigned to the nodes with higher relevance. We note that the weights are determined by the constraints and thus are fixed. Since we introduce the recursive WGAN configuration, the order of coalescence (each corresponding to a geodesic curve) may impact the final barycentric WGAN model, and hence the performance of Barycentric Fast Adaptation. To this end, we compute the coalescence of models of nodes with higher relevance at latter recursions to ensure that the final barycentric model is closer to the models of nodes with higher relevance.

2.6.7 Ternary WGAN-based Barycentric Fast Adaptation

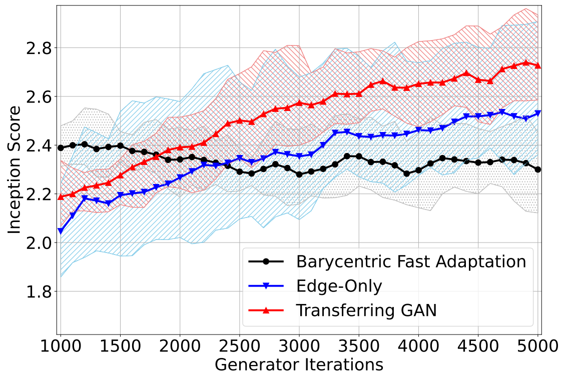
With the model initialization in the form of a full-precision barycenter model computed in offline training, we next train a ternary WGAN with 2 discriminators for the target node to compress the generative model further. In particular, we use the same split of classes as the experiment illustrated in Figure 2.5(e), and compare the image quality obtained by Ternary WGAN-based fast adaptation against both full precision counterpart and *Edge-Only*. It can be seen from the FID scores (Figure 2.5(f)), the ternary WGAN-based Barycentric Fast Adaptation results in negligible performance degradation compared to its full precision counterpart, and is still much better compared to the *Edge-Only* approach.



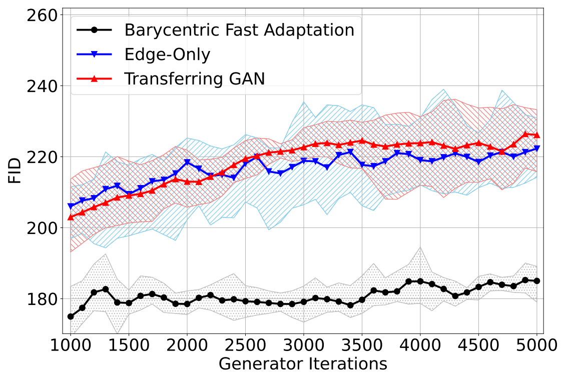
(a) IS for the Overlapping Case with 100 Data Samples at Node 0.



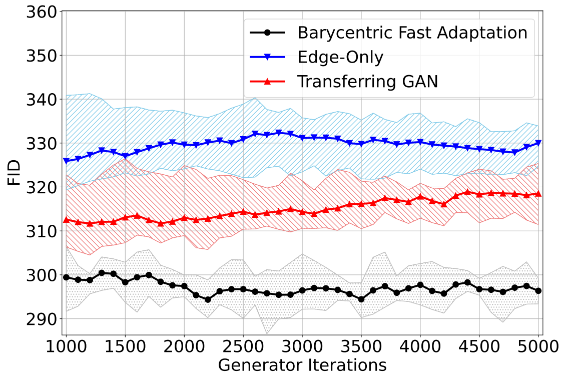
(b) IS for the Overlapping Case with 50 Data Samples at Node 0.



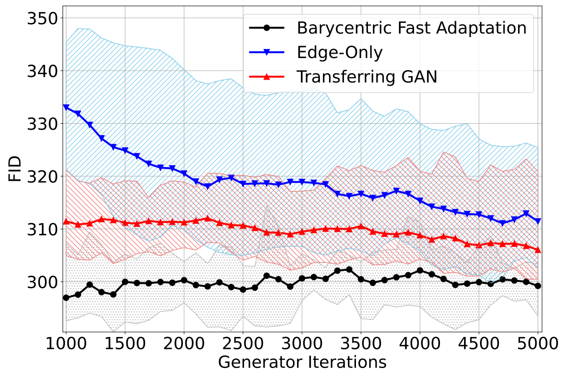
(c) IS for the Overlapping Case with 20 Data Samples at Node 0.



(d) FID Score for the Overlapping Case with 100 Data Samples at Node 0.



(e) FID Score for the Overlapping Case with 50 Data Samples at Node 0.



(f) FID Score for the Overlapping Case with 20 Data Samples at Node 0.

Figure 2.7: Performance Evaluation Comparisons of Various WGAN Model Training Techniques Using FID and Inception Scores. Note That Higher Inception Score and Lower FID Score Indicate Better Performance.

2.6.8 Performance Evaluation Using Inception Score

In addition to the FID score, we also use Inception Score (IS), another widely used metric, to signify the robustness of the performance evaluation. Each of the 3 different numerical experiments is repeated 5 times, and the performance evaluation using FID and Inception scores is illustrated in Figure 2.7. Clearly, both FID and IS evaluations corroborate the superior performance of the Barycentric Fast Adaptation algorithm, as well as the small deviation from the mean performance. The worst and best case performances of the Barycentric Fast Adaptation and its counterparts are illustrated in Table 2.1 and 2.2. *Best-Mean, Worst-Mean, Best* and *Worst* denote the best mean performance, the worst mean performance, the best performance in all 5 runs and the worst performance in all 5 runs for the corresponding metrics, respectively. Table 2.1 and 2.2 further showcase the superior performance of Barycentric Fast Adaptation in comparison to its counterparts, particularly when the number of available samples at Node 0 is limited.

An important observation herein is that both FID and IS quantify the quality and the class diversity of the generated images. Specifically, the FID score leverages another large dataset (reference dataset) (the whole dataset in our experiments) to *relatively* compute the quality and class diversity of the generated images, whereas IS does not utilize a reference dataset, i.e., IS is *absolute*. A significant implication of this difference is that the FID score of a generated dataset can be different with respect to different reference datasets, while IS will be constant. Therefore, IS cannot quantify the effects of generator overfitting as well as the FID score does. In Figure 2.7(c) and 2.7(f), only 20 data samples are used to train the WGAN generator models, and hence the final WGAN models are prone to extreme overfitting. The WGAN models might generate the same images even for different values of $\mathbf{z}$, i.e., the image diversity

Table 2.1: Performance Comparisons of Different WGAN Model Training Algorithms.

	Metric\ Experiment	Inception Score			
		Best-Mean	Worst-Mean	Best	Worst
Fig. 2.7(a), 2.7(d)	Fast Adaptation	3.28 ± 0.13	3.11 ± 0.08	3.42	2.95
	Edge-Only	3.06 ± 0.11	2.85 ± 0.18	3.33	2.67
	Transferring GANs	2.78 ± 0.32	2.55 ± 0.15	3.11	2.40
Fig. 2.7(b), 2.7(e)	Fast Adaptation	2.45 ± 0.13	2.31 ± 0.13	2.58	2.12
	Edge-Only	2.34 ± 0.08	2.13 ± 0.09	2.42	2.03
	Transferring GANs	2.35 ± 0.16	2.07 ± 0.07	2.50	1.97
Fig. 2.7(c), 2.7(f)	Fast Adaptation	2.40 ± 0.12	2.28 ± 0.16	2.55	2.12
	Edge-Only	2.53 ± 0.27	2.05 ± 0.19	2.91	1.86
	Transferring GANs	2.74 ± 0.19	2.19 ± 0.13	2.96	2.07

within every class might be very low. In accordance with the generator overfitting phenomenon, we observe that the FID scores for all 3 methods are stationary after 2000 iterations in Figure 2.7(f), whereas the IS curves continue to improve for Edge-Only and Transferring GANs in Figure 2.7(c). This indicates generator overfitting occurs in WGAN model trained with the Edge-Only and Transferring GANs methods, whereas both the IS and FID scores for the Barycentric Fast Adaptation method are stationary, indicating no generator overfitting.

Table 2.2: Performance Comparisons of Different WGAN Model Training Algorithms.

Metric\ Experiment		Frechet-Inception Distance			
		Best-Mean	Worst-Mean	Best	Worst
Fig. 2.7(a), 2.7(d)	Fast Adaptation	175 ± 7	185 ± 4	169	195
	Edge-Only	206 ± 7	222 ± 9	194	239
	Transferring GANs	203 ± 11	226 ± 8	193	236
Fig. 2.7(b), 2.7(e)	Fast Adaptation	294 ± 5	300 ± 4	287	306
	Edge-Only	326 ± 11	332 ± 7	319	341
	Transferring GANs	312 ± 8	319 ± 4	305	326
Fig. 2.7(c), 2.7(f)	Fast Adaptation	297 ± 7	302 ± 4	291	314
	Edge-Only	311 ± 13	333 ± 12	300	348
	Transferring GANs	306 ± 10	312 ± 7	300	325

2.6.9 Image Morphing via Barycentric Fast Adaptation

The proposed barycentric fast adaptation approach is useful for many applications, including image morphing [118], clustering [33], super resolution [82] and privacy-aware synthetic data generation [117] at edge nodes. To get a more concrete sense, Figure 2.8 illustrates a comparison of image morphing using three methods, namely *Barycentric Fast Adaptation*, *Transferring GANs* and *ensemble*. Observe that Trans-

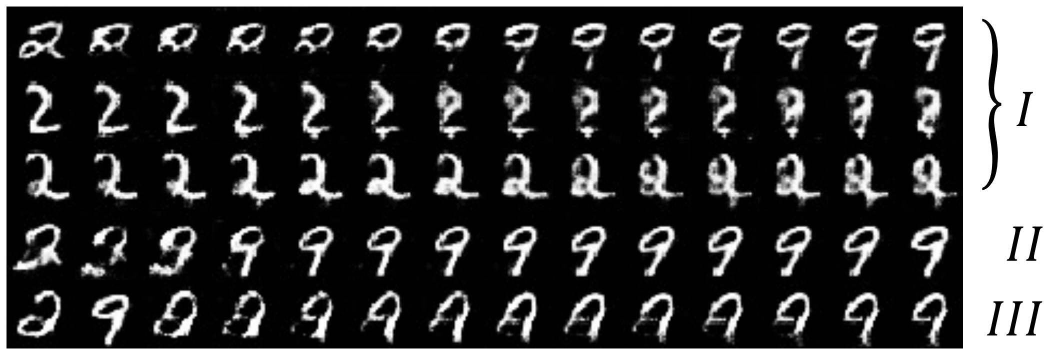
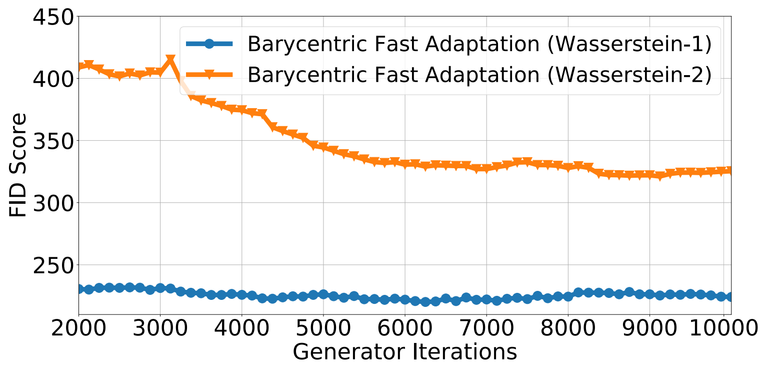


Figure 2.8: The Illustrations of Image Morphing Using 3 Different Techniques: *I*) Barycentric Fast Adaptation, *II*) Transferring GANs and *III*) Ensemble Method. 5000 Samples from Classes "2" and "9" in MNIST Are Used in Experiments and Horizontal Axis Represents Training Iterations.

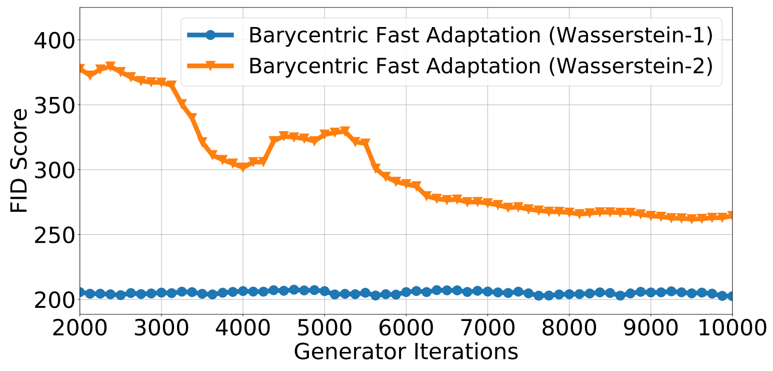
ferring GANs quickly morphs images from class "2" to class "9", but forgetting the previous knowledge. In contrast, Barycentric Fast Adaptation morphs class "2" to a barycenter model between the two classes "2" and "9," because it uses generative replay in the training, thus mitigating catastrophic forgetting. In addition, the transition from "2" to "9" is smoother in comparison to its counterparts, because barycentric transformation is shape-preserving. The ensemble method learns both classes "2" and "9" at the end, but its morphing process takes longer.

2.7 Performance Comparison of W1GAN and W2GAN-based Fast Adaptation

To compare the performance of the recursive W1GAN and W2GAN configurations, we design two numerical experiments on MNIST dataset. In the *overlapping classes* experiment setup, we split the the classes in MNIST into two equal subsets and place all the samples from each half to two different edge nodes. The edge server collects the pretrained models from both of the edge nodes and trains a barycenter W1GAN/W2GAN model. This barycenter model is then leveraged at a target edge Node 0, which encompasses only 1000 (or 100) samples from all 10 classes, by performing the fast adaptation. In the *non-overlapping classes* experiment setup,



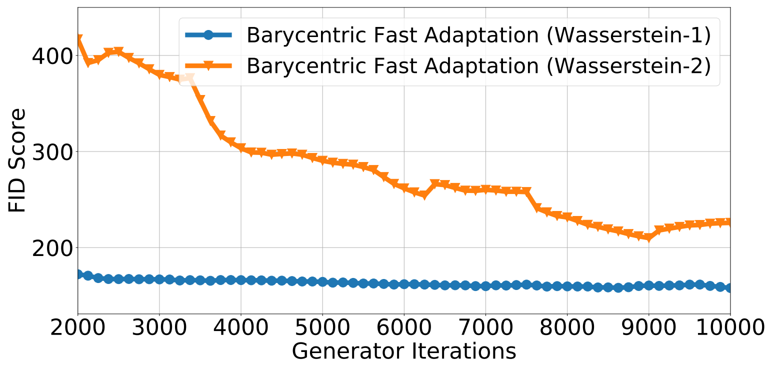
(a) Comparison of Convergence over MNIST: The Overlapping Case with 1000 Samples at Node 0.



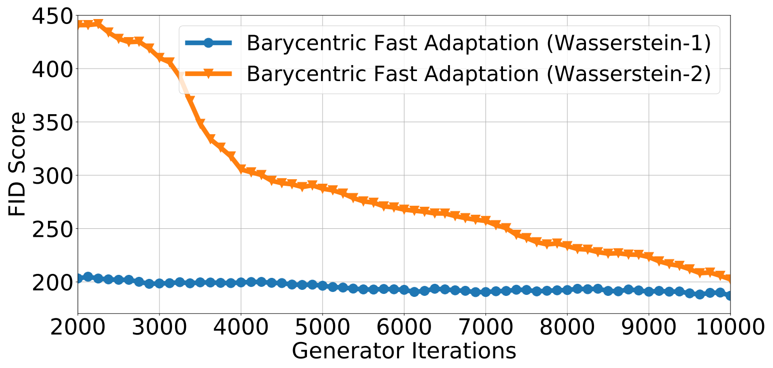
(b) Comparison of Convergence over MNIST: The Non-overlapping Case with 200 Samples at Node 0.

Figure 2.9: The Convergence Rate Comparison of the Barycentric Fast Adaptation for W_1 and W_2^2.

we split the the classes in MNIST into three subsets: 1) all the samples from first 4 classes are placed into the first edge node, 2) all the samples from the second 4 classes are placed in the second edge node, 3) and 200 (or 20) samples from the remaining 2 classes are placed on the target edge Node 0. Subsequently, the same experiment setup described in *overlapping classes* setting is used. We have used the same DNN architecture used in [79] for both recursive W1GAN and W2GAN configurations since DNNs must be convex in W2GAN setting.



(a) Comparison of Convergence over MNIST: The Overlapping Case with 100 Samples at Node 0.



(b) Comparison of Convergence over MNIST: The Non-overlapping Case with 20 Samples at Node 0.

Figure 2.10: The Convergence Rate Comparison of the Barycentric Fast Adaptation for W_1 and W_2^2.

As illustrated in Figure 2.9 and 2.10, in all experiments, the Wasserstein-1 based recursive configuration outperforms the quadratic Wasserstein-2 based recursive configuration. In particular, we observe the worse performance in *overlapping* setting, suggesting that the recursive W2GAN configuration cannot leverage the previous network knowledge as efficient as its W1GAN counterpart.

2.8 Conclusions

In this work, we propose a systematic framework for continual learning of generative models via adaptive coalescence of pre-trained models from other edge nodes. Particularly, we cast the continual learning problem as a constrained optimization problem that can be reduced to a Wasserstein-1 barycenter problem. Appealing to optimal transport theory, we characterize the geometric properties of geodesic curves therein and use displacement interpolation as the foundation to devise recursive algorithms for finding adaptive barycenters. Next, we take a two-stage approach to efficiently solve the barycenter problem, where the barycenter of the pre-trained models is first computed offline in the cloud via a "recursive" WGAN configuration based on displacement interpolation. Then, the resulting barycenter is treated as the meta-model initialization and fast adaptation is used to find the generative model using the local samples at the target edge node. A weight ternarization method, based on joint optimization of weights and threshold for quantization, is developed to compress the edge generative model further. Extensive experimental studies corroborate the efficacy of the proposed framework.

2.9 Appendix A: Analytical Results

2.9.1 *Proof of Proposition 1 for Wasserstein-1 GAN*

Proof. Let $\{\mu_k\}_{1:K}$ be any set of probability measures on a refined forming set and ν_k^* denote a continuous probability measure with no atoms, which minimizes the problem $\min_{\nu_k} W_1(\mu_k, \nu_k) + W_1(\nu_{k-1}^*, \nu_k)$ [10]. By Proposition 4, there exists multiple refined forming sets, and the proceeding proof holds true for any refined forming set induced by the original set of probability distributions. The proceeding proof utilizes the geodesic property and the existence of a barycenter in Wasserstein-1 space, for which the details can be found in [132, 10] and [81], respectively. Let the barycenter at iteration $k = 1$ be selected as $\nu_1^* = \mu_1$ and suppose that $\alpha \notin \mathcal{B}_{\mathcal{R}}$ is a distribution satisfying

$$\min_{\nu_2} W_1(\mu_2, \nu_2) + W_1(\nu_1^*, \nu_2) = W_1(\mu_2, \alpha) + W_1(\mu_1, \alpha) \tag{2.34}$$

at recursion $k = 2$. Note that if $\alpha \notin \mathcal{B}_{\mathcal{R}}$, α cannot reside on the geodesic curve $g_t(\mu_1, \mu_2)_{0 \le t \le 1}$ since $g_t(\mu_1, \mu_2)_{0 \le t \le 1} \in \mathcal{B}_{\mathcal{R}}$. By considering any distribution β_2 which resides on geodesic curve $g_t(\mu_1, \mu_2)$, we can also show that:

$$\begin{aligned} W_1(\mu_1, \beta_2) + W_1(\mu_2, \beta_2) &= W_1(\mu_1, \beta_2) + W_1(\beta_2, \mu_2) \\ &= W_1(\mu_1, \mu_2) < W_1(\mu_1, \alpha) + W_1(\alpha, \mu_2) \\ &= \min_{\nu_2} W_1(\mu_2, \nu_2) + W_1(\nu_1^*, \nu_2), \end{aligned} \tag{2.35}$$

indicating that β attains a lower cost than the minimizer ν_2^*, which is a contradiction, indicating that ν_2^* must reside in $\mathcal{B}_{\mathcal{R}}$. Similarly, ν_3^* must also reside in $\mathcal{B}_{\mathcal{R}}$:

$$\begin{aligned} W_1(\mu_3, \beta_3) + W_1(\nu_2^*, \beta_3) &= W_1(\mu_3, \beta_3) + W_1(\beta_3, \nu_2^*) \\ &= W_1(\mu_3, \nu_2^*) < W_1(\mu_3, \alpha) + W_1(\alpha, \nu_2^*). \end{aligned} \tag{2.36}$$

By induction, $\beta_k \in \mathcal{B}_{\mathcal{R}}$ attains a lower cost compared with $\alpha \notin \mathcal{B}_{\mathcal{R}}$ at the kth iteration:

$$\begin{aligned} W_1(\mu_k, \beta_k) + W_1(\nu_{k-1}^*, \beta_k) &= W_1(\mu_k, \beta_k) + W_1(\beta_k, \nu_{k-1}^*) \\ &= W_1(\mu_k, \nu_{k-1}^*) < W_1(\mu_k, \alpha) + W_1(\alpha, \nu_{k-1}^*). \end{aligned} \tag{2.37}$$

Hence, $\nu_k^* = \beta_k \in \mathcal{B}_{\mathcal{R}}, \forall k$. Consequently, all barycenters to at each iteration must reside in the baryregion $\mathcal{B}_{\mathcal{R}}$.

Similarly, we can show that for stage II the following holds:

$$\begin{aligned} W_1(\mu_0,\beta_K) + W_1(\nu_K^*, \beta_K) &= W_1(\mu_0, \beta_K) + W_1(\beta_K, \nu_K^*) \\ &= W_1(\mu_0, \nu_K^*) < W_1(\mu_0, \alpha) + W_1(\alpha, \nu_K^*). \end{aligned} \tag{2.38}$$

Consequently, ν^* also resides in $\mathcal{B}_{\mathcal{R}}$, which completes the proof. $\square$

2.9.2 Remark 1 for Quadratic Wasserstein-2 Cost Function

For ease of exposition, we herein restate the seminal result by [2].

Proposition 2. *[2] The barycenter of $\{(\mu_k, \lambda_k)\}_k$, i.e., $\arg\min_{\nu} \sum_k^K \lambda_k W_2^2(\mu_k, \nu)$, constitutes a unique solution ν^* if $\{\mu_k\}_{1:K}$ vanishes on small sets. The unique solution is characterized as $\nu^* = \nabla\phi_k \sharp \mu_k$, where ϕ_k is a convex potential defined in terms of the Kantorovich potentials (*3.5 *in [2]).*

Corollary 1. *If $K = 2$, the set of barycenters ν_t^* is characterized as:*

$$\nu_t^* = \arg\min_{\nu} tW_2^2(\mu_1, \nu) + (1-t)W_2^2(\mu_2, \nu) = (t\mathrm{id} + (1-t)\nabla\phi^*)\sharp\mu_1, \tag{2.39}$$

where id *is the identity map and $\nabla\phi^*$ is the conjugate Brenier's map between μ_0 and μ_1.*

Corollary 1 implies that v_t^* is the geodesic curve between μ_1 and μ_2. Hence, the solution to the W_2^2 barycenter problem with $K = 2$ and fixed weight pair $(t, 1-t)$ always resides on the geodesic curve between μ_1 and μ_2 [94, 2].

Proof. (Displacement Interpolation for Adaptive Barycenters with W_2^2 Cost Function). Let $\{\mu_k\}_{1:K}$ be a set of probability measures on a refined forming set, which vanish on small sets, and ν_k^* denote a continuous probability measure with no atoms, which minimizes the problem $\min_{\nu_k} W_1(\mu_k, \nu_k) + W_1(\nu_{k-1}^*, \nu_k)$ [10]. The following proof builds upon Corollary 1. We consider the following barycenter sequence generated by the recursive W2GAN configuration:

$$\mathcal{S} = \{\nu_1^* = \mu_1, \nu_2^* = \arg\min_{\nu} tW_2^2(\mu_2, \nu) + (1-t)W_2^2(\nu_1^*, \nu),$$
$$\ldots, \nu_K^* = \arg\min_{\nu} tW_2^2(\mu_K, \nu) + (1-t)W_2^2(\nu_{K-1}^*, \nu)\}.$$

For ease of exposition we assign μ_1 as the unique barycenter in the first recursion, i.e., $\nu_1^* = \mu_1 \in \mathcal{B}_{\mathcal{R}}$. In iteration 2, the new barycenter can be stated in terms of the previous barycenter as

$$\begin{aligned}\nu_2^*(t, 1-t) &= \arg\min_{\nu} tW_2^2(\mu_2, \nu) + (1-t)W_2^2(\nu_1^*, \nu) \\ &= ((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_1^*. \end{aligned} \tag{2.40}$$

We note that $\nu_2^*(t, 1-t) = ((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_1^*$ defines a geodesic between μ_2 and ν_1^*, and hence $\nu_2^*(t, 1-t) \in \mathcal{B}_{\mathcal{R}}$ by definition. Further, the barycenter, $\nu_2^*(t, 1-t)$, is unique by Proposition 2 and Corollary 1. By induction, the kth barycenter is expressed as

$$\begin{aligned}\nu_k^*(t, 1-t) &= \arg\min_{\nu} tW_2^2(\mu_k, \nu) + (1-t)W_2^2(\nu_{k-1}^*, \nu) \\ &= ((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_{k-1}^*. \end{aligned} \tag{2.41}$$

As before, $\nu_k^*(t, 1-t)((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_{k-1}^* \in \mathcal{B}_{\mathcal{R}}$ and $\nu_k^*(t, 1-t)$ is the unique barycenter for a fixed $(t, 1-t)$ pair, and hence we conclude that all the barycenters in the sequence $\mathcal{S}$ reside inside the baryregion $\mathcal{B}_{\mathcal{R}}$. Similarly, for the fast adaptation stage, the final barycenter ν^* can be shown to reside on the geodesic $((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_K^*$ between ν_K^* and $\hat{\mu}_0$:

$$\begin{aligned}\nu^*(t, 1-t) &= \arg\min_{\nu} tW_2^2(\hat{\mu}_0, \nu) + (1-t)W_2^2(\nu_K^*, \nu) \\ &= ((1-t)\mathrm{id} + t\nabla\phi^*)\sharp\nu_K^*, \end{aligned} \tag{2.42}$$

and hence $\nu^* \in \mathcal{B}_{\mathcal{R}}$, which completes the proof. $\square$

2.9.3 Bounds on the Gap Between One-shot Barycentric and Recursive Barycentric Algorithms

The W_1 barycenter problem is analytically challenging, because the geodesic curve between two distributions is not unique, which may lead to multiple barycenters at every recursion [please see Example 7.4, in [133]]. Proposition 1 shows that all of the barycenters reside inside the baryregion $\mathcal{B}_{\mathcal{R}}$, for any $\{\lambda_k\}$. It follows that $\mathcal{B}_{\mathcal{R}}$ also provides a bound on the gap between the *one-shot* and *recursive* Wasserstein barycenter problems, under the W_1 cost function. For the W_2 cost function, it is shown in [22] that the approximation gap can be driven to be 0. Proposition 3 demonstrates how to achieve this by using the proposed recursive algorithm.

Proposition 3. *Let* $\{\lambda_1, \lambda_2, \dots, \lambda_K\}$ *and* $\{\lambda_{\mu_1}, \lambda_{\mu_2}, \dots, \lambda_{\mu_K}\}$ *denote the weights of the distributions* $\{\mu_1, \mu_2, \dots, \mu_K\}$ *in* one-shot *and* recursive W_2 *barycenter problems, respectively, and let* $\{\lambda_{\nu_1^*}, \lambda_{\nu_2^*}, \dots, \lambda_{\nu_K^*}\}$ *denote the weights of the barycenters in* recursive W_2 *barycenter problem. The solutions of* one-shot *and* recursive W_2 *barycenter problems are the same if* $\lambda_{\nu_k^*} = {}^{\sum_{\ell=1}^{k}\lambda_\ell}/_{\sum_{\ell=1}^{k+1}\lambda_\ell}$ *and* $\lambda_{\mu_k} = {}^{\lambda_k}/_{\sum_{\ell=1}^{k+1}\lambda_\ell}$ *are satisfied* $\forall k \in \mathcal{K}$.

Proof. Without loss of generality, we assume $\sum_{i=1}^{K}\lambda_i = 1$. Then, from Corollary 1, we have that

$$\begin{aligned}
K = 1 \rightarrow \qquad\qquad \nu_1^* =&\mu_1;\\
\lambda_{\nu_1^*} =&\lambda_{\mu_1} = \lambda_1\\
K = 2 \rightarrow \qquad\qquad \nu_1^* =&\mu_1, \nu_2^* = (\lambda_{\nu_1^*}\text{i.d.} + \lambda_{\mu_k} T_{\mu_2}^{\nu_1^*})\sharp\nu_1^*;\\
\lambda_{\nu_1^*} =&{}^{\lambda_1}/_{\lambda_1+\lambda_2}, \lambda_{\mu_2} = {}^{\lambda_2}/_{\lambda_1+\lambda_2}\\
K = 3 \rightarrow \qquad\qquad \nu_1^* =&\mu_1, \nu_2^* = T_{\nu_2^*}^{\nu_1^*}\sharp\nu_1^* = (\lambda_{\nu_1^*}\text{i.d.} + \lambda_{\mu_k} T_{\mu_2}^{\nu_1^*})\sharp\nu_1^*;\\
\nu_3^* =&(\lambda_{\nu_2^*} T_{\nu_2^*}^{\nu_1^*} + \lambda_{\mu_3} T_{\mu_3}^{\nu_1^*})\sharp\nu_1^*\\
=&(\lambda_{\nu_2^*}\lambda_{\nu_1^*}\text{i.d.} + \lambda_{\nu_2^*}\lambda_{\mu_2} T_{\mu_2}^{\nu_1^*} + \lambda_{\mu_3} T_{\mu_3}^{\nu_1^*})\sharp\nu_1^*;\\
\lambda_{\nu_1^*} =&{}^{\lambda_1}/_{\lambda_1+\lambda_2}, \lambda_{\mu_2} = {}^{\lambda_2}/_{\lambda_1+\lambda_2},\\
\lambda_{\nu_2^*} =&{}^{\lambda_1+\lambda_2}/_{\lambda_1+\lambda_2+\lambda_3}, \lambda_{\mu_3} = {}^{\lambda_3}/_{\lambda_1+\lambda_2+\lambda_3}.
\end{aligned} \tag{2.43}$$

By induction, we have:

$$\begin{aligned} K = k \rightarrow \qquad \nu_i^* &= (\lambda_{\nu_{i-1}^*} T_{\nu_{i-1}^*}^{\nu_1^*} + \lambda_{\mu_i} T_{\mu_i}^{\nu_1^*}) \sharp \nu_1^*, \forall i = 1, \ldots, k; \\ \lambda_{\nu_i^*} &= {\textstyle\sum_{j=1}^{i} \lambda_j / \sum_{j=1}^{i+1} \lambda_j}, \\ \lambda_{\mu_{i+1}} &= {\textstyle \lambda_{i+1} / \sum_{j=1}^{i+1} \lambda_j}, \forall i = 1, \ldots, k-1. \end{aligned} \tag{2.44}$$

Then, for $K = k + 1$, we can show:

$$\begin{aligned} &K = k+1 \rightarrow \nu_{i+1}^* = (\lambda_{\nu_i^*} T_{\nu_i^*}^{\nu_1^*} + \lambda_{\mu_{i+1}} T_{\mu_{i+1}}^{\nu_1^*}) \sharp \nu_1^* \\ &= (\lambda_{\nu_i^*} (\lambda_{\nu_{i-1}^*} T_{\nu_{i-1}^*}^{\nu_1^*} + \lambda_{\mu_i} T_{\mu_i}^{\nu_1^*}) + \lambda_{\mu_{i+1}} T_{\mu_{i+1}}^{\nu_1^*}) \sharp \nu_1^* \\ &= (\lambda_{\nu_i^*} \lambda_{\nu_{i-1}^*} T_{\nu_{i-1}^*}^{\nu_1^*} + \lambda_{\nu_i^*} \lambda_{\mu_i} T_{\mu_i}^{\nu_1^*} + \lambda_{\mu_{i+1}} T_{\mu_{i+1}}^{\nu_1^*}) \sharp \nu_1^*, \forall i = 1, \ldots, k \\ &\lambda_{\nu_i^*} = {\textstyle\sum_{j=1}^{i} \lambda_j / \sum_{j=1}^{i+1} \lambda_j}, \ \lambda_{\mu_{i+1}} = {\textstyle\lambda_{i+1} / \sum_{j=1}^{i+1} \lambda_j}, \forall i = 1, \ldots, k \end{aligned} \tag{2.45}$$

which completes the proof. □

2.9.4 Refined Forming Set

The following definition identifies a more compact forming set for baryregions when they exist.

Definition 4. *(**Refined Forming Set**) Let $\{\mu_k\}_{k \in \kappa}$ be a subset of the forming set $\{\mu_k\}_{1:K}$ for a set $\kappa \subset \mathcal{K}$, and let $\mathcal{B}_{\mathcal{R}}(\kappa)$ represent the baryregion facilitated by $\{\mu_k\}_{k \in \kappa}$. The smallest subset $\{\mu_k\}_{k \in \kappa^*}$, satisfying $\mathcal{B}_{\mathcal{R}}(\kappa^*) \supseteq \mathcal{B}_{\mathcal{R}}$, is defined as the refined forming set of $\mathcal{B}_{\mathcal{R}}$.*

A refined forming set can characterize a baryregion as complete as the original forming set, but can better capture the geometric properties of the barycenter problem. In particular, a refined forming set κ^* dictates that $\{\mu_k\}_{k \in \kappa^*}$ engenders exactly the same geodesic curves as in $\mathcal{B}_{\mathcal{R}}$.

Proposition 4. *(**Non-uniqueness**) A refined forming set of $\{\mu_k\}_{1:K}$ is not necessarily unique.*

Proof. To prove Proposition 1, it suffices to construct a counter example. Consider a forming set $\{\mu_k\}_{1:4}$ with the probability measures $\mu_1 = \delta_{(0,0)}$, $\mu_2 = \delta_{(1,0)}$, $\mu_3 = \delta_{(0,1)}$, and $\mu_4 = \delta_{(1,1)}$, where $\delta_{(\mathbf{a},\mathbf{b})}$ is the delta function with value 1 at $(x, y) = (\mathbf{a}, \mathbf{b})$ and 0 otherwise. Further, let $\{\mu_k\}_{k\in\{1,4\}}$ and $\{\mu_k\}_{k\in\{2,3\}}$ be two subsets of the forming set. Then, the length of the minimal geodesic curve between μ_1 and μ_4 can be computed as:

$$\begin{aligned} W_1(\mu_1(x), \mu_4(y)) &= \inf_{\gamma\in\Pi(\mu_1,\mu_4)} \int_{\mathcal{X}\times\mathcal{Y}} d(x, y)d\gamma(x, y) \\ &= \int_{\mathcal{Y}}\int_{\mathcal{X}} d([0, 0]^T, [1, 1]^T)\delta_{([0,0]^T,[1,1]^T)}dxdy = 2. \end{aligned} \tag{2.46}$$

By recalling that there exist infinitely many minimal geodesics satisfying (2.46), we check the lengths of two other sets of geodesics that traverse through μ_2 and μ_3, respectively. First, for μ_2,

$$\begin{aligned} &W_1(\mu_1(x), \mu_4(y)) \le W_1(\mu_1(x), \mu_2(z)) + W_1(\mu_2(z), \mu_4(y)) \\ &\quad= \inf_{\gamma\in\Pi(\mu_1,\mu_2)} \int_{\mathcal{X}\times\mathcal{Z}} d(x, z)d\gamma(x, z) + \inf_{\gamma\in\Pi(\mu_2,\mu_4)} \int_{\mathcal{Z}\times\mathcal{Y}} d(z, y)d\gamma(z, y) \\ &\quad= \int_{\mathcal{Z}}\int_{\mathcal{X}} d([0, 0]^T, [1, 0]^T)\delta_{([0,0]^T,[1,0]^T)}dxdz + \int_{\mathcal{Y}}\int_{\mathcal{Z}} d([1, 0]^T, [1, 1]^T)\delta_{([1,0]^T,[1,1]^T)}dzdy \\ &\quad= 2 \le W_1(\mu_1(x), \mu_4(y)), \end{aligned} \tag{2.47}$$

based on the triangle inequality and the definition of first-type Wasserstein distance. Similarly for μ_3, we can show that

$$W_1(\mu_1(x), \mu_4(y)) \le W_1(\mu_1(x), \mu_3(z)) + W_1(\mu_3(z), \mu_4(y)) \le W_1(\mu_1(x), \mu_4(y)), \tag{2.48}$$

through the selections $\gamma(x, z) = \delta_{([0,0]^T,[0,1]^T)}$ and $\gamma(z, y) = \delta_{([0,1]^T,[1,1]^T)}$. As a result, there exists at least a single minimal geodesic between μ_1 and μ_4 passing through μ_ℓ for $\ell \in \{2, 3\}$, indicating that $\mu_2, \mu_3 \in \mathcal{R}(\{\mu_k\}_{k\in\{1,4\}})$ and $\mathcal{B}_{\mathcal{R}}(\{\mu_k\}_{k\in\{1,4\}}) \supseteq \mathcal{B}_{\mathcal{R}}$. Observing that there exists no smaller forming set than $\{\mu_k\}_{k\in\{1,4\}}$, we conclude that $\{\mu_k\}_{k\in\{1,4\}}$ is a refined forming set.

Following the same line, we can have that $\{\mu_k\}_{k\in\{2,3\}}$ is another refined forming set of $\{\mu_k\}_{1:4}$ by first showing the following three inequalities:

$$W_1(\mu_2(x), \mu_3(y)) = \int_{\mathcal{Y}}\int_{\mathcal{X}} d([1,0]^T, [0,1]^T)\delta_{([1,0]^T,[0,1]^T)}dxdy = 2, \tag{2.49}$$

$$W_1(\mu_2(x), \mu_3(y)) \le W_1(\mu_2(x), \mu_1(z)) + W_1(\mu_1(z), \mu_3(y)) \le W_1(\mu_2(x), \mu_3(y)), \tag{2.50}$$

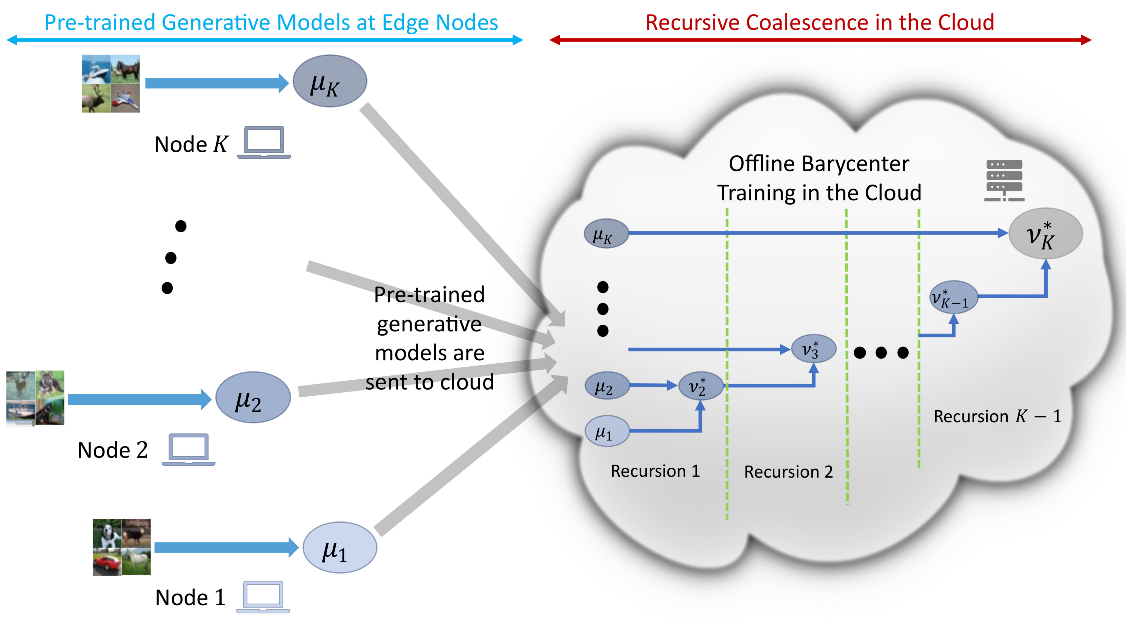
$$W_1(\mu_2(x), \mu_3(y)) \le W_1(\mu_2(x), \mu_4(z)) + W_1(\mu_4(z), \mu_3(y)) \le W_1(\mu_2(x), \mu_3(y)), \tag{2.51}$$

where the transport maps $\gamma(x,z) = \delta_{([1,0]^T,[0,0]^T)}$ and $\gamma(z,y) = \delta_{([0,0]^T,[0,1]^T)}$ for (2.50), and $\gamma(x,z) = \delta_{([1,0]^T,[1,1]^T)}$ and $\gamma(z,y) = \delta_{([1,1]^T,[0,1]^T)}$ for (2.51). Consequently, there exists at least a single minimal geodesic between μ_2 and μ_3 passing through μ_ℓ for $\ell \in \{1,4\}$, indicating that $\mu_1, \mu_4 \in \mathcal{R}(\{\mu_k\}_{k\in\{2,3\}})$ and $\mathcal{B}_{\mathcal{R}}(\{\mu_k\}_{k\in\{2,3\}}) \supseteq \mathcal{B}_{\mathcal{R}}$. Since there exists no smaller forming set than $\{\mu_k\}_{k\in\{2,3\}}$, we have that $\{\mu_k\}_{k\in\{2,3\}}$ is another refined forming set, thereby completing the proof of non-uniqueness. $\square$

2.10 Appendix B: Algorithms and Experiment Settings

2.10.1 Algorithms

For the proposed two-stage adaptive coalescence algorithm, the offline training in Stage I is done in the cloud, and the fast adaptation in Stage II is carried out at the edge server, in the same spirit as the model update of Google EDGE TPU. Particularly, as illustrated in Figure 2.11, each edge node sends its pre-trained generative model (instead of its own training dataset) to the cloud. As noted before, the amount of bandwidth required to transmit data from an edge node to cloud is also significantly reduced by transmitting only a generative model, because neural network model parameters require much smaller storage than the dataset itself.



Pre-trained Generative Models at Edge Nodes
Recursive Coalescence in the Cloud
μ_K
Node K
μ_2
Node 2
μ_1
Node 1
Pre-trained generative models are sent to cloud
Offline Barycenter Training in the Cloud
μ_K
ν_K^*
ν_{K-1}^*
ν_3^*
μ_2
ν_2^*
μ_1
Recursion 1
Recursion 2
Recursion $K-1$

Figure 2.11: Illustration of Offline Training in the Cloud: Each Edge Node Sends Pre-trained Generative Model (Instead of Datasets) to the Cloud, Based on Which the Cloud Computes Adaptive Barycenters Using the Recursive Configuration.

2.10.2 Experiment Settings

This section outlines the architecture of deep neural networks and hyper-parameters used in the experiments.

Network architectures deployed in the experiments

Figures 2.12, 2.13 and 2.14 depict the details of the DNN architecture used in our experiments; the shapes for convolution layers follow (batch size, number of filters, kernel size, stride, padding); and the shapes for network inputs follow (batch size, number of channels, heights, widths).

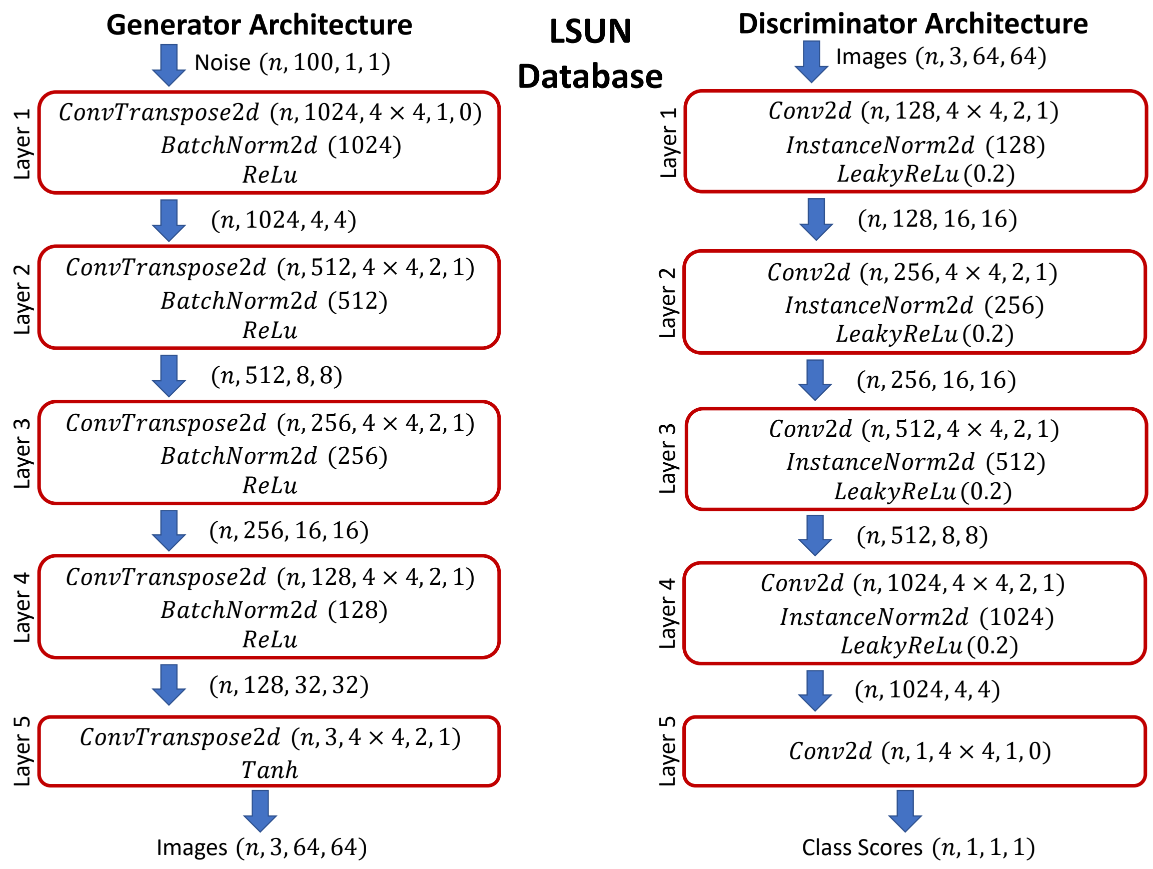


Figure 2.12: The DNN Architecture Used in Experiments for LSUN Dataset.

Hyper-parameters used in the experiments

All experiments are conducted in PyTorch on a server with RTX 2080 Ti and 64GB of memory. The selection of most parameter values, e.g., the number of generator iterations, batch size, optimizer, gradient penalty factor, and the number of discriminator iterations per generator iterations, follows [16, 59, 147]. For other parameters, we select the values giving the best performance via trial-and-error. In Table 2.4 and 2.5 all hyper-parameters are listed. We have considered different ranges of values for different parameters. The number of generator iterations (fast adaptation) ranges from 800 up to 100000. For better illustration, the figures depict only the iterations

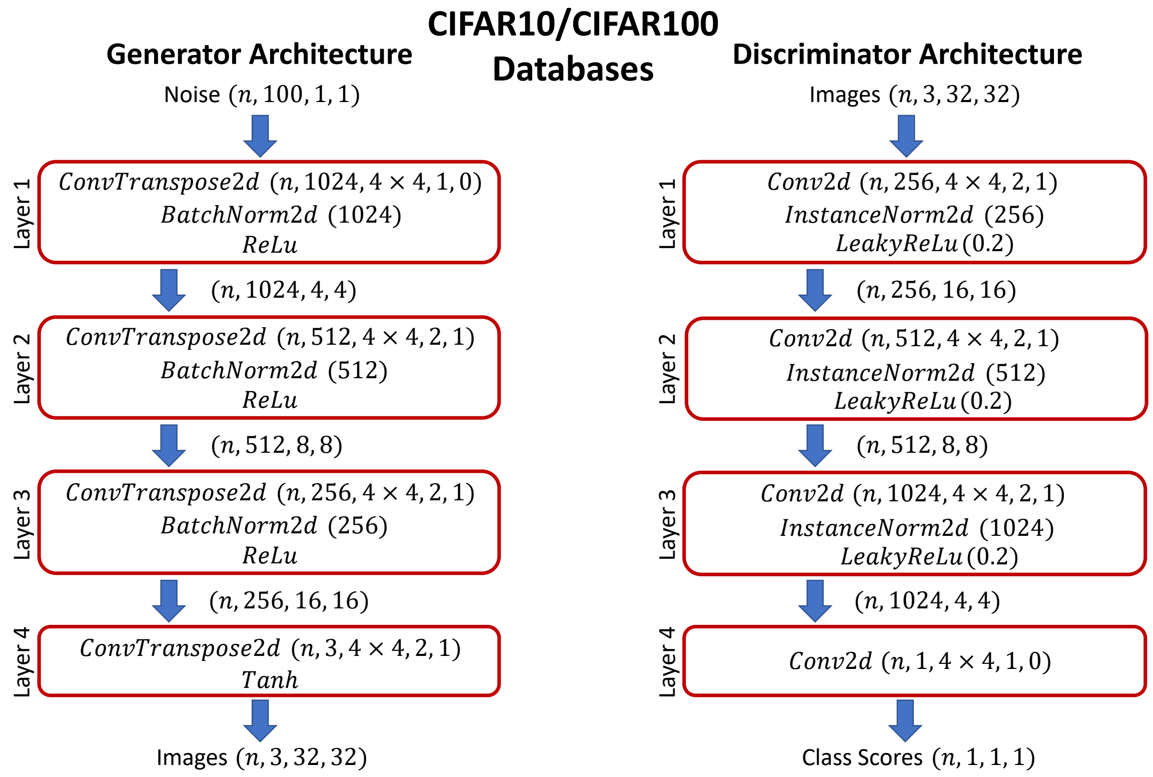


Figure 2.13: The DNN Architecture Used in Experiments for CIFAR10 and CIFAR100 Datasets.

until satisfactory image quality is achieved. For the number of samples at the target edge node, $500 \sim 10000$ samples in CIFAR10, $20 \sim 500$ samples in MNIST and $500 \sim 1000$ samples in LSUN and CIFAR100 are used. Each experiment is smoothed via a moving average filter for better visualization. More details and instructions to modify the hyper-parameters are available in the accompanying code, which will be publicly available on GitHub once the review process is over.

Table 2.3: Legend for Parameters Illustrated in Tables 2.4 and 2.5.

Parameter Name	Abbreviation
Number of Generator Iterations (Offline Training)	A
Number of Generator Iterations (Fast-Adaptation)	B
Batch Size	C
Optimizer Parameters	D
Number of Discriminator Iterations per Generator Iteration	E
Gradient-Penalty Factor	F
Number of Samples in Node 0	G
Number of Training Samples	H
Average Duration of Pretraining/Transfering per Iteration	I
Average Duration of Trainining Barycenter per Iteration	J

Table 2.4: List of Hyper-parameters Used in All Experiments with CIFAR10 Database.

	CIFAR10					
Ab.\Fig.	2.5(e)	2.20(a)	2.5(c)	2.7(a), 2.7(d)	2.7(b), 2.7(e)	2.7(c), 2.7(f)
A	100000	100000	5000	5000	5000	5000
B	5000	2000	3000	5000	5000	5000
C	64	64	64	64	64	64
D	Adam($\beta_1 = 0.5, \beta_2 = 0.999$, l.r.$= 0.0001$)					
E	5	5	5	5	5	5
F	10	10	10	10	10	10
G	Varies	1000	1000	100	50	20
$\{\lambda_k\}$	Equal	Equal	Equal	Equal	Equal	Equal
H	50000	50000	50000	50000	50000	50000
I	0.755 seconds			0.756 seconds		
J	1.480 seconds			1.482 seconds		

Table 2.5: List of Hyper-parameters Used in All Experiments with CIFAR100, MNIST and LSUN Databases.

	CIFAR100							
Ab.\Fig.	2.18(b)	2.17(b)	2.19(a)	2.21(a)	2.21(b)	2.22(b)	2.22(a)	2.5(d)
A	10000	10000	10000	10000	5000	5000	10000	10000
B	10000	10000	10000	10000	5000	5000	10000	7000
C	64	64	64	64	64	64	64	64
D				Adam($\beta_1 = 0.5, \beta_2 = 0.999$, l.r.$= 0.0001$)				
E	5	5	5	5	5	5	5	5
F	10	10	10	10	10	10	10	10
G	1000	Varies	500	500	1000	1000	5000	400
$\{\lambda_k\}$	Equal	Equal	Equal	Varies	Equal	Equal	Equal	Equal
H	50000	50000	50000	50000	50000	50000	50000	50000
I		0.755 seconds			0.756 seconds			
J		1.480 seconds			1.482 seconds			

	MNIST					LSUN		
Ab.\Fig.	2.16(a)	2.16(b)	2.5(a)	2.19(b)	2.5(b)	2.18(a)	2.5(f)	2.17(a)
A	1000	1000	3000	1000	1000	10000	10000	10000
B	800	800	1000	800	800	10000	10000	5000
C	256	256	256	256	256	64	64	64
D				Adam($\beta_1 = 0.5, \beta_2 = 0.999$, l.r.$= 0.0001$)				
E	5	5	5	5	5	5	5	5
F	10	10	10	10	10	10	10	10
G	Varies	Varies	8	100	20	1000	1000	Varies
$\{\lambda_k\}$	Equal	Equal	Equal	Equal	Equal	Equal	Equal	Equal
H	60000	60000	60000	60000	60000	100000	100000	100000
I			0.535 seconds				1.055 seconds	
J			1.020 seconds				2.125 seconds	

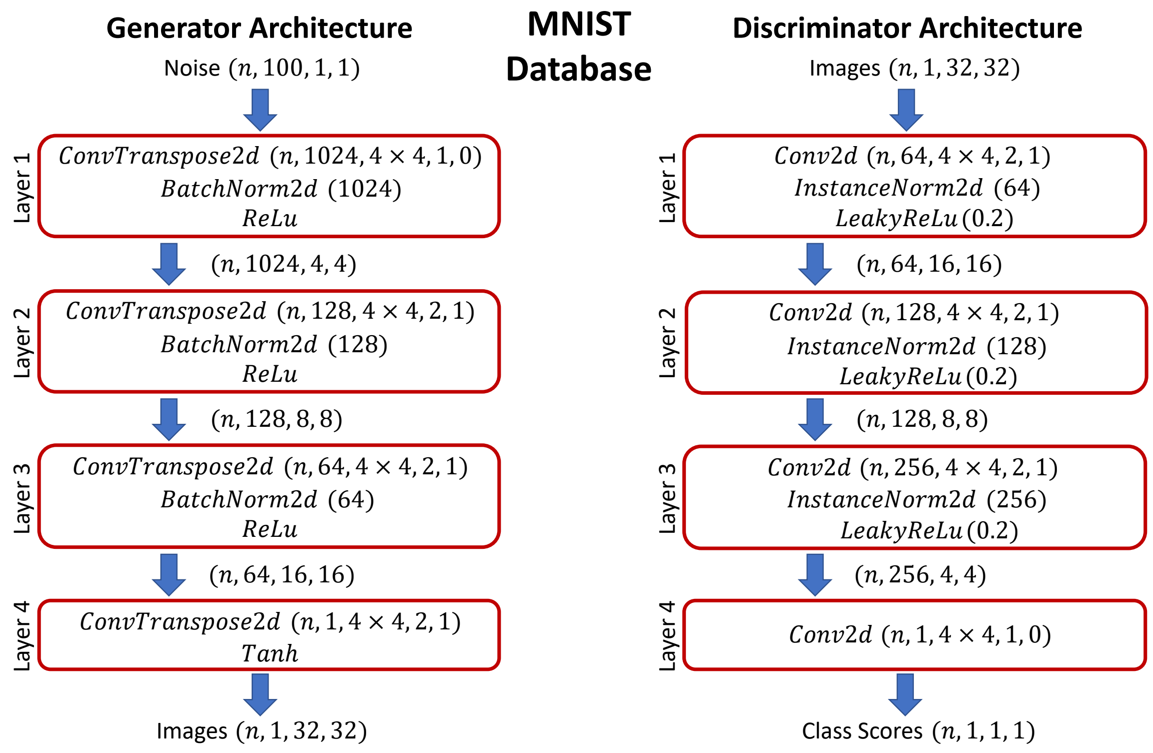


Figure 2.14: The DNN Architecture Used in Experiments for MNIST Dataset.

2.11 Appendix C: Experiments and Further Discussion

2.11.1 Evaluation Metrics

An overview of FID score

Quantifying the quality of images is an important problem for performance comparison in the literature on GANs. A variety of metrics have been proposed in the literature to quantify image quality with the consideration of over-fitting and mode collapse issues. This chapter adopts the FID score [65], which has been shown to be able to accurately evaluate image quality and over-fitting, independent of the number of classes. Since most of the datasets considered in this chapter (CIFAR10, LSUN and

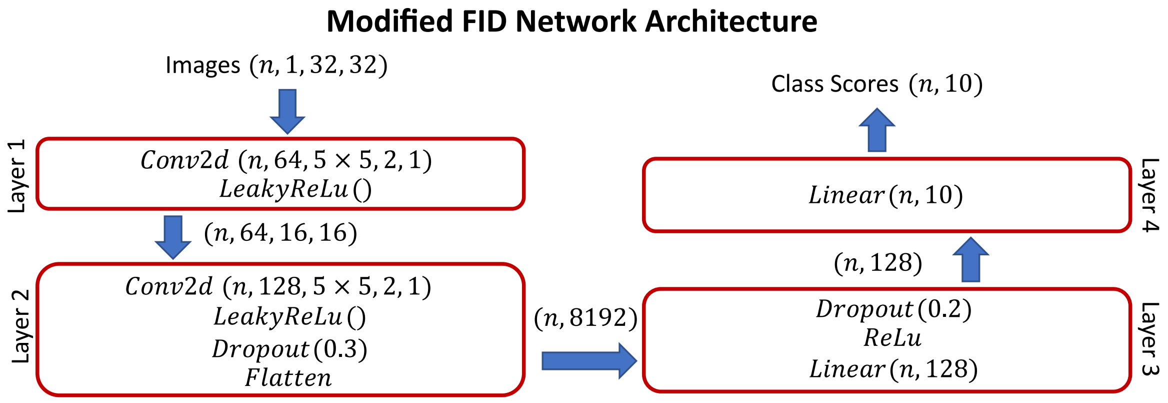


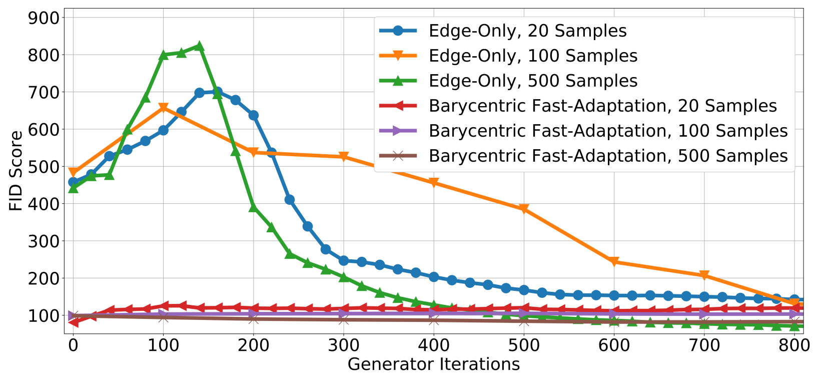
Figure 2.15: The DNN Architecture Used for Extracting Features in MNIST Images and Computing the Modified FID Scores.

MNIST) only contain 10 classes and they are further split into subsets, using a metric independent of classes is essential for our study, and the metrics highly dependent on the number of classes, e.g., Inception score (IS), may not be appropriate here.

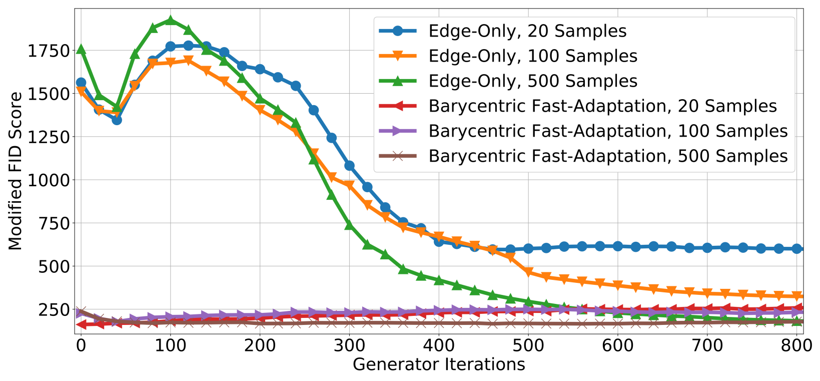
Similar to IS, a pre-trained ‘Inception’ network is utilized to extract useful features for obtaining the FID score, such that the features of real and fake images can then be used to compute correlations between these images so as to evaluate the quality of images. A perfect score of 1 can be obtained only if the features of both real and fake datasets are the same, i.e., fake images span every image in the real datasets. Consequently, if a generative model is trained only on a subset of the real-world dataset, the model would over-fit the corresponding subset and does not capture the features of the remaining real samples, thus yielding a bad FID score.

Modified FID score for MNIST dataset

Since the ‘Inception’ network is pre-trained on ‘ILSVRC 2012’ dataset, both IS and FID scores are most suitable for RGB images (e.g., CIFAR), which however cannot accurately capture the valuable features in MNIST images, simply because ‘ILSVRC 2012’ dataset does not contain MNIST classes.



(a) Evolution of Image Quality on MNIST Using FID Score Under Different Number of Samples at Target Edge Node.

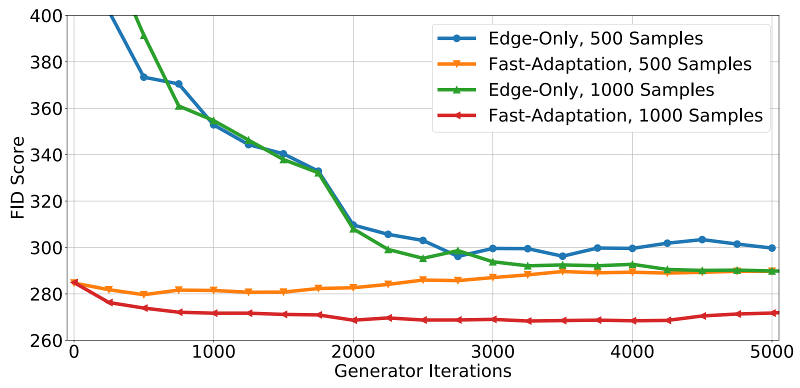


(b) Evolution of Image Quality on MNIST Using Modified FID Score Under Different Number of Samples at Target Edge Node.

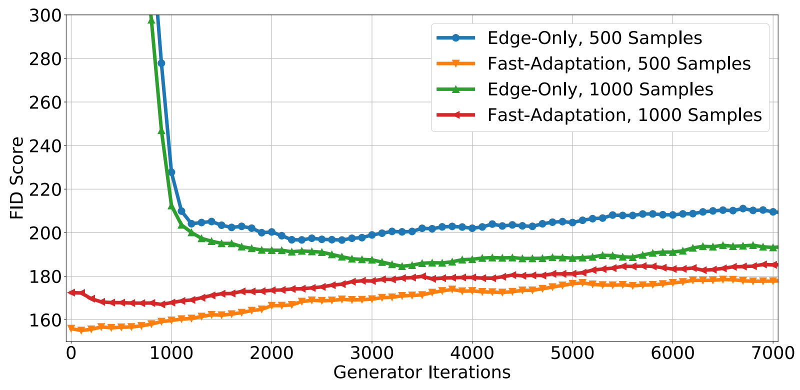
Figure 2.16: Image Quality Performance of Two Stage Adaptive Coalescence Algorithm in Various Scenarios.

To resolve these issues, we particularly train a new neural network to extract useful features for MNIST dataset. The network architecture of the corresponding DNN is shown in Figure 2.15. Fully trained network achieves an accuracy rate of 99.23% for classifying the images in MNIST. Though the corresponding architecture is much simpler in comparison to the 'Inception' network, the high classification accuracy indicates that the network can extract the most valuable features in MNIST dataset.

To further demonstrate the difference between FID and modified FID scores, we evaluate the results of Experiment 4 using both approaches, as shown in Figure 2.16(a)



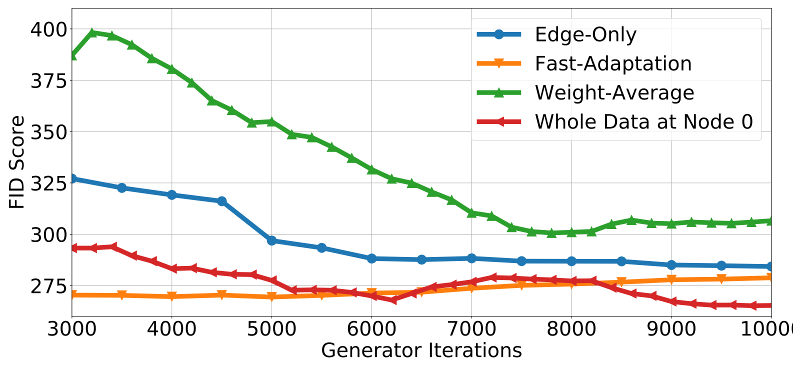
(a) Evolution of Image Quality on LSUN Using FID Score Under Different Number of Samples at Target Edge Node.



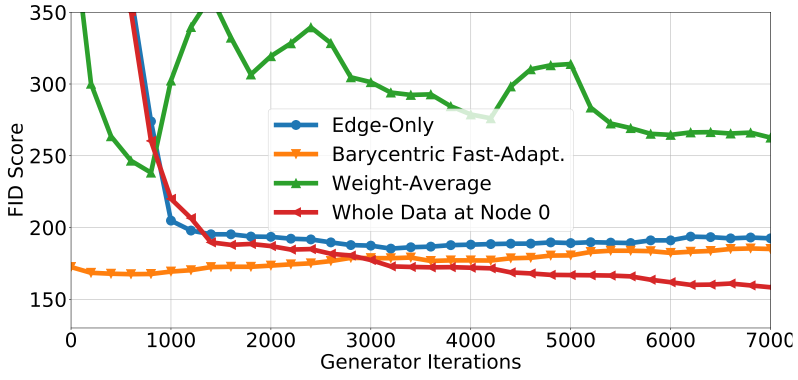
(b) Evolution of Image Quality on CIFAR100 Using FID Score Under Different Number of Samples at Target Edge Node.

Figure 2.17: Image Quality Performance of Barycentric Fast Adaptation in Various Scenarios.

and 2.16(b), respectively. It can be seen that upon convergence, the FID scores for the 'Edge-Only' with different number of samples are similar, whereas the modified FID scores under different cases are more distinct from each other and correctly reflect the learning performance. Besides, 'Edge-Only' with 20 samples incorrectly performs better than 'Edge-Only' with 100 samples in the FID score, while 'Edge-Only' with 20 and 100 samples perform as expected with the modified FID score. Hence, the modified FID score can better capture the image features compared with the FID



(a) Convergence of Barycentric Fast-adaptation Compared to 3 Different Baselines: Case for LSUN.



(b) Convergence of Barycentric Fast-adaptation Compared to 3 Different Baselines: Case for CIFAR100.

Figure 2.18: Image Quality Performance of Barycentric Fast Adaptation in Various Scenarios.

score, and is a more suitable metric to evaluate the image quality in experiments with MNIST.

2.11.2 Additional Experiments on MNIST, CIFAR10, CIFAR100 and LSUN

Tabulated FID scores

We first present the best achieved FID scores for all different experiments illustrated in Section 2.6. Table 2.6 showcases that Barycentric Fast Adaptation achieves better image quality than its counterparts at the end of training. This is due to the fact that the Barycentric Fast Adaptation leverages barycentric model from offline training, while counterpart techniques relies on the few samples at Node 0.

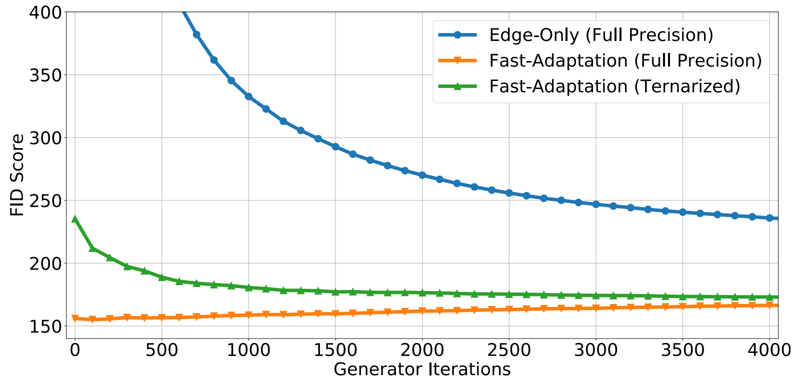
Fine-tuning via fast adaptation

We investigate the convergence and the image quality of various training scenarios on MNIST, CIFAR10, CIFAR100 and LSUN datasets. To demonstrate the improvements by using the proposed framework based on *Barycentric Fast-Adaptation*, we conduct extensive experiments and compare performance with 3 additional baselines: 1) *Edge-Only*: only local dataset with few samples at the target edge node is used in WGAN training; 2) *Weight-Average*: an initial model for training a WGAN model at the target edge node is computed by weight-averaging pre-trained models across other edge nodes, and then *Barycentric Fast-Adaptation* is used to train a WGAN model; 3) *Whole Data at Node* 0: the whole dataset available across all edge nodes is used in WGAN training.

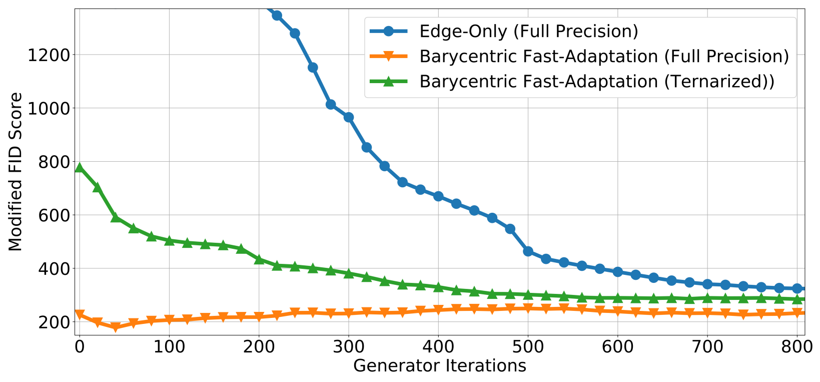
As illustrated in Figure 2.16(b), 2.17(a) and 2.17(b), *Barycentric Fast Adaptation* outperforms *Edge-Only* in all scenarios with different sizes of the training set. In particular, the significant gap of modified FID scores between two approaches in the initial stages indicates that the barycenter found via offline training and adopted as

Table 2.6: The Final FID Scores Achieved for Distinct Experiment Setups in Figure 2.5.

FID Scores for	Edge-Only	Ensemble	Trans. GAN	Bary. Fast Adapt.
MNIST/Overlap.	563.66	N/A	550.42	266.51
CIFAR10/Overlap.	184.41	N/A	179.80	151.54
CIFAR100/Overlap.	190.13	N/A	187.16	182.37
LSUN/Overlap.	291.75	N/A	N/A	279.87
MNIST/Non-overlap.	879.02	832.06	833.23	613.81



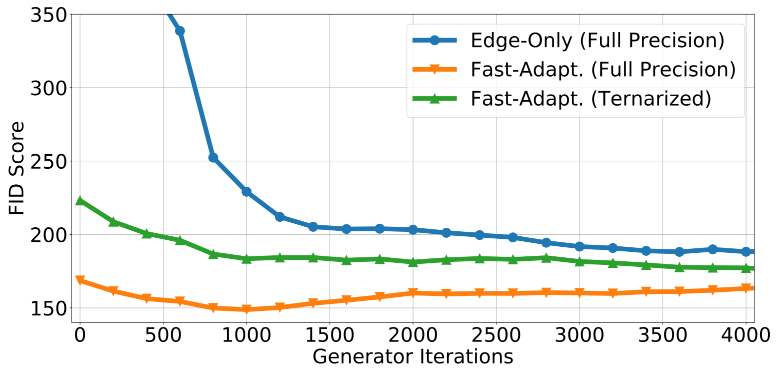
(a) Convergence of Ternarized and Full Precision Barycentric Fast Adaptation Methods on CIFAR100.



(b) Convergence of Image Quality of Ternarized and Full Precision Barycentric Fast Adaptation Techniques on MNIST.

Figure 2.19: Image Quality Performance of Two Stage Adaptive Coalescence Algorithm in Various Scenarios.

the model initialization for fast adaptation, is indeed close to the underlying model at the target edge node, hence enabling faster and more accurate edge learning than *Edge-Only*. Moreover, upon convergence, the *Barycentric Fast Adaptation* approach achieves a better FID score (hence better image quality) than *Edge-Only*, because the former converges to a barycenter residing between the coalesced model computed offline and the empirical model at target edge node. We further notice that *Barycentric Fast Adaptation* noticeably addresses catastrophic forgetting problem apparent



(a) Convergence of Ternarized and Full Precision Fast Adaptation Methods on CIFAR10.

Figure 2.20: Image Quality Performance of Two Stage Adaptive Coalescence Algorithm in Various Scenarios.

in *Transferring GANs* and *Edge-Only*, but cannot eliminate it completely in Figure 2.17 and 2.18. As it is illustrated in Figure 2.21, catastrophic forgetting can be eliminated by selecting appropriate η_k values. As expected, the modified FID score gap between two approaches decreases as the number of data samples at the target node increases, simply because the empirical distribution becomes more 'accurate'.

Figures 2.18(a) and 2.18(b) compare the performance of *Barycentric Fast-Adaptation* on LSUN and CIFAR100 with additional 2 baselines *Weight-Average* and *Whole Data at Node* 0. Again, *Barycentric Fast-Adaptation* outperforms all baselines in the initial stages of training, but as expected, *Whole Data at Node* 0 achieves the best FID score upon convergence as it utilizes whole reference dataset. Unsurprisingly, *Weight-Average* performs poorly since weight averaging does not constitute a shape-preserving transformation of pre-trained models, while *Barycentric Fast-Adaptation* can by utilizing displacement interpolation in the Wasserstein space.

Ternary WGAN based fast adaptation

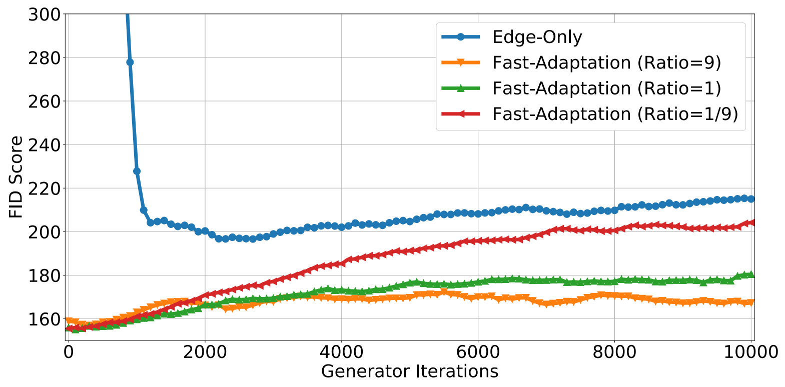
Following the same spirit of the experiment for LSUN, we compare the image quality obtained by ternary WGAN-based fast adaptation against both full precision counterpart and *Edge-Only* for CIFAR100, CIFAR10 and MNIST datasets. It can be seen from the modified FID scores (Figure 2.19(b), 2.20(a) and 2.19(a)) that the ternary WGAN-based fast adaptation facilitates image quality in between its full precision counterpart and the *Edge-Only* approach, which indicates that the ternary WGAN-based fast adaptation provides negligible performance degradation compared to the full precision method.

2.11.3 Additional Experiment Settings

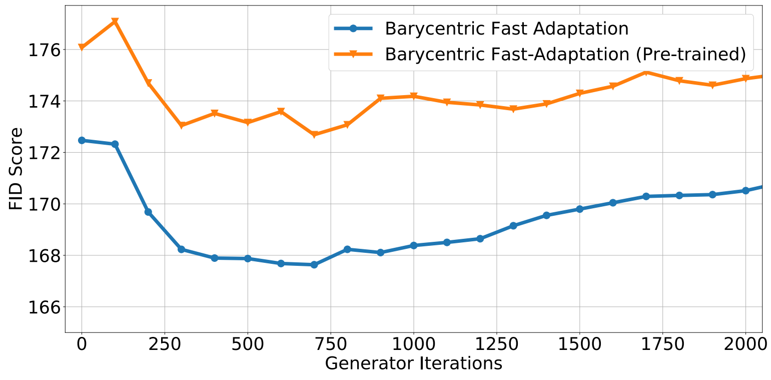
This subsection features additional experiment setups, which are not considered as primary use cases for the proposed *Barycentric Fast Adaptation*, but might provide useful insights regarding the algorithm.

The impact of Wasserstein ball radii

To demonstrate the impact of the Wasserstein ball radii, we design an experiment with different radius values in the fast adaptation stage. The CIFAR100 dataset is equally split to 2 edge nodes and an offline barycenter is computed with equal Wasserstein ball radii. We trained 3 different models for fast adaptation with varying weights $\lambda_k = {}^1\!/\!_{\eta_k}$. As noted in Section 1, radius η_k represents the relevance (hence utility) of the knowledge transfer, and the smaller it is, the more informative the corresponding Wasserstein ball is. As illustrated in Figure 2.21(a), the performance of *Barycentric Fast Adaptation* improves as the weight λ_k increases, because the knowledge transfer from the offline barycenter is more informative. Consequently, the fast adaptation benefits from the coalesced model more, which mitigates the effects of catastrophic



(a) Evolution of Image Quality on CIFAR100 for Different Wasserstein Ball Radii Values.



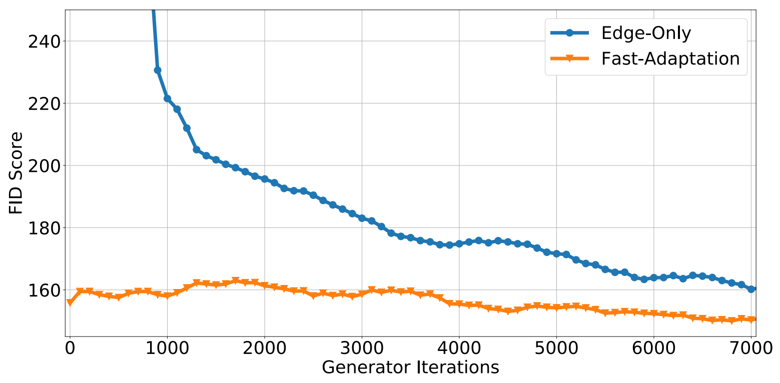
(b) Evolution of the Quality of Images Generated by Fast Adaptation Using Pre-trained Model or Using Few Samples at Target Node.

Figure 2.21: Image Quality Performance of Two Stage Adaptive Coalescence Algorithm in Various Scenarios. CIFAR100 Dataset is Used in All Experiments.

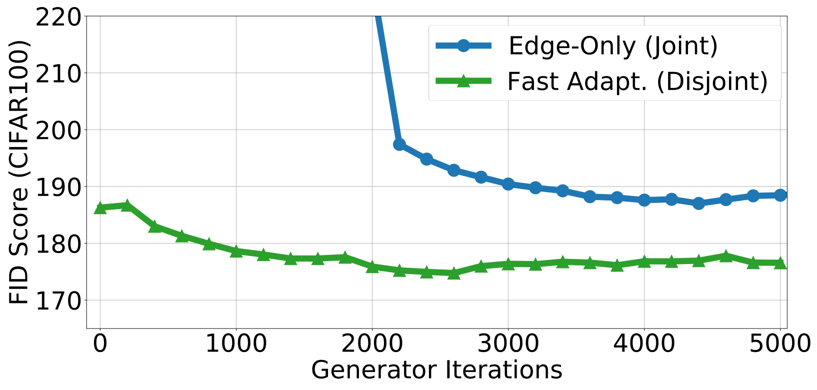
forgetting, leading to better image quality.

Pre-training WGAN at target edge node

In this experiment, we explore the possible effects of using a pre-trained WGAN model, which is trained using the local samples at the target edge node, instead of using the samples at target edge node as in the proposed barycentric fast adaptation phase. Specifically, the CIFAR100 dataset is split into 2 equal size subsets and each



(a) Evolution of the Quality of Images Generated by Fast Adaptation for Different Number of Data Samples with Same Data Classes at Edge Nodes.



(b) Evolution of the Quality of Images Generated by Fast Adaptation for Disjoint Dataset at Target Node.

Figure 2.22: Image Quality Performance of Two Stage Adaptive Coalescence Algorithm in Various Scenarios. CIFAR100 Dataset Is Used in All Experiments.

subset is placed on one of two edge nodes, based on which an offline barycenter model is trained. In addition, another WGAN model is pre-trained using local samples at the target edge node as in *Edge-Only*. Subsequently, model fusion is applied using the offline barycenter model and the pre-trained WGAN model at the target edge node. Figure 2.21(b) demonstrates that the performance of this approach is negatively impacted, when compared to the proposed Barycentric Fast Adaptation.

Disjoint classes at the target edge node

In this experiment, we investigate the performance degradation of fast adaptation when the datasets in the source edge nodes and at the target edge node do not have data samples from the same class. To this end, two disjoint subsets from CIFAR100, 50 classes and 40 classes, are placed on 2 edge nodes, from which an offline barycenter is trained. A subset of samples from the remaining 10 classes are placed on the target edge node. Figure 2.22(b) shows the performance benefit of *Barycentric Fast Adaptation* compared to *Edge-Only.* As expected, *Barycentric Fast Adaptation* with disjoint classes yield less knowledge transfer from offline training to fast adaptation (yet they still share common features), but perform better than its *Edge-Only* counterpart.

The impact of sample sizes

Next, we explore if the offline barycenter model offers any benefit to fast adaptation when all the edge nodes possess the same dataset classes, but with different sample sizes. For this purpose, 250, 200 and 50 disjoint samples are sampled from each class in CIFAR100 and placed at two edge nodes and target node, respectively. We here notice that the offline barycenter is now just a barycenter of two close empirical distributions, which share the same underlying distributions. Therefore, this setup is more suitable to transfer learning rather than edge learning. Nonetheless, *Barycentric Fast Adaptation* utilizes the additional samples from offline training, in the same spirit to *transfer learning* and improves FID score in comparison to *Edge-Only*, which only has access to 5000 samples (Figure 2.22(a)).

2.11.4 Additional Synthetic Images

In this section, we present more synthetic images generated using *Edge-Only, Transferring GANs, Barycentric Fast Adaptation* and *ternarized Barycentric Fast*

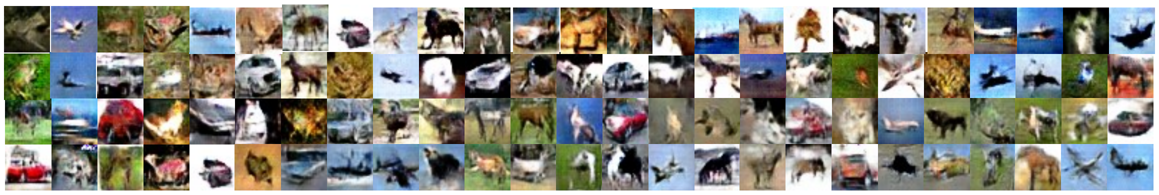


Figure 2.23: Image Samples Generated at 1000th Iteration Using Barycentric Fast Adaptation on CIFAR10 Dataset.

Adaptation techniques. Figure 2.23, 2.24 and 2.25 illustrate 100 additional images generated by *Barycentric Fast Adaptation, Transferring GANs* and *ternarized Barycentric Fast Adaptation* techniques, respectively. For Barycentric Fast Adaptation and Transferring GANs, the synthetic images are collected at iteration 1000, since both techniques attains a good FID score at early stages of training. However, Transferring GANs suffers from catastrophic forgetting in latter stages of training, while Barycentric Fast Adaptation can significantly prevent catastrophic forgetting, generating high quality synthetic images even at latter stages of training. We collected synthetic images from ternary Barycentric Fast Adaptation at iteration 5000 since as expected it takes longer for this technique to converge to a good generative model. However, it saves significant memory in comparison to full precision Barycentric Fast Adaptation at the expense of negligible performance degradation.

Finally, Figure 2.26 and 2.27 show images generated using Edge-Only at iterations 5000 and 90000 iterations, respectively. As it can be observed from the images in Figure 2.26, Edge-Only has not converged to a good GAN model yet at iteration 5000. Observe that the image quality at iteration 90000 in Figure 2.27 is significantly better, since the Edge-Only has converged to the empirical distribution at Node 0, but it is still as not good as that generated by using Barycentric Fast Adaptation.

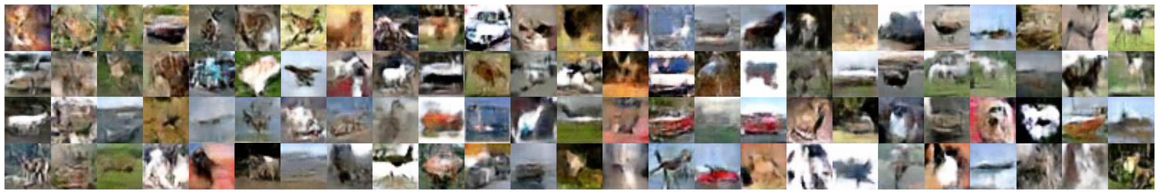


Figure 2.24: Image Samples Generated at 1000th Iteration Using Transferring GANs on CIFAR10 Dataset.

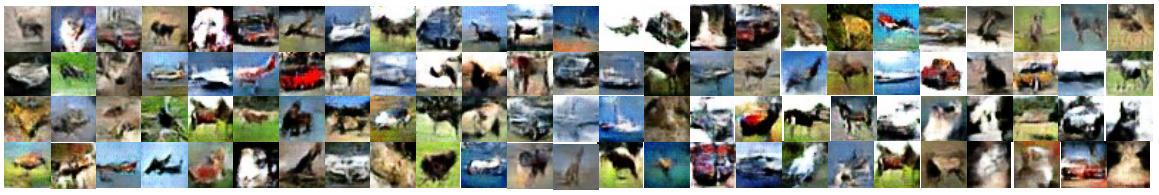


Figure 2.25: Image Samples Generated at 5000th Iteration Using Ternarized Barycentric Fast Adaptation on CIFAR10 Dataset.

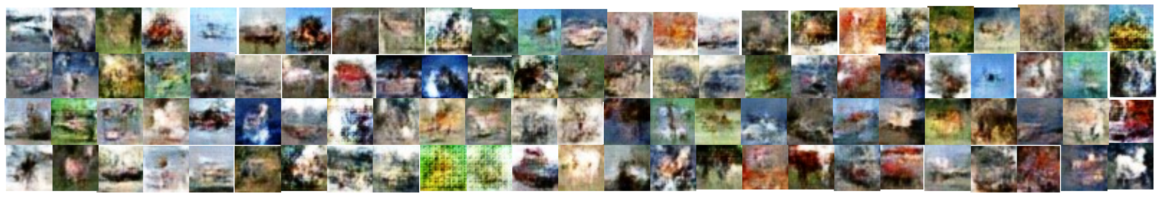


Figure 2.26: Image Samples Generated at 5000th Iteration Using Edge-Only on CIFAR10 Dataset.

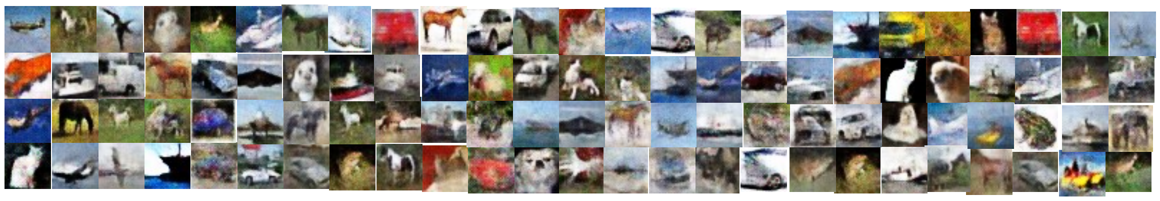


Figure 2.27: Image Samples Generated at 90000th Iteration Using Edge-Only on CIFAR10 Dataset.

Chapter 3

FEDERATED LEARNING OVER WIRELESS NETWORKS

3.1 Introduction

In many edge networks, mobile and IoT devices collecting a huge amount of data are often connected to each other or a central node wirelessly. The unreliable nature of wireless connectivity, together with constraints in computing resources at edge devices, puts forth a significant challenge for the computation, communication and coordination required to learn an accurate model at the network edge. In this chapter, we consider a many-to-one wireless architecture for distributed learning at the network edge, where the edge devices collaboratively train a machine learning model, using local data, in a distributed manner. This departs from conventional approaches which rely heavily on cloud computing to handle high complexity processing tasks, where one significant challenge is to meet the stringent low latency requirement. Further, due to privacy concerns, it is highly desirable to derive local learning model updates without sending data to the cloud. In such distributed learning scenarios, the communication between the edge devices and the server can become a bottleneck, in addition to the other challenges in achieving edge intelligence.

In this chapter, we consider a wireless edge network with M devices and an edge server, where a high-dimensional machine learning model is trained using distributed learning. In such a setting with unreliable and rate-limited communications, local updates at sender devices should be carefully crafted and compressed to make full use of the wireless communication resources available and should work in concert with the receiver (edge server) so as to learn an accurate model. Notably, lossy wireless communications for edge intelligence presents unique challenges and opportunities [172], subject to bandwidth and power requirements, on top of the employed multiple access techniques. Since it often suffices to compute a function of the sum of the local updates for training the model, over-the-air computing is a favorable alternative to the standard multiple-access communications for edge learning. More specifically, over-the-air computation [54, 1] takes advantage of the superposition property of wireless multiple-access channel via simultaneous analog transmissions of the local messages, and then computes a function of the messages at the receiver, scaling signal-to-noise ratio (SNR) well with increasing number of users. In a nutshell, when multiple edge devices collaboratively train a model, it is plausible to employ distributed learning over-the-air.

We seek to answer the following key questions: 1) What is the impact of the wireless communication bandwidth/power on the accuracy and convergence of the edge learning? 2) What coordinates in local gradient signals should be communicated by each edge device to the receiver? 3) How should the coordination be carried out so that multiple sender devices can work in concert with the receiver? 4) What is the optimal way for the receiver to process the received noisy gradient signals to be used for the stochastic gradient descent algorithm? 5) How should each sender device carry out power allocation across subcarriers to transmit its local updates? Intuitively, it is sensible to allocate more power to a coordinate with larger gradient value to speed up the convergence. Further, power allocation should also be channel-aware.

To answer the above questions, we consider an integrated learning and communication scheme where multiple edge devices send their local gradient updates over multi-carrier communications to the receiver for learning (Figure 3.1). Let K denote the number of subcarriers for communications, where K is determined by the wireless bandwidth. First, K dimensions of the gradient updates are determined (by the re-

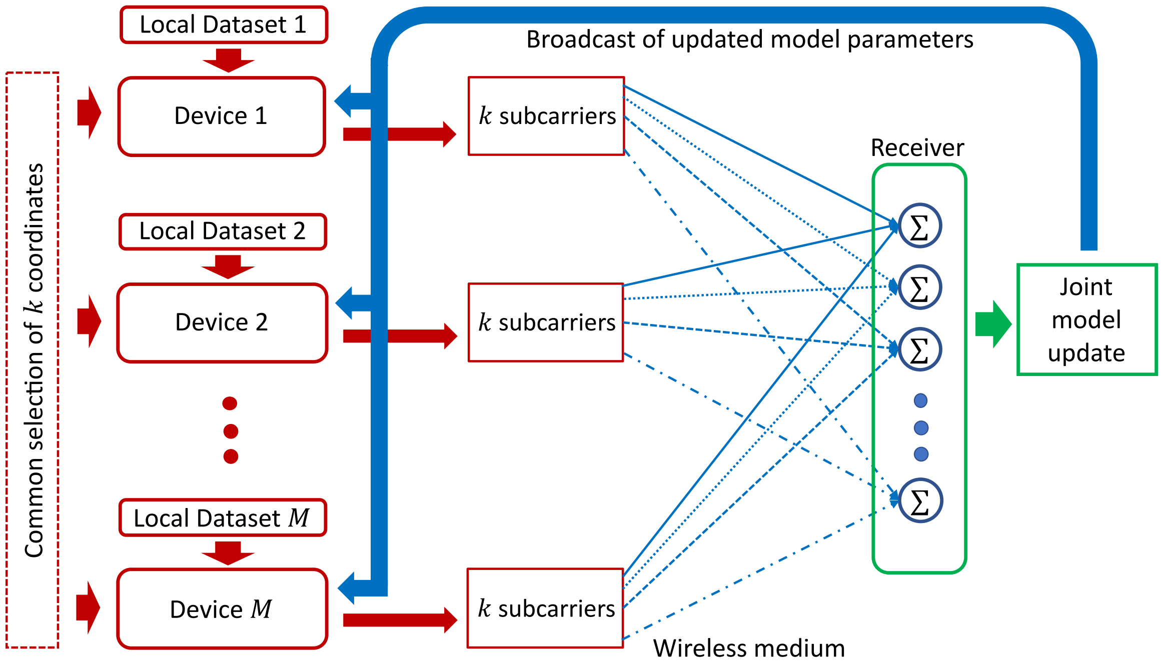


Figure 3.1: A Bandlimited Coordinate Descent Algorithm for Distributed Learning over Wireless Multi-access Channel.

ceiver) to be transmitted. Multiple methods can be used for selecting K coordinates, e.g., selecting the top-k (in absolute value) coordinates of the sum of the gradients or randomized uniform selection. This chapter will focus on randomly uniform selection (we elaborate further on this in Section 3.5). During the subsequent communications, the gradient updates are transmitted only in the K-selected dimensions via over-the-air computing over K corresponding sub-carriers, each experiencing time-varying channel conditions and hence time-varying transmission errors. The devices are subject to power constraints, giving rise to a key question on how to allocate transmission power across dimension, at each edge device, based on the gradient update values and channel conditions. Thus, we explore joint optimization of the power allocation and the learning rate to obtain the best estimate of the gradient updates and minimize the impact of the communication error. We investigate a centralized solution to this problem as a benchmark, and then devise sub-optimal distributed solutions amenable to practical implementation. We note that we have also studied the impact of errors of synchronization across devices in this setting.

The main contributions of this chapter are summarized as follows:

- We take a holistic approach to study federated learning algorithms over wireless MAC channels, and the proposed bandlimited coordinated descent(BLCD) algorithm is built on innovative integration of computing in the air, multi-carrier communications, and wireless resource allocation.

- We characterize the impact of communication error and compression, in terms of its resulting gradient bias and mean squared error (MSE), on the convergence performance of the proposed algorithms. Specifically, when the communication error is unbiased, the BLCD algorithm would converge to a stationary point under very mild conditions on the loss function. In the case the bias in the communication error does exist, the iterates of the BLCD algorithm would return to a contraction region centered around a scaled version of the bias infinitely often.

- To minimize the impact of the communication error, we study joint optimization of power allocation at individual devices and learning rates at the receiver. Observe that since there exists tradeoffs between bias and variance, minimizing the MSE of the communication error does not necessarily amount to minimizing the bias therein. Our findings reveal that optimal power allocation across different sub-carriers should take into account both the gradient values and channel conditions, thus generalizing the widely used water-filling policy. We also develop sub-optimal distributed solutions amenable to implementation. In particular, due to the power constraints at individual devices, it is not always feasible to achieve unbiased estimators of the gradient signal across the coordinates. To address this complication, we develop a distributed algorithm which can drive the bias in the communication error to (close to) zero under given power constraints and then reduce the corresponding variance as much as possible.

3.2 Related Work

Communication-efficient SGD algorithms are of great interest to reduce latency caused by the transmission of the high dimensional gradient updates with minimal performance loss. Such algorithms in the ML literature are based on compression via quantization [7, 149, 20, 151], sparsification [4, 125, 8] and federated learning [78] (or local updates [124]), where lossless communication is assumed to be provided. At the wireless edge, physical-layer design and communication loss should be taken into consideration for the adoption of the communication-efficient algorithms.

Power allocation for over-the-air computation is investigated for different scenarios in many other works [45, 92, 148, 171, 25] including MIMO, reduced dimensional MIMO, standard many to one channel and different channel models. In related works on ML over wireless channels, [173, 155, 163, 12, 11, 13, 3, 112] consider over-the-air transmissions for training of the ML model. The authors in [12] propose sparsification of the updates with compressive sensing for further bandwidth reduction, and recovered sum of the compressed sparse gradients is used for the update. They also apply a similar framework for federated learning and fading channels in [11]. [173] considers a broadband aggregation for federated learning with opportunistic scheduling based on the channel coefficients for a set of devices uniformly distributed over a ring. Lastly, [112] optimize the gradient descent based learning over multiple access fading channels. It is worth noting that the existing approaches for distributed learning in wireless networks do not fully account for the characteristics of lossy wireless channels. It is our hope that the proposed BLCD algorithms can lead to an innovative architecture of distributed edge learning over wireless networks that accounts for computation, power, spectrum constraints and packet losses.

3.3 Federated Learning over Wireless Multi-access Networks

3.3.1 Distributed Edge Learning Model

Consider an edge computing environment with M devices $\mathcal{M} = \{1, \ldots, M\}$ and an edge server. As illustrated in Figure 3.1, a high-dimensional ML model is trained at the server by using an SGD based algorithm, where stochastic gradients are calculated at the devices with the data points obtained by the devices and a (common) subset of the gradient updates are transmitted through different subcarriers via over-the-air.

The general edge learning problem is as follows:

$$\min_{w \in \mathbb{R}^d} f(w) := \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{\xi_m}[l(w, \xi_m)], \tag{3.1}$$

in which $l(\cdot)$ is the loss function, and edge device m has access to inputs ξ_m. Such optimization is typically performed through empirical risk minimization iteratively. In the sequel, we let w_t denote the parameter value of the ML model at communication round t, and at round t edge device m uses its local data $\xi_{m,t}$ to compute a stochastic gradient $g_t^m(w_t) := \nabla l(w_t, \xi_{m,t})$. Define $g_t(w_t) = \frac{1}{M} \sum_{m=1}^{M} g_t^m(w_t)$. The standard vanilla SGD algorithms is given as:

$$w_{t+1} = w_t - \gamma g_t(w_t) \tag{3.2}$$

with γ being the learning rate. Nevertheless, different updates can be employed for different SGD algorithms, and this chapter will focus on communication-error-aware SGD algorithms.

3.3.2 Bandlimited Coordinate Descent Algorithm

Due to the significant discrepancy between the wireless bandwidth constraint and the high-dimensional nature of the gradient signals, a sparse variant of the SGD algorithm over wireless multiple-access channel, named as bandlimited coordinate descent (BLCD), is proposed in which at each iteration only a common set of K coordinates, $I(t) \subset \{1, \ldots, d\}$ (with $K \ll d$), of the gradients are selected to be transmitted through over-the-air computing for the gradient updates. The details of coordinate selection for the BLCD algorithm are relegated to Section 3.6. Worth noting is that due to the unreliable nature of wireless connectivity, the communication is assumed to be lossy, resulting in erroneous estimation of the updates at the receiver. Moreover, gradient correction is performed by keeping the difference between the update made at the receiver and the gradient value at the transmitter for the subsequent rounds, as gradient correction dramatically improves the convergence rate with sparse gradient updates [125]. For convenience, we first define the gradient sparsification operator as follows.

Definition 5. *$C_I : \mathbb{R}^d \to \mathbb{R}^d$ for a set $I \subseteq \{1, \ldots, d\}$ as follows: for every input $x \in \mathbb{R}^d$, $\big(C_I(x)\big)_j$ is $(x)_j$ for $j \in I$ and 0 otherwise.*

Since this operator C_I compress a d-dimensional vector to a k-dimension one, we will also refer this operator as compensation operator in the rest of the chapter. With a bit abuse of notation, we let C_t denote $C_{I(t)}$ for convenience in the following. Following [73], we incorporate the sparsification error made in each iteration (by the compression operator C_t) into the next step to alleviate the possible gradient bias therein and improve the convergence possible. Specifically, as in [73], one plausible way for compression error correction is to update the gradient correction term as

Algorithm 4 Bandlimited Coordinate Descent Algorithm

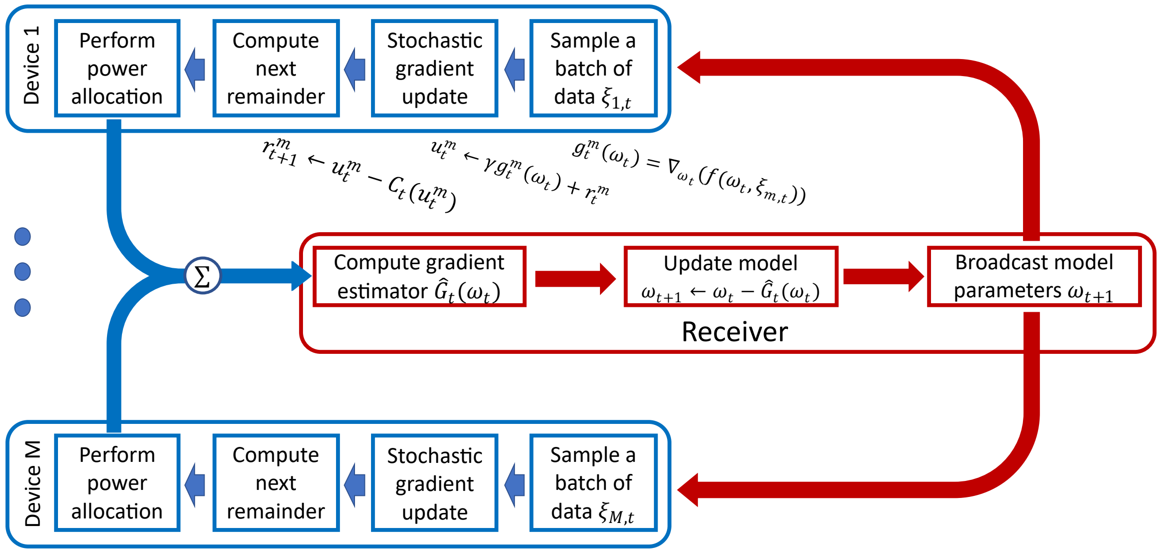
```
1: Input: Sample batches $\xi_{m,t}$, model parameters $w_1$, initial learning rate $\gamma$, sparsification operator $C_t(.)$, $\forall m = 1, \ldots, M; \forall t = 1, \ldots, T$.
2: Initialize: $r_t^m := 0$.
3: for $t = 1 : T$ do
4:     for $m = 1 : M$ do
5:         $g_t^m(w_t) :=$ stochasticGradient$(f(w_t, \xi_{m,t}))$
6:         $u_t^m := \gamma g_t^m(w_t) + r_t^m$
7:         $r_{t+1}^m := u_t^m - C_t(u_t^m)$
8:         Compute power allocation coefficients $b_{km}^*, \forall k = 1, \ldots, K$.
9:         Transmit $\mathbf{b}^* \odot C_t(u_t^m)$
10:    end for
11:    Compute gradient estimator $\hat{G}_t(w_t)$
12:    $w_{t+1} := w_t - \hat{G}_t(w_t)$.
13:    Broadcast $w_{t+1}$ back to all transmitters.
14: end for
```

follows:

$$r_{t+1}^m = u_t^m - C_t(u_t^m), \tag{3.3}$$

$$u_t^m \triangleq \gamma g_t^m(w_t) + r_t^m, \tag{3.4}$$

in which r_{t+1}^m keeps the error in the sparsification operator that is in the memory of user m at around t, and u_t^m is the scaled gradient with correction at device m where the scaling factor γ is the learning rate in equation (3.2). (We refer readers to [73] for more insights of this error-feedback based compression SGD.) Due to the lossy nature of wireless communications, there would be communication errors and the gradient estimators at the receiver would be erroneous. In particular, the gradient estimator


Device 1
Perform power allocation
Compute next remainder
Stochastic gradient update
Sample a batch of data $\xi_{1,t}$
$r_{t+1}^m \leftarrow u_t^m - C_t(u_t^m)$
$u_t^m \leftarrow \gamma g_t^m(\omega_t) + r_t^m$
$g_t^m(\omega_t) = \nabla_{\omega_t}(f(\omega_t, \xi_{m,t}))$
Σ
Compute gradient estimator $\hat{G}_t(\omega_t)$
Update model $\omega_{t+1} \leftarrow \omega_t - \hat{G}_t(\omega_t)$
Broadcast model parameters ω_{t+1}
Receiver
Device M
Perform power allocation
Compute next remainder
Stochastic gradient update
Sample a batch of data $\xi_{M,t}$
Figure 3.2: The Illustration of the Operation of Bandlimited Coordinate Descent Algorithm.

at the receiver in the BLCD can be written as:

$$\hat{G}_t(w_t) = \frac{1}{M}\sum_{m=1}^{M} C_t\left(u_t^m\right) + \epsilon_t, \tag{3.5}$$

where ϵ_t denotes the random communication error in round t. In a nutshell, the bandlimited coordinate descent algorithm is outlined in Algorithm 4 (Figure 3.2).

Recall that $g_t(w_t) = \frac{1}{M}\sum_{m=1}^{M} g_t^m(w_t)$ and define $r_t \triangleq \frac{1}{M}\sum_{m=1}^{M} r_t^m$. Thanks to the common sparsification operator across devices, the update in the SGD algorithm at communication round t is given by:

$$w_{t+1} = w_t - \left[C_t(\gamma g_t(w_t) + r_t) + \epsilon_t\right]. \tag{3.6}$$

To quantify the impact of the communication error, we use the corresponding communication error free counterpart as the benchmark, defined as follows:

$$\hat{w}_{t+1} = w_t - C_t(\gamma g_t(w_t) + r_t). \tag{3.7}$$

It is clear that $w_{t+1} = \hat{w}_{t+1} - \epsilon_t$. For convenience, we define $\tilde{w}_t \triangleq w_t - r_t$. It can be shown that $\tilde{w}_{t+1} = \tilde{w}_t - \gamma g_t(w_t) - \epsilon_t$. Intuitively, w_{t+1} in (3.6) is a noisy version of the iterate $\hat{w}_{t+1}$ in (3.7), which implies that $\tilde{w}_{t+1}$ is a noisy version of the compression-error correction of $\hat{w}_{t+1}$ in (3.7), where the "noisy perturbation" is incurred by the communication error.

3.3.3 BLCD Coordinate Transmissions over Multi-access Channel

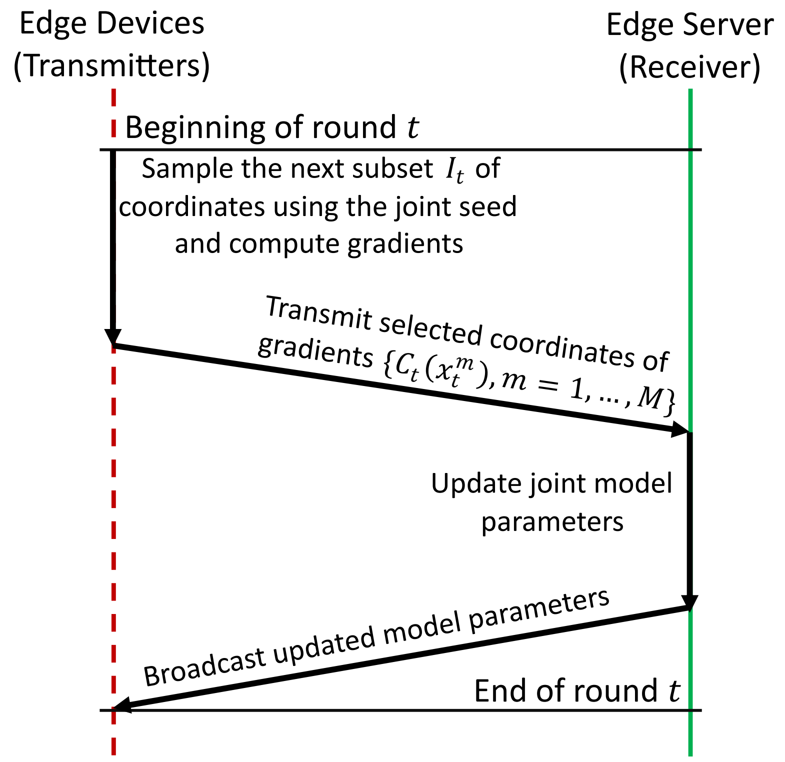

Edge Devices (Transmitters)
Edge Server (Receiver)
Beginning of round t
Sample the next subset I_t of coordinates using the joint seed and compute gradients
Transmit selected coordinates of gradients $\{C_t(x_t^m), m = 1, \ldots, M\}$
Update joint model parameters
Broadcast updated model parameters
End of round t

Figure 3.3: A Multi-access Communication Protocol for Bandlimited Coordinate Selection and Transmission.

A key step in the BLCD algorithm is to achieve coordinate synchronization of the transmissions among many edge devices. To this end, we introduce a receiver-driven low-complexity multi-access communication protocol, as illustrated in Figure 3.3, with the function $C_t(x)$ denoting the compression of x at round t. Let $I(t)$ (of size K) denote the subset of coordinates chosen for transmission by the receiver at round

t. Observe that the updates at the receiver are carried out only in the dimensions $I(t)$. Further, the edge receiver can broadcast its updated iterate to participant devices, over the reverse link. This task is quite simple, given the broadcast nature of wireless channels. In the transmissions, each coordinate of the gradient updates is mapped to a specific subcarrier and then transmitted through the wireless MAC channel, and the coordinates transmitted by different devices over the same subcarrier are received by the edge server in the form of an aggregate sum.

When there are many edge devices, over-the-air computation can be used to take advantage of superposition property of wireless multiple-access channel via simultaneous analog transmissions of the local updates. More specifically, at round t, the received signal in subcarrier k is given by:

$$y_k(t) = \sum_{m=1}^{M} b_{km}(t) h_{km}(t) x_{km}(t) + n_k(t) \tag{3.8}$$

where $b_{km}(t)$ is a power scaling factor, $h_{km}(t)$ the channel gain, and $x_{km}(t)$ the message of user m through the subcarrier k, respectively, and $n_k(t) \sim \mathcal{N}(0, \sigma^2)$ is the channel noise.

To simplify notation, we omit (t) when it is clear from the context in the following. Specifically, the message $x_{km} = (C_t(u_t^m))_{l(k)}$ given the learning algorithm for a one-to-one function $l(k) = (I(t))_k$, which indicates the k-th element of $I(t)$, transmitted through the k-th subcarrier. The total power that a device can use in the transmission is limited in practical systems. Without loss of generality, we assume that there is a power constraint at each device, given by $\sum_{k=1}^{K} |b_{km} x_{km}|^2 \leq E_m,\ \forall m \in \{1, \ldots, M\}$. Note that b_{km} hinges heavily upon both $\boldsymbol{h}_m = [h_{1m}, \ldots, h_{Km}]^\top$ and $\boldsymbol{x}_m = [x_{1m}, \ldots, x_{Km}]^\top$, and a key next step is to optimize $b_{km}(\boldsymbol{h}_m, \boldsymbol{x}_m)$. In each round, each device optimizes its power allocation for transmitting the selected coordinates of its update signal over the K subcarriers, aiming to minimize the communication error so as to achieve a good estimation of $G_t(w_t)$ (or its scaled version) for the gradient update, where

$$G_t(w_t) \triangleq \frac{1}{M} \sum_{m=1}^{M} C_t(u_t^m). \tag{3.9}$$

From the learning perspective, based on $\{y_k\}_{k=1}^K$, it is of paramount importance for the receiver to get a good estimate of $G_t(w_t)$. Since $n_k(t)$ is Gaussian noise, the optimal estimator is in the form of

$$\left(\widehat{G}_t(w_t)\right)_k = \begin{cases} \alpha_{l(k)} y_{l(k)}, & k \in I(t) \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where $\{\alpha_k\}_{k=1}^K$ are gradient estimator coefficients for subcarriers. It follows that the communication error (i.e., the gradient estimation error incurred by lossy communications) is given by:

$$\epsilon_t = \widehat{G}_t(w_t) - G_t(w_t). \tag{3.11}$$

We note that $\{\alpha_k\}_{k=1}^K$ are intimately related to the learning rates for the K coordinates, making the learning rate effectively $\{\gamma\alpha_k\}_{k=1}^K$. It is interesting to observe that the learning rates in the proposed BLCD algorithm are essentially different across the dimensions, due to the unreliable and dynamically changing channel conditions across different subcarriers.

3.4 Impact of Communication Error and Compression on BLCD Algorithm

Recall that thanks to the common sparsification operator across devices, the update in the SGD algorithm at communication round t is given by:

$$w_{t+1} = w_t - \left[C_t(\gamma g_t(w_t) + r_t) + \epsilon_t\right]. \tag{3.12}$$

Needless to say, the compression operator C_t plays a critical role in sparse transmissions. In this chapter, we impose the following standard assumption on the compression rate of the operator.

Assumption 1. *For a set of the random compression operators $\{C_t\}_{t=1}^T$ and any $x \in \mathbb{R}^d$, it holds*

$$\mathbb{E}\left\|x - C_t(x)\right\|^2 \leq (1-\delta)\left\|x\right\|^2 \tag{3.13}$$

for some $\delta \in (0,1]$.

We impose the following standard assumptions on the non-convex objective function $f(\cdot)$ and the corresponding stochastic gradients $g_t^m(w_t)$ computed with the data samples of device m in round t. (We assume that the data samples $\{\xi_{m,t}\}$ are i.i.d. across the devices and time.)

Assumption 2. *(Smoothness) A function $f: \mathbb{R}^d \to \mathbb{R}$ is L-smooth if for all $x, y \in \mathbb{R}^d$, it holds*

$$|f(y) - f(x) - \langle \nabla f(x), y - x\rangle| \leq \frac{L}{2}\left\|y - x\right\|^2. \tag{3.14}$$

Assumption 3. *For any $x \in \mathbb{R}^d$ and for any $m = 1, \ldots, M$, a stochastic gradient $g_t^m(x), \forall t$, satisfies*

$$\mathbb{E}[g_t^m(x)] = \nabla f(x), \;\; \mathbb{E}\left\|g_t^m(x)\right\|^2 \leq G^2 \tag{3.15}$$

where $G > 0$ is a constant.

It follows directly from [73] that $\mathbb{E}[\|r_t\|_2^2] \leq \frac{4(1-\delta)}{\delta^2}\gamma^2 G^2$. Recall that $\tilde{w}_{t+1} = \tilde{w}_t - \gamma g_t(w_t) - \epsilon_t$ and that $\tilde{w}_{t+1}$ can be viewed as a noisy version of the compression-error correction of $\hat{w}_{t+1}$ in (3.7), where the "noisy perturbation" is incurred by the communication error. For convenience, let $\mathbb{E}_t[\epsilon_t]$ denote the gradient bias incurred by the communication error and $\mathbb{E}_t[\|\epsilon_t\|_2^2]$ be the corresponding mean square error, where $\mathbb{E}_t$ is taken with respect to channel noise. Let $\eta = \frac{L-1+2\gamma}{\gamma(2-\rho\gamma)}$ with $0 < \rho < 2$ and let f^* denote the globally minimum value of f. We have the following main result on the iterates in the BLCD algorithm.

Theorem 1. *Under Assumptions 1, 2 and 3, the iterates* $\{w_t\}$ *in the BLCD algorithm satisfies that*

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\left(\|\nabla f(w_t)\|_2 - \eta\|\underbrace{\mathbb{E}_t[\epsilon_t]}_{bias}\|_2\right)^2 \\
&\qquad \le \frac{1}{T+1}\sum_{t=0}^{T}\left[\frac{L\eta}{L-1+2\gamma}\underbrace{\mathbb{E}_t[\|\epsilon_t\|_2^2]}_{MSE} + \left(1+\eta^2\right)\|\underbrace{\mathbb{E}_t[\epsilon_t]}_{bias}\|_2^2\right] \\
&\qquad + \frac{2}{T+1}\frac{f(w_0)-f^*}{\gamma(2-\rho\gamma)} + \left(\frac{L}{\rho}\frac{2(1-\delta)}{\delta^2}+\frac{1}{2}\right)\frac{2L\gamma G^2}{2-\rho\gamma}.
\end{aligned} \tag{3.16}$$

Proof. Recall that $\tilde{w}_t = w_t - r_t$. It can be shown that $\tilde{w}_{t+1} = \tilde{w}_t - \gamma g_t(w_t) - \epsilon_t$. As shown in (3.19), using the properties of the iterates in the BLCD algorithm and the smoothness of the objective function f, we can establish an upper bound on $\mathbb{E}_t[f(\tilde{w}_{t+1})]$ in terms of $f(\tilde{w}_t)$, the corresponding gradient $\nabla f(w_t)$, and the gradient bias and MSE due to the communication error. Then, (3.16) can be obtained after some further algebraic manipulations.We here restate equations (3.6) and (3.7) as follows:

$$w_{t+1} = w_t - [C_t(\gamma g_t(w_t) + r_t) + \epsilon_t] \tag{3.17}$$

$$\hat{w}_{t+1} = w_t - C_t(\gamma g_t(w_t) + r_t) \tag{3.18}$$

It is clear that $w_{t+1} = \hat{w}_{t+1} - \epsilon_t$. For convenience, we define $\tilde{w}_t = w_t - r_t = \hat{w}_t - r_t - \epsilon_{t-1}$.

It can be shown that $\tilde{w}_{t+1} = \tilde{w}_t - \gamma g_t(w_t) - \epsilon_t$ and:

$$\begin{aligned}
\mathbb{E}_t[f(\tilde{w}_{t+1})] &\leq f(\tilde{w}_t) + \langle \nabla f(\tilde{w}_t), \mathbb{E}_t[\tilde{w}_{t+1} - \tilde{w}_t] \rangle + \frac{L}{2}\mathbb{E}_t[\|\tilde{w}_{t+1} - \tilde{w}_t\|^2] \\
&= f(\tilde{w}_t) - \langle \nabla f(\tilde{w}_t), \gamma \mathbb{E}_t[g_t(w_t)] + \mathbb{E}_t[\epsilon_t] \rangle + \frac{L}{2}\mathbb{E}_t[\|\gamma g_t(w_t)\|^2] \\
&+ \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|^2] + L\mathbb{E}_t[\langle \gamma g_t(w_t), \epsilon_t \rangle] \\
&= f(\tilde{w}_t) - \langle \nabla f(w_t), \gamma \mathbb{E}_t[g_t(w_t)] + \mathbb{E}_t[\epsilon_t] \rangle - \langle \nabla f(\tilde{w}_t) - \nabla f(w_t), \gamma \mathbb{E}_t[g_t(w_t)] + \mathbb{E}_t[\epsilon_t] \rangle \\
&+ \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2] + L\mathbb{E}_t[\langle \gamma g_t(w_t), \epsilon_t \rangle] + \frac{L}{2}\mathbb{E}_t[\|\gamma g_t(w_t)\|^2 \\
&\leq f(\tilde{w}_t) - \gamma \|\nabla f(w_t)\|_2^2 - \langle \nabla f(w_t), \mathbb{E}_t[\epsilon_t] \rangle + \frac{\rho}{2}\|\gamma \nabla f(w_t) + \mathbb{E}_t[\epsilon_t]\|_2^2 + \frac{L^2}{2\rho}\mathbb{E}_t[\|r_t\|_2^2] \\
&+ \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2] + L\langle \nabla f(w_t), \mathbb{E}_t[\epsilon_t] \rangle + \frac{L\gamma^2}{2}\mathbb{E}_t\|g_t(w_t)\|_2^2 \\
&\leq f(\tilde{w}_t) - \gamma \|\nabla f(w_t)\|_2^2 + (L-1)\|\nabla f(w_t)\|\|\mathbb{E}_t[\epsilon_t]\| + \frac{\rho}{2}\left(\gamma^2 \|\nabla f(w_t)\|_2^2 + \|\mathbb{E}_t[\epsilon_t]\|_2^2\right. \\
&+ \left. 2\gamma \langle \nabla f(w_t), \mathbb{E}_t[\epsilon_t] \rangle\right) + \frac{L^2}{2\rho}\mathbb{E}_t[\|r_t\|_2^2] + \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2] + \frac{L\gamma^2}{2}G^2
\end{aligned}$$

$$\begin{aligned}
&\leq f(\tilde{w}_t) - \gamma \|\nabla f(w_t)\|_2^2 + (L - 1 + 2\gamma)\|\nabla f(w_t)\|\|\mathbb{E}_t[\epsilon_t]\| + \frac{\gamma^2 \rho}{2}\|\nabla f(w_t)\|_2^2 \\
&+ \frac{L^2}{2\rho}\mathbb{E}_t[\|r_t\|_2^2] + \|\mathbb{E}_t[\epsilon_t]\|_2^2 + \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2] + \frac{L\gamma^2}{2}G^2 \\
&= f(\tilde{w}_t) - \gamma \left[1 - \frac{\rho}{2}\gamma\right] \|\nabla f(w_t)\|_2^2 + (L - 1 + 2\gamma)\|\nabla f(w_t)\|\|\mathbb{E}_t[\epsilon_t]\| \\
&+ \frac{L^2}{2\rho}\mathbb{E}_t[\|r_t\|_2^2] + \|\mathbb{E}_t[\epsilon_t]\|_2^2 + \frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2] + \frac{L\gamma^2}{2}G^2. \qquad (3.19)
\end{aligned}$$

Based on [73], we have that:

$$\mathbb{E}_t[\|r_t\|_2^2] \leq \frac{4(1-\delta)}{\delta^2}\gamma^2 G^2. \qquad (3.20)$$

By using equations 3.19 and 3.20, it follows that:

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\left\{\gamma(1 - \frac{\rho}{2}\gamma)\|\nabla f(w_t)\|_2^2 - (L - 1 + 2\gamma)\|\mathbb{E}_t[\epsilon_t]\|\|\nabla f(w_t)\|\right\} \\
&\qquad \leq \frac{1}{T+1}[f(w_0) - f^*] + \frac{L^2}{\rho}\frac{2(1-\delta)}{\delta^2}\gamma^2 G^2 + \frac{L\gamma^2}{2}G^2 + \frac{1}{T+1}\sum_{t=0}^{T}\left[\|\mathbb{E}_t[\epsilon_t]\|_2^2\right. \\
&\qquad + \left.\frac{L}{2}\mathbb{E}_t[\|\epsilon_t\|_2^2]\right] \qquad (3.21)
\end{aligned}$$

Through some further algebraic manipulation, we have that:

$$\begin{aligned}\frac{1}{T+1}\sum_{t=0}^{T}&\left(\|\nabla f(w_t)\|_2-\frac{L-1+2\gamma}{\gamma(2-\rho\gamma)}\|\mathbb{E}_t[\epsilon_t]\|_2\right)^2\\&\le\frac{2}{T+1}\frac{f(w_0)-f^*}{\gamma(2-\rho\gamma)}+\left(\frac{L}{\rho}\frac{2(1-\delta)}{\delta^2}+\frac{1}{2}\right)\frac{2L\gamma G^2}{2-\rho\gamma}\\&+\frac{1}{T+1}\sum_{t=0}^{T}\left[\frac{L}{\gamma(2-\rho\gamma)}\mathbb{E}_t[\|\epsilon_t\|_2^2]+\left(1+\frac{(L-1+2\gamma)^2}{\gamma^2(2-\rho\gamma)^2}\right)\|\mathbb{E}_t[\epsilon_t]\|_2^2\right]\end{aligned}\tag{3.22}$$

For convenience, let $\eta=\frac{L-1+2\gamma}{\gamma(2-\rho\gamma)}$, and define a contraction region as follows:

$$C_\beta=\{\|\nabla f(w_t)\|_2\ge(\eta+\Delta)\|\mathbb{E}_t[\epsilon_t]\|_2\}\,.\tag{3.23}$$

It follows from (3.22) that iterates in the BLCD algorithm returns to the contraction region infinitely often with probability one. Further, when setting $\gamma=\frac{1}{\sqrt{T+1}}$, we have that:

$$\begin{aligned}\frac{1}{T+1}\sum_{t=0}^{T}&\left(\|\nabla f(w_t)\|_2-\frac{L-1+2\gamma}{\gamma(2-\rho\gamma)}\|\mathbb{E}_t[\epsilon_t]\|_2\right)^2\\&\le\frac{L}{(2-\rho\gamma)}\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}_t[\|\epsilon_t\|_2^2]+\left(1+\frac{(L-1+2\gamma)^2}{\gamma^2(2-\rho\gamma)^2}\right)\frac{1}{T+1}\sum_{t=0}^{T}\|\mathbb{E}_t[\epsilon_t]\|_2^2\\&+\frac{f(w_0)-f^*}{(1-\frac{1}{2}\rho\gamma)}+\frac{1}{\sqrt{T+1}}\left(\frac{L}{\rho}\frac{2(1-\delta)}{\delta^2}+\frac{1}{2}\right)\frac{2LG^2}{2-\rho\gamma},\end{aligned}\tag{3.24}$$

which completes the proof. □

Remarks. Based on Theorem 1, we have a few observations in order.

- We first examine the four terms on the right hand side of (3.16): The first two terms capture the impact on the gradient by the time average of the bias in the communication error ϵ_t and that of the corresponding the mean square, denoted as MSE; the two items would go to zero if the bias and the MSE diminish; the third term is a scaled version of $f(w_0)-f^*$ and would go to zero as long as $\gamma=O(T^{-\beta})$ with $\beta<1$; and the fourth term is proportional to γ and would go to zero when $\gamma\to 0$.

- If the right hand side of (3.16) diminishes as $T \to \infty$, the iterates in the BLCD algorithm would "converge" to a neighborhood around $\eta\|\mathbb{E}_t[\epsilon_t]\|_2$, which is a scaled version of the bias in the communication error. For convenience, let $\bar{\epsilon} = \limsup_t \|\mathbb{E}_t[\epsilon_t]\|_2$, and define a contraction region as follows:

$$A_\gamma = \{w_t : \|\nabla f(w_t)\|_2 \leq (\eta + \Delta)\bar{\epsilon}\} . \tag{3.25}$$

  where $\Delta > 0$ is an arbitrarily small positive number. It then follows that the iterates in the BLCD algorithm would "converge" to a contraction region given by A_γ, in the sense that the iterates return to A_γ infinitely often. Note that f is assumed to any nonconvex smooth function, and there can be many contraction regions, each corresponding to a stationary point.

- When the communication error is unbiased, the gradients would diminish to 0 and hence the BLCD algorithm would converge to a stationary point. In the case the bias in the communication error does exist, there exists intrinsic tradeoff between the size of the contraction region and $\eta\|\mathbb{E}_t[\epsilon_t]\|_2$. When the learning rate γ is small, the right hand side of (3.16) would small, but η can be large, and vice verse. It makes sense to choose a fixed learning rate that would make η small. In this way, the gradients in the BLCD algorithm would "concentrate" around a (small) scaled version of the bias.

- Finally, the impact of gradient sparsification is captured by δ. For instance, when (randomly) uniform selection is used, $\delta = \frac{k}{d}$. We will elaborate on this in Section 3.6.

Further, we have the following corollary.

Corollary 2. *Under Assumptions 1, 2, and 3, we have that if* $\mathbb{E}_t[\epsilon_t] = 0$ *and* $\gamma =$

$\frac{1}{\sqrt{T+1}}$, *the BLCD algorithm converges to a stationary point and satisfies that*

$$\frac{1}{T+1}\sum_{t=0}^{T}\|\nabla f(w_t)\|_2^2$$
$$\leq \frac{1}{2-\frac{\rho}{\sqrt{T+1}}}\left\{\frac{2(f(w_0)-f^*)}{\sqrt{T+1}}+\frac{2LG^2}{\sqrt{T+1}}\left(\frac{L}{\rho}\frac{2(1-\delta)}{\delta^2}+\frac{1}{2}\right)+\frac{L}{T+1}\sum_{t=0}^{T}\underbrace{\mathbb{E}_t[\|\epsilon_t\|_2^2]}_{MSE}\right\} \quad (3.26)$$

3.5 Communication Error Minimization via Joint Optimization of Power Allocation and Learning Rates

Theorem 1 reveals that the communication error has a significant impact on the convergence behavior of the BLCD algorithm. In this section, we turn our attention to minimizing the communication error (in term of MSE and bias) via joint optimization of power allocation and learning rates.

With loss of generality, we focus on iteration t (with abuse of notation, we omit t in the notation for simplicity). Recall that the coordinate updates in the BLCD algorithm, sent by different devices over the same subcarrier, are received by the edge server as an aggregate sum, which is used to estimate the gradient value in that specific dimension. We denote the power coefficients and estimators as $\boldsymbol{b} \triangleq [b_{11}, b_{12}, \ldots, b_{1M}, b_{21}, \ldots, b_{KM}]$ and $\boldsymbol{\alpha} \triangleq [\vec{\alpha}_1, \ldots, \vec{\alpha}_K]$. In each round, each sender device optimizes its power allocation for transmitting the selected coordinates of their updates over the K subcarriers, aiming to achieve the best convergence rate. We assume that the perfect channel state information is available at the corresponding transmitter, i.e., $\boldsymbol{h}_m = [h_{1m}, \ldots, h_{Km}]^\top$ is available at the sender m only.

Based on (3.11), the mean squared error of the communication error in iteration t is given by

$$\mathbb{E}_t[\|\epsilon_t\|_2^2] = \mathbb{E}\left[\left\|\widehat{G}_t(w_t) - G_t(w_t)\right\|^2\right] \quad (3.27)$$

where the expectation is taken over the channel noise. For convenience, we denote $\mathbb{E}_t[\|\epsilon_t\|_2^2]$ as MSE$_1$, which can be rewritten as the sum of the variance and the square of the bias:

$$\mathrm{MSE}_1(\boldsymbol{\alpha}, \boldsymbol{b}) = \sum_{k=1}^{K} \underbrace{\left[\sum_{m=1}^{M} \left(\alpha_k b_{km} h_{km} - \frac{1}{M} \right) x_{km} \right]^2}_{\text{bias in } k\text{th coordinate}} + \underbrace{\sum_{k=1}^{K} \sigma^2 \alpha_k^2}_{\text{variance}} \tag{3.28}$$

Recall that $\{\alpha_k\}_{k=1}^K$ are intimately related to the learning rates for the K coordinates, making the learning rate effectively $\{\gamma\alpha_k\}_{k=1}^K$.

3.5.1 Centralized Solutions to Minimizing MSE (Scheme 1)

In light of the above, we can cast the MSE minimization problem as a learning-driven joint power allocation and learning rate problem, given by:

$$\textbf{P1:} \min_{\boldsymbol{\alpha}, \boldsymbol{b}} \quad \mathrm{MSE}_1(\boldsymbol{\alpha}, \boldsymbol{b}) \tag{3.29}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} |b_{km} x_{km}|^2 \leq E_m, \qquad \forall m \tag{3.30}$$

$$b_{km} \geq 0, \;\; \alpha_k \geq 0 \qquad \forall k, m \tag{3.31}$$

which minimizes the MSE for every round. The above formulated problem is non-convex because the objective function involves the product of variables. Nevertheless, it is biconvex, i.e., for one of the variables being fixed, the problem is convex for the other one. In general, we can solve the above bi-convex optimization problem in the same spirit as in the EM algorithm, by taking the following two steps, each optimizing over a single variable, iteratively:

$$\textbf{P1-a:} \min_{\boldsymbol{\alpha}} \quad \mathrm{MSE}_1(\boldsymbol{\alpha}, \boldsymbol{b}) \quad \text{s.t.} \quad \alpha_k \geq 0, \qquad \forall k$$

$$\begin{aligned} \textbf{P1-b:} \min_{\boldsymbol{b}} \quad & \mathrm{MSE}_{11}(\boldsymbol{\alpha}, \boldsymbol{b}) \\ \text{s.t.} \quad & \sum_{k=1}^{K} |b_{km} x_{km}|^2 \leq E_m, \qquad \forall m, \\ & b_{km} \geq 0, \qquad \forall k, m. \end{aligned}$$

Since **P1-a** is unconstrained,for given $\{b_{km}\}$, the optimal solution to **P1-a** is given by:

$$\alpha_k^* = \max\left\{ \frac{\big(\sum_{m=1}^{M} x_{km}\big)\big(\sum_{m=1}^{M} b_{km}h_{km}x_{km}\big)}{M\big[\sigma^2 + \big(\sum_{m=1}^{M} b_{km}h_{km}x_{km}\big)^2\big]}, 0 \right\}. \tag{3.32}$$

Then, we can solve **P1-b** by optimizing $\boldsymbol{b}$ only. Solving the sub-problems **P1-a** and **P1-b** iteratively leads to a local minimum, however, not necessarily to the global solution.

Observe that the above solution requires the global knowledge of x_{km}'s and h_{km}'s of all devices, which is difficult to implement in practice. We will treat it as a *benchmark* only. Next, we turn our attention to developing distributed sub-optimal solutions.

3.5.2 Distributed Solutions Towards Zero Bias and Variance Reduction (Scheme 2)

As noted above, the centralized solution to **P1** requires the global knowledge of x_{km}'s and h_{km}'s and hence is not amenable to implementation. Further, minimizing the MSE of the communication error does not necessarily amount to minimizing the bias therein since there exists tradeoffs between bias and variance. Thus motivated, we next focus on devising distributed sub-optimal solutions which can drive the bias in the communication error to (close to) zero, and then reduce the corresponding variance as much as possible.

Specifically, observe from (3.28) that the minimization of MSE cost does not necessarily ensure $\hat{G}$ to be an unbiased estimator, due to the intrinsic tradeoff between bias and variance. To this end, we take a sub-optimal approach where the optimization problem is decomposed into two subproblems: In the subproblem at the transmitters, each device m utilizes its available power and local gradient/channel information to compute a power allocation policy in terms of $\{b_{1m}, b_{2m}, \ldots, b_{Km}\}$. In the subproblem at the receiver, the receiver finds the best possible α_k for all $k = 1, \ldots, K$. Another complication is that due to the power constraints at individual devices, it is not always feasible to achieve unbiased estimators of the gradient signal across the coordinates. Nevertheless, for given power constraints, one can achieved unbiased estimators of a scaled down version of the coordinates of the gradient signal. In light of this, we formulate the optimization problem at each device (transmitter) m to ensure an unbiased estimator of a scaled version ζ_m of the transmitted coordinates, as follows:

$$\textbf{Device m:} \max_{\{b_{km}\}_{k=1:K}} \zeta_m \tag{3.33}$$

$$\text{s.t.} \sum_{k=1}^{K} b_{km}^2 x_{km}^2 \leq E_m, \quad b_{km} \geq 0, \tag{3.34}$$

$$\zeta_m x_{km} - b_{km} h_{km} x_{km} = 0, \qquad \forall k = 1, \ldots, K, \tag{3.35}$$

where maximizing ζ_m amounts to maximizing the corresponding SNR (and hence improving the gradient estimation accuracy). The first constraint in the above is the power constraint, and the second constraint is imposed to ensure that there is no bias of the same scaled version of the transmitted signals across the dimensions for user m. The power allocation solution can be found using Karush-Kuhn-Tucker (KKT) conditions as follows:

$$\zeta_m^* = \sqrt{\frac{E_m}{\sum_{k=1}^{K} \frac{x_{km}^2}{h_{km}^2}}}, \ b_{km}^* = \frac{\zeta_m^*}{h_{km}}, \ \forall k. \tag{3.36}$$

Observe that using the obtained power allocation policy in (3.36), all K transmitted coordinates for device m have the same scaling factor ζ_m. Next, we will ensure zero bias by choosing the right $\boldsymbol{\alpha}$ for gradient estimation at the receiver, which can be obtained by solving the following optimization problem since all transmitted gradient signals are superimposed via the over-the-air transmission:

$$\textbf{Receiver side:} \min_{\{\alpha_k\},} \sum_{k=1}^{K} \nu_k^2(\alpha_k, \{b_{km}^*\}) \tag{3.37}$$

$$\text{s.t. } e_k(\alpha_k, \{b_{km}^*\}) = 0, \quad \alpha_k \geq 0, \qquad \forall k = 1, \ldots, K, \tag{3.38}$$

where e_k and ν_k^2 denote the bias and variance components, given as follows:

$$\begin{aligned} e_k(\alpha_k, \{b_{km}^*\}) &= \alpha_k \left(\sum_{m=1}^{M} \zeta_m^* x_{km} \right) - \frac{1}{M} \sum_{m=1}^{M} x_{km}, \\ \nu_k^2(\alpha_k, \{b_{km}^*\}) &= \alpha_k^2 \sigma^2, \end{aligned} \tag{3.39}$$

for all $k = 1, \ldots, K$. For given $\{\zeta_m^*\}$, it is easy to see that

$$\alpha_k^* = \frac{\frac{1}{M}\sum_{m=1}^{M} x_{km}}{\sum_{m=1}^{M} \zeta_m^* x_{km}} \simeq \frac{1}{\sum_{m=1}^{M} \zeta_m^*}, \qquad \forall k. \tag{3.40}$$

We note that in the above, from an implementation pointview, since $\{x_{km}\}$ is not available at the receiver, it is sensible to set $\alpha_k^\dagger \simeq \frac{1}{\sum_{m=1}^{M} \zeta_m^*}$. Further, $\{\zeta_m^*\}$ is not readily available at the receiver either. Nevertheless, since there is only one parameter ζ_m^* from each sender m, the sum $\sum_{m=1}^{M} \zeta_m^*$ can be sent over a control channel to the receiver to compute $\alpha_k^\dagger$. It is worth noting that in general the bias exists even if E_m is the same for all senders.

Next, we take a closer look at the case when the number of subchannels K is large (which is often the case in practice). Suppose that $\{x_{km}\}$ are i.i.d. across subchannels and users, and so are $\{h_{km}\}$. We can then simplify ζ_m^* further. For ease of exposition, we denote $\mathbb{E}[x_{km}^2] = \varphi^2 + \bar{x}^2$ and $\mathbb{E}\left[\frac{1}{h_{km}^2}\right] = \varpi^2$. When K is large, for every user m we have that:

$$\zeta_m^* = \frac{\sqrt{E_m}}{\sqrt{\sum_{k=1}^{K} \frac{x_{km}^2}{h_{km}^2}}} \underset{\substack{\text{when } K \\ \text{is large}}}{\Longrightarrow} \zeta_m^* \approx \frac{\sqrt{E_m}}{\sqrt{K(\varphi^2 + \bar{x}^2)\varpi^2}} \tag{3.41}$$

As a result, the bias and variance for each dimension k could be written as:

$$e_k(\alpha_k^*, \{b_{km}^*\}) = \sum_{m=1}^{M} \left[\frac{\sqrt{E_m}}{\sum_{m=1}^{M} \sqrt{E_m}} - \frac{1}{M} \right] x_{km}, \qquad \forall k. \tag{3.42}$$

$$\nu_k^2 = \frac{K\varpi^2(\varphi^2 + \bar{x}^2)}{\left(\sum_{m=1}^{M} \sqrt{E_m}\right)^2} \sigma^2, \qquad \forall k. \tag{3.43}$$

Observe that when E_m is the same across the senders, the bias term $\mathbb{E}_t[\epsilon_t] = \mathbf{0}$ in the above setting according to (3.42)*.*

3.5.3 A User-centric Approach Using Single-User Solution (Scheme 3)

In this section, we consider a suboptimal user-centric approach, which provides insight on the power allocation across the subcarriers from a single device perspective. We formulate the single device (say user m) problem as

$$\begin{aligned}\textbf{P2:}\ \min_{\{b_{km}\},\{\alpha_k\}}\ & \sum_{k=1}^{K}\Big[\big(\alpha_k b_{km} h_{km}-1\big)x_{km}\Big]^2+\sigma^2\sum_{k=1}^{K}\alpha_k^2\\ \text{s.t.}\ & \sum_{k=1}^{K}|b_{km}x_{km}|^2\le E_m;\ b_k\ge 0,\ \alpha_k\ge 0,\forall k.\end{aligned}$$

Theorem 2. *The optimal solution $\{b^*_{km},\alpha^*_k\}$ to **P2** is given by:*

$$(b^*_{km})^2=\left[\sqrt{\frac{\sigma^2}{\lambda x_{km}^2 h_{km}^2}}-\frac{\sigma^2}{h_{km}^2 x_{km}^2}\right]^+,\qquad \forall k, \tag{3.44}$$

$$\alpha^*_k=\frac{b^*_{km}h_{km}x_k^2}{\sigma^2+(b^*_{km})^2h_{km}^2x_{km}^2},\qquad \forall k, \tag{3.45}$$

where λ_m is a key parameter determining the waterfilling level:

$$\sum_{k=1}^{K}\left[\sqrt{\frac{1}{\lambda_m}}\sqrt{\frac{x_{km}^2\sigma^2}{h_{km}^2}}-\frac{\sigma^2}{h_{km}^2}\right]^+=E_m. \tag{3.46}$$

Proof. Note that there are no terms with b_k in **P2**, all in terms of b_k^2. By defining the auxiliary variables $\tilde{b}_k=b_k^2$, $\tilde{h}_k=h_k^2/\sigma^2$ and $\tilde{x}_k=1/x_k^2$, we re-formulate **P2** as:

$$\begin{aligned}\textbf{P2-1:}\ \min_{\tilde{\boldsymbol{b}}}\ & \sum_{k=1}^{K}(\tilde{b}_k\tilde{h}_k+\tilde{x}_k)^{-1}\\ \text{s.t.}\ & \sum_{k=1}^{K}\frac{\tilde{b}_k}{\tilde{x}_k}\le E\\ & \tilde{b}_k\ge 0,\quad \forall k\end{aligned} \tag{3.47}$$

which is a convex problem and can be solved in a closed form. Lagrangian function can be formed as:

$$L_{22}(\tilde{\boldsymbol{b}},\boldsymbol{\lambda},\boldsymbol{\mu})=\sum_{k=1}^{K}(\tilde{b}_k\tilde{h}_k+\tilde{x}_k)^{-1}+\lambda(\frac{\tilde{b}_k}{\tilde{x}_k}-E)-\sum_{k=1}^{K}\mu_k\tilde{b}_k \tag{3.48}$$

leading to the KKT conditions:

$$\frac{\partial L_{12}(\tilde{\boldsymbol{b}}, \boldsymbol{\lambda}, \boldsymbol{\mu})}{\partial \tilde{b}_k} = -\tilde{h}_k(\tilde{b}_k\tilde{h}_k + \tilde{x}_k)^{-2} + \frac{\lambda}{\tilde{x}_k} - \mu_k = 0, \tag{3.49}$$

$$\sum_{k=1}^{K} \frac{\tilde{b}_k^*}{\tilde{x}_k} \leq E, \quad \tilde{b}_k^* \geq 0, \tag{3.50}$$

$$\lambda \geq 0, \quad \mu_k \geq 0, \tag{3.51}$$

$$\lambda\left(E - \sum_{k=1}^{K} \frac{\tilde{b}_k^*}{\tilde{x}_k}\right) = 0, \quad \mu_k \tilde{b}_k^* = 0. \tag{3.52}$$

For μ_k=0, and $\lambda > 0$, it leads to the solution:

$$\tilde{b}_k^* = \max\left\{\frac{\sqrt{\frac{\tilde{h}_k\tilde{x}_k}{\lambda}} - \tilde{x}_k}{\tilde{h}_k}, 0\right\} = \left[\frac{\sqrt{\frac{\tilde{h}_k\tilde{x}_k}{\lambda}} - \tilde{x}_k}{\tilde{h}_k}\right]^+ \tag{3.53}$$

with $E = \sum_{k=1}^{K} \frac{\tilde{b}_k^*}{\tilde{x}_k}$. By combining both equations, we obtain:

$$E = \sum_{k=1}^{K} \left[\frac{\sqrt{\tilde{h}_k\tilde{x}_k}\lambda' - \tilde{x}_k}{\tilde{h}_k\tilde{x}_k}\right]^+ \tag{3.54}$$

for $\lambda' = \sqrt{1/\lambda}$. It can be solved by a water-filling algorithm, where the solution can be found by increasing λ' until the equality is satisfied. After obtaining the optimal λ', it can be placed in $\tilde{b}_k^*$ and we can take $b_k^* = \sqrt{\tilde{b}_k^*}$ as a solution to **P2**. $\square$

Observe that Theorem 2 reveals that the larger the gradient value (and the smaller channel gain) in one subcarrier, the higher power the it should be allocated to in general, and that $\{x_{km}/h_{km)}\}$ can be used to compute the water level for applying the water filling policy.

Based on the above result, in the multi-user setting, each device can adopt the above single-user power allocation solution as given in Theorem 2. This solution can be applied individually without requiring any coordination between devices.

Next, we take a closer look at the case when the number of subchannels K is large. Let $\bar{E}_m$ denote the average power constraint per subcarrier. When K is large, after some algebra, the optimization problem **P2** can be further approximated as follows:

$$\begin{aligned}\textbf{P3:}\ \min_{b_{km}}\ & \mathbb{E}\left[\frac{x_{km}^2\sigma^2}{b_{km}^2 h_{km}^2 x_{km}^2+\sigma^2}\right]\\ \text{s.t.}\ \ & \mathbb{E}\left[b_{km}^2 x_{km}^2\right]\le \bar{E}_m,\ b_{km}\ge 0,\end{aligned} \tag{3.55}$$

where the expectation is taken with respect to $\{h_{km}\}$ and $\{x_{km}\}$. The solution for $k = 1, \ldots, K$ is obtained as follows:

$$b_{km}^* = \sqrt{\left[\frac{\sigma|x_{km}|^{-1}}{h_{km}\sqrt{\lambda_m}}-\frac{\sigma^2}{x_{km}^2 h_{km}^2}\right]^+} \tag{3.56}$$

$$\lambda_m < \frac{h_{km}^2 x_{km}^2}{\sigma_k^2} \Rightarrow b_{km}^* > 0 \tag{3.57}$$

We can compute the bias and the variance accordingly.

3.6 Coordinate Selection for Bandlimited Coordinate Descent Algorithms

The selection of which coordinates to operate on is crucial to the performance of sparsified SGD algorithms. It is not hard to see that selecting the top-k (in absolute value) coordinates of the sum of the gradients provides the best performance. However, in practice it may not always be feasible to obtain top-k of the sum of the gradients, and in fact there are different solutions for selecting k dimensions with large absolute values; see e.g., [11, 71]. Note that each device individually transmitting top-k coordinates of their local gradients is not applicable to the scenario of over-the-air communications considered here. Sequential device-to-device transmissions provides an alternative approach [115], but these techniques are likely to require more bandwidth with wireless connection.

Another approach that is considered is the use of compression and/or sketching for the gradients to be transmitted. For instance, in [11], a system that updates SGD via decompressing the compressed gradients transmitted through over-the-air communication is examined. To the best of our knowledge, such techniques do not come with rigorous convergence guarantees. A similar approach is taken in [71], where the sketched gradients are transmitted through an error-free medium and these are then used to obtain top-k coordinates; the devices next simply transmit the selected coordinates. Although such an approach can be taken with over-the-air computing since only the summation of the sketched gradients is necessary; this requires the transmission of $\mathcal{O}(k \log d)$ dimensions. To provide guarantees with such an approach $\mathcal{O}(k \log d + k)$ up-link transmissions are needed. Alternatively, uniformly selected $\mathcal{O}(k \log d + k)$ coordinates can be transmitted with similar bandwidth and energy requirements. For the practical learning models with non-sparse updates, uniform coordinate selection tend to perform better. Moreover, the common K dimensions can be selected uniformly via synchronized pseudo-random number generators without any information transfer. To summarize, uniform selection of the coordinates is more attractive based on the energy, bandwidth and implementation considerations compared to the methods aiming to recover top-k coordinates; indeed, this is the approach we adopt.

3.7 Experimental Results

In this section, we evaluate the accuracy and convergence performance of the BLCD algorithm, when using one of the following three schemes for power allocation and learning rate selection (aiming to minimize the impact of communication error): 1) Scheme 1: the bi-convex program based solution, 2) Scheme 2: the distributed solution towards zero bias in Section 3.5; 3) Scheme 3: the single-user solution. We use the communication error free scheme as the baseline to evaluate the performance degradation. We also consider the naive scheme (Scheme 4) using equal power allocation for all dimensions, i.e., $b_{km} = \sqrt{E/\sum_{k=1}^{K} x_{km}^2}$.

In our first experiment, we consider a simple single layer neural network trained

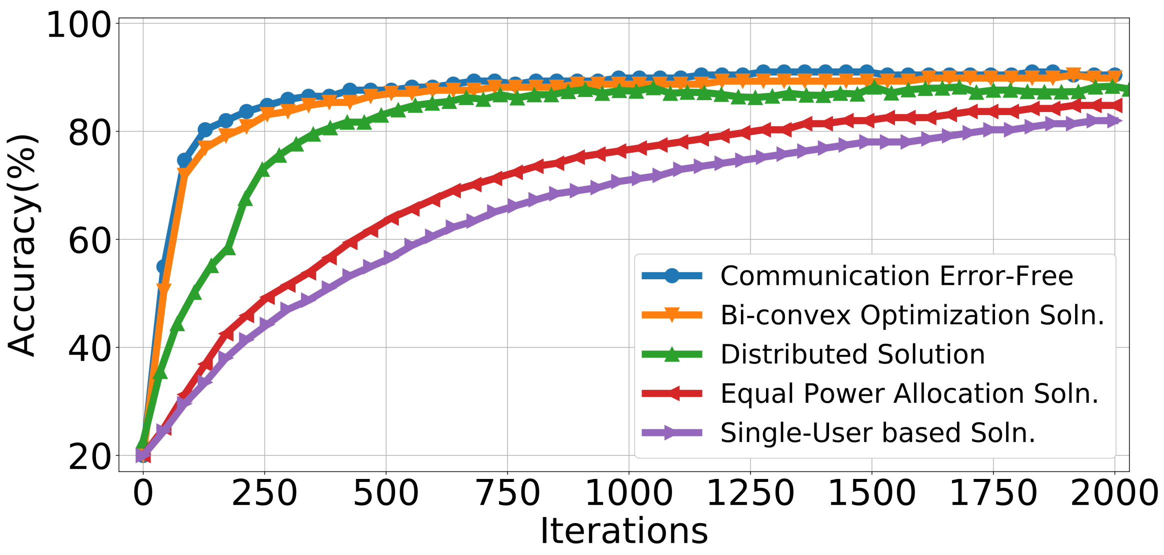


Figure 3.4: Testing Accuracy over Training Iterations for $\alpha_k = 1/8$, $E_{avg} = 0.1$ and a Batch Size of 4. Training Model Consists of a Single Layer Neural Network with 7840 Differentiable Parameters.

on the MNIST dataset. The network has 7840 parameters. $K = 64$ dimensions are uniformly selected as the support of the sparse gradient transmissions. For convenience, we define E_{avg} as the average sum of the energy (of all devices) per dimension normalized by the channel noise variance, i.e., $E_{avg} = EM\,\mathbb{E}[h_{km}^2]/K\sigma^2$. Without loss of generality, we take the variance of the channel noise as $\sigma^2 = 1$ and $\{h_{km}\}$ are independent and identically distributed Rayleigh random variables with mean 1. The changes on E_{avg} simply amount to different SNR values. In Table 3.1 and Figure 3.4, we take $K = 64$, $M = 8$, batch size 4 to calculate each gradient, and the learning rate $\gamma = 0.01$. In the second experiment, we expand the model to a more sophisticated 5-layer neural network with 61706 parameters. We use 10 workers with varying batch sizes and we utilize $K = 1024$ sub-channels for sparse gradient signal transmission.

It can be seen from Figure 3.4 that in the presence of the communication error, the centralized solution (Scheme 1) based on bi-convex programming converges quickly and performs the best, and it can achieve accuracy close to the ideal error-free

Table 3.1: Training Accuracy (%) of the Standard Update Against E_{avg}

E_{avg}	Equal-Power	Distributed	Bi-Convex	Communication-error Free
0.01	83.19	82.42	89.07	91.52
0.05	85.13	86.45	90.25	91.52
0.1	88.24	90.03	90.67	91.52
0.5	89.87	90.42	91.02	91.52
1	89.90	90.85	91.14	91.52
10	90.05	91.10	91.20	91.52

scenario. Further, the distributed solution (Scheme 2) can eventually approach the performance of Scheme 1, but the single-user solution (Scheme 3) performs poorly, so does the naive scheme using equal power allocation (Scheme 4). Clearly, there exists significant gap between its resulting accuracy and that in the error-free case, and this is because the bias in Scheme 3 is more significantly. Table 3.1 summarizes the accuracy after the complete training in the first experiment.

Next, Figures 3.5 and 3.6 and Table 3.2 depict the results in the second experiment using a much larger-scale deep neural network. It can be observed from Figures 3.5 and 3.6 that the SNR can have significant impact of the final accuracy. It is interesting to observe that when the SNR grows, the distributed solution (Scheme 2) can achieve accuracy close to the ideal error-free case, but the single-user solution (Scheme 4) would not. It is worth noting that due to the computational complexity of bi-convex programming in this large-scale case, Scheme 4 could be solved effectively (we did not present it here). Further, the batch size at each worker can impact the convergence rate, but does not impact the final accuracy.

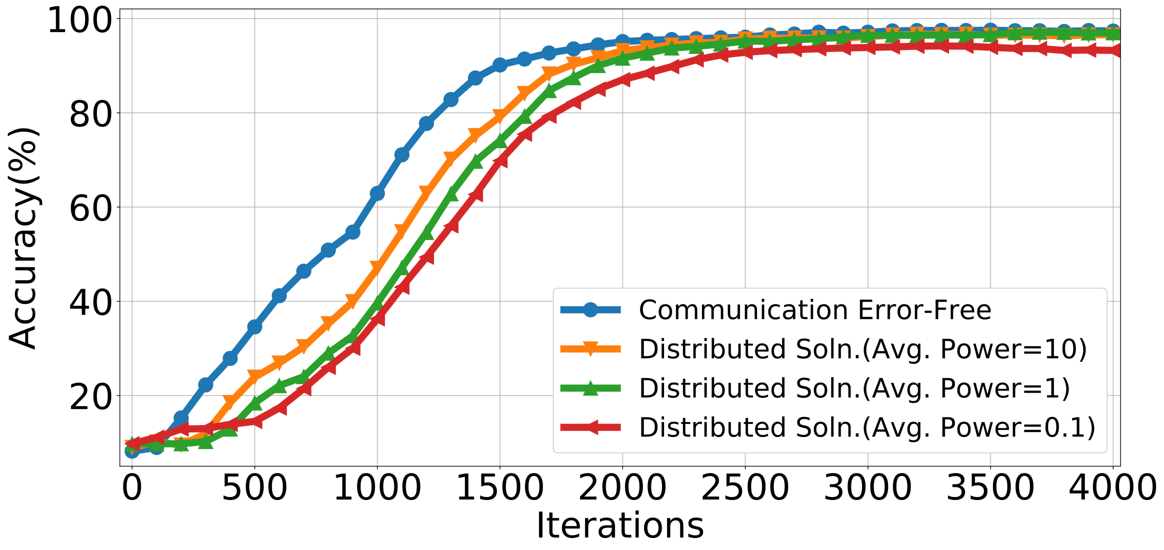


Figure 3.5: Testing Accuracy over Training Iterations for 10 Workers and a Batch Size of 256. Training Model Consists of a 5-layer Deep Neural Network with 61706 Differentiable Parameters.

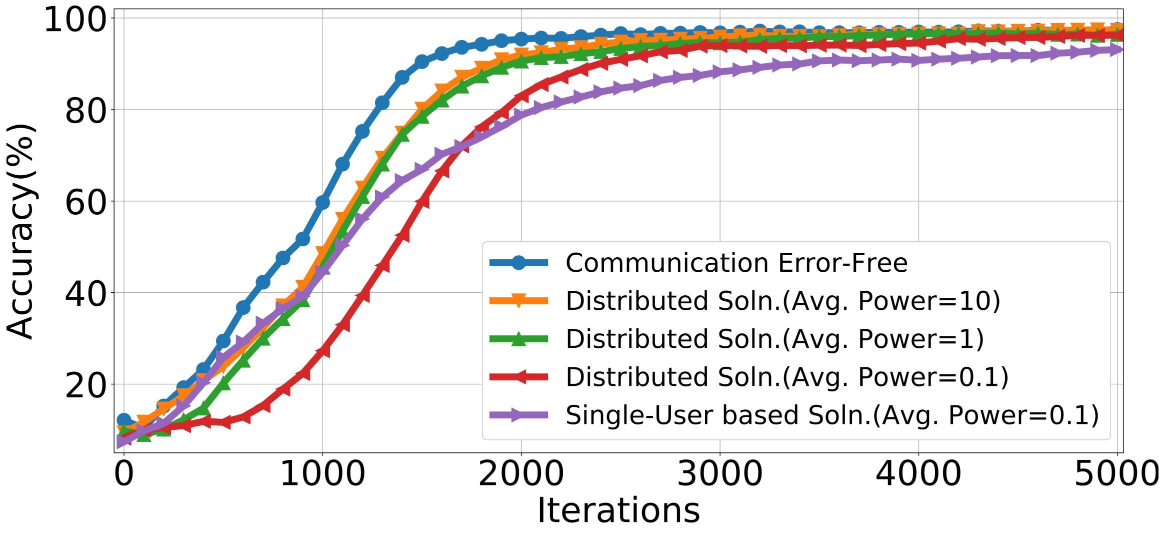


Figure 3.6: Testing Accuracy over Training Iterations for 10 Workers and a Batch Size of 4. Training Model Consists of a 5-layer Deep Neural Network with 61706 Differentiable Parameters.

3.8 Conclusions

In this chapter, we consider a many-to-one wireless architecture for distributed learning at the network edge, where multiple edge devices collaboratively train a ma-

Table 3.2: Final Training Accuracy (%) of the Standard Update Against E_{avg} for a Batch Size of 256.

E_{avg}	0.01	0.05	0.1	1
Distributed Soln. (Scheme 2)	90.47	95.23	96.40	96.42
Communication Error-free	96.49	96.49	96.49	96.49

chine learning model, using local data, through a wireless channel. Observing the unreliable nature of wireless connectivity, we design an integrated communication and learning scheme, where the local updates at edge devices are carefully crafted and compressed to match the wireless communication resources available. Specifically, we propose SGD-based bandlimited coordinate descent algorithms employing over-the-air computing, in which a subset of k-coordinates of the gradient updates across edge devices are selected by the receiver in each iteration and then transmitted simultaneously over k sub-carriers. We analyze the convergence of the algorithms proposed, and characterize the effect of the communication error. Further, we chapter joint optimization of power allocation and learning rates therein to maximize the convergence rate. Our findings reveal that optimal power allocation across different sub-carriers should take into account both the gradient values and channel conditions. We then develop sub-optimal solutions amenable to implementation and verify our findings through numerical experiments.

3.9 Appendix

3.9.1 Derivation of Equation 3.32

Proof. Since **P1-a** is convex, the Lagrangian function is given as:

$$L_{1a}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathrm{MSE}_1(\boldsymbol{b}, \boldsymbol{\alpha}) + \sum_{k=1}^{K} \lambda_k \alpha_k \tag{3.58}$$

Then, the Karush-Kuhn-Tucker (KKT) conditions are given as follows:

$$\frac{\partial L_{1a}(\boldsymbol{\alpha}, \boldsymbol{\lambda})}{\partial \alpha_k} = 2\Big(\sum_{m=1}^{M} b_{km}h_{km}x_{km}\Big)\Big(\sum_{m=1}^{M} \big(\alpha_k b_{km}h_{km} - \frac{1}{M}\big)x_{km}\Big) + 2\sigma^2\alpha_k + \lambda_k = 0$$

$$\lambda_k \geq 0, \quad \lambda_k\alpha_k^* = 0, \quad \alpha_k^* \geq 0. \tag{3.59}$$

$$\text{It follows that: } \alpha_k^* = \max\left\{\frac{\big(\sum_{m=1}^{M} x_{km}\big)\big(\sum_{m=1}^{M} b_{km}h_{km}x_{km}\big)}{M\big[\sigma^2 + \big(\sum_{m=1}^{M} b_{km}h_{km}x_{km}\big)^2\big]}, 0\right\}, \tag{3.60}$$

which completes the derivation. □

3.9.2 An Equal Power Allocation Approach

For comparison, we also consider equal power allocation to each dimension, in other words, we set $b_m = b_{km}$. Satisfying the power constraint of the devices, the equal power solution can be written by $b_m = \sqrt{E/\sum_{k=1}^{K} x_{km}^2}$. Therefore, each device applies b_m in each dimension, taking the advantage of the distribution of the data which is independent and identical.

3.9.3 Decomposable Optimization for Recovering Gradients in Over-the-Air Communication

The over-the-air approach adopted in this chapter dictates the development of a more comprehensive estimator design. To this end, a generalized optimization problem is defined for obtaining the optimal estimator for convergence of SGD algorithm. As a first step towards this optimization problem, we define the mean squared-error (MSE) cost of the communication cost ϵ_t in terms of the received signal

$\mathbf{y} = [y_1, y_2, \ldots, y_K]$ as:

$$\begin{aligned}
\mathbb{E}[(\hat{G} - G)^2] = \mathbb{E}[(\boldsymbol{\alpha} \odot \mathbf{y} - \mathbf{G})^2] &= \frac{1}{K}\sum_{k=1}^{K}\left[\alpha_k\left(\sum_{m=1}^{M} b_{km}h_{km}x_{km} + n_k\right) - G_k\right]^2 \\
&= \frac{1}{K}\sum_{k=1}^{K}\underbrace{\left(\mathbb{E}\left[\alpha_k\left(\sum_{m=1}^{M} b_{km}h_{km}x_{km} + n_k\right)\right] - \mathbb{E}\left[\frac{1}{M}\sum_{m=1}^{M} x_{km}\right]\right)^2}_{e_k} \\
&+ \underbrace{\mathbb{E}\left[\left(\alpha_k\left(\sum_{m=1}^{M} b_{km}h_{km}x_{km} + n_k\right) - \mathbb{E}\left[\alpha_k\left(\sum_{m=1}^{M} b_{km}h_{km}x_{km} + n_k\right)\right]\right)^2\right]}_{\nu_k^2}, \quad (3.61)
\end{aligned}$$

where the estimator is selected as $\alpha_k\left(\sum_{m=1}^{M} b_{km}h_{km}x_{km} + n_k\right)$ for $k = 1, 2, \ldots, K$, in which α_k, b_{km}, h_{km} and n_k denote a correction factor at receiver to recover the kth dimension of true gradient, power allocation factor at transmitter m for dimension k of local gradient x_{km} for equalizing channel fading, the fading at the kth sub-channel between mth transmitter and receiver, and the thermal additive noise at the receiver for sub-channel k, respectively. In (3.61), ν_k^2 and e_k denote the estimator variance and estimator bias, respectively. As apparent in (3.61), the minimization of MSE cost does not ensure $\hat{G}_k$ to be an unbiased estimator of G_k since mitigating variance may attenuate the cost more. Consequently, we formulate the unbiased optimization problem as follows:

$$\begin{aligned}
&\underset{\{\alpha_k\},\{b_{km}\}}{\text{argmin}} \; \sum_{k=1}^{K}\nu_k^2(\alpha_k, \{b_{km}\}) \\
&\quad \text{s.t.} \;\; e_k(\alpha_k, \{b_{km}\}) = 0, \\
&\qquad\quad \sum_{k=1}^{K} b_{km}^2 x_{km}^2 \le E_m, \qquad\qquad \forall m = 1, \ldots, M, \\
&\qquad\quad b_{km} \ge 0, \quad \alpha_k \ge 0, \qquad \forall k = 1, \ldots, K; \;\; \forall m = 1, \ldots, M, \qquad (3.62)
\end{aligned}$$

where E_m denotes the power available to transmitter m. We here notice that h_{km} can be obtained via CSI and E_m, all $\{x_{1m}, x_{2m}, \ldots, x_{Km}\}$ and $\{b_{1m}, b_{2m}, \ldots, b_{Km}\}$ are only available to transmitter m and all α_k are only available to receiver. On the other hand, n_k is not available to neither parties, and hence the distribution knowledge of n_k is available to both transmitters and receiver. Consequently, the optimization problem defined in (3.61) should be decomposed into two stages. In the first stage, each transmitter m utilizes its available power and local gradient information to compute an optimal power allocation set $\{b_{1m}, b_{2m}, \dots, b_{Km}\}$. In the second stage, the receiver solves a consecutive optimization problem to find optimal α_k for all $k = 1, \dots, K$. Ideally, the proposed two-stage algorithm must obtain the same solution as to (3.62). To this end, we formulate the optimization problem at each transmitter as follows:

$$\begin{aligned} &\underset{\{b_{km}\}_{k=1:K}}{\text{argmax}} \ \zeta_m \\ &\qquad \text{s.t. } \mathbb{E}\left[(\zeta_m x_{km} - b_{km} h_{km} x_{km})^2\right] = 0, \\ &\qquad\qquad \sum_{k=1}^{K} b_{km}^2 x_{km}^2 \le E_m, \quad b_{km} \ge 0, \qquad\qquad \forall k = 1, \dots, K. \end{aligned} \tag{3.63}$$

The first constraint in (3.63) ensures that there is no additive bias in the transmitted signal (i.e., the bias can be removed by a multiplicative factor), while second constraint is the power constraint. The first constraint can be restated in a simpler form as $\zeta_m = b_{km} h_{km}$, and then the solution can simply be obtained using KKT conditions as follows:

$$\begin{aligned}
\text{Lagrangian:}\quad & \mathcal{L}(\{b_{km}\},\{\lambda_k\},\vartheta,\{\beta_k\}) = \zeta_m - \sum_{k=1}^{K}\lambda_k(b_{km}h_{km}-\zeta_m) \\
& \qquad - \vartheta\left(\sum_{k=1}^{K} b_{km}^2 x_{km}^2 - E_m\right) + \sum_{k=1}^{K}\beta_k b_{km} \\
\text{Stationarity:}\quad & \frac{\partial\mathcal{L}}{\partial b_{km}} = 0 - \lambda_k h_{km} - 2\vartheta x_{km}^2 b_{km} + \beta_k = 0 \to b_{km} = \frac{\beta_k - \lambda_k h_{km}}{2\vartheta x_{km}^2} \\
\text{Primal Feasibility:}\quad & b_{km}h_{km} = \zeta_m,\quad \sum_{k=1}^{K} b_{km}^2 x_{km}^2 \le E_m,\quad b_{km}\ge 0,\quad \forall k = 1\ldots,K, \\
\text{Dual Feasibility:}\quad & \vartheta \ge 0,\quad \beta_k \ge 0,\quad \forall k = 1,\ldots,K, \\
\text{Comp. Slackness:}\quad & \vartheta\left(\sum_{k=1}^{K} b_{km}^2 x_{km}^2 - E_m\right) = 0,\quad \sum_{k=1}^{K}\beta_k b_{km} = 0.
\end{aligned}$$

Then, the solution becomes:

$$\zeta_m^* = \sqrt{\frac{E_m}{\sum_{k=1}^{K}\frac{x_{km}^2}{h_{km}^2}}}, \qquad b_{km}^* = \frac{\zeta_m}{h_{km}},\quad \forall k = 1,\ldots,K. \tag{3.64}$$

The analytical result computed in (3.64) illustrates that each transmitter utilizes all of their available power to amplify their local gradient transmission. Clearly, each transmitted signal is biased with a multiplicative factor ζ_m^*. Yet, this bias can be removed by multiplying with $\boldsymbol{\alpha}$ in the receiver. However, the optimal $\boldsymbol{\alpha}$ must be obtained by solving an optimization problem since all biased transmitted gradient signals are superimposed via the over-the-air transmission. Consequently, in the second stage, the receiver solves the following optimization problem:

$$\begin{aligned}
\underset{\{\alpha_k\},}{\operatorname{argmin}}\ & \sum_{k=1}^{K}\nu_k^2(\alpha_k,\{b_{km}^*\}) \\
\text{s.t.}\ & e_k(\alpha_k,\{b_{km}^*\}) = 0,\quad \alpha_k \ge 0, \qquad \forall k = 1,\ldots,K.
\end{aligned} \tag{3.65}$$

Subsequently, we derive $e_k(\alpha_k, \{b^*_{km}\})$ and $\nu^2_k(\alpha_k, \{b^*_{km}\})$. For ease of exposition, cumbersome steps are omitted here:

$$\begin{aligned}
e_k(\alpha_k, \{b^*_{km}\}) =& \mathbb{E}\left[\alpha_k y_k - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right] = \mathbb{E}\left[\alpha_k \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km} + n_k\right) - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right] \\
=& \alpha_k \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km}\right) - \frac{1}{M}\sum_{m=1}^{M} x_{km} = \alpha_k \left(\sum_{m=1}^{M} \zeta^*_m x_{km}\right) - \frac{1}{M}\sum_{m=1}^{M} x_{km} \\
\nu^2_k(\alpha_k, \{b^*_{km}\}) =& \mathbb{E}\left[\left(\alpha_k \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km} + n_k\right) - \mathbb{E}\left[\alpha_k \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km} + n_k\right)\right]\right)^2\right] \\
=& \mathbb{E}\left[\left(\alpha_k \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km} - h_{km} b^*_{km} x_{km} + n_k\right)\right)^2\right] = \alpha^2_k \sigma^2, \qquad (3.66)
\end{aligned}$$

where the expectation is taken with respect to n_k for the given realizations of all x_{km} and $\sigma^2 = \mathbb{E}[n^2_k]$. By similarly applying KKT conditions and solving them, we obtain the following solution:

$$\alpha^*_k = \frac{\frac{1}{M}\sum_{m=1}^{M} x_{km}}{\sum_{m=1}^{M} h_{km} b^*_{km} x_{km}} = \frac{\frac{1}{M}\sum_{m=1}^{M} x_{km}}{\sum_{m=1}^{M} \zeta^*_m x_{km}}. \qquad (3.67)$$

From the implementation pointview, since $\{x_{km}\}$ is not available at the receiver, it is sensible to set $\alpha^\dagger_k \simeq \frac{1}{\sum_{m=1}^{M} \zeta^*_m}$. We herein also notice that ζ^*_m for all $m = 1, \ldots, M$ are not available at the receiver as well. Luckily, $\alpha^\dagger_k$ is a function of $\sum_{m=1}^{M} \zeta^*_m$, and hence a subchannel could be allocated for over-the-air transmission of $\sum_{m=1}^{M} \zeta^*_m$. Subsequently, it follows that the bias is:

$$\begin{aligned}
e_k(\alpha^*_k, \{b^*_{km}\}) &= \mathbb{E}\left[\frac{1}{\sum_{m=1}^{M} \zeta^*_m} y_k - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right] \\
&= \mathbb{E}\left[\frac{1}{\sum_{m=1}^{M} \zeta^*_m} \left(\sum_{m=1}^{M} h_{km} b^*_{km} x_{km} + n_k\right) - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right] \\
&= \mathbb{E}\left[\frac{1}{\sum_{m=1}^{M} \zeta^*_m} \left(\sum_{m=1}^{M} \zeta^*_m x_{km} + n_k\right) - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right] \\
&= \mathbb{E}\left[\frac{\sum_{m=1}^{M} \zeta^*_m x_{km} + n_k}{\sum_{m=1}^{M} \zeta^*_m} - \frac{1}{M}\sum_{m=1}^{M} x_{km}\right], \qquad (3.68)
\end{aligned}$$

Assuming that the distributions of $\{x_{km}\}$ for all $k = 1, \ldots, K; m = 1, \ldots, M$ are identical across subchannels and users, and so are $\{h_{km}\}$, ζ_m^* can be simplified. For ease of exposition, we denote $\mathbb{E}[x_{km}^2] = \varphi^2 + \bar{x}^2$ and $\mathbb{E}\left[\frac{1}{h_{km}^2}\right] = \varpi^2$. When the number of subchannels K is large, we have that:

$$\zeta_m^* = \frac{\sqrt{E_m}}{\sqrt{\sum_{k=1}^K \frac{x_{km}^2}{h_{km}^2}}} \underset{\text{when } K \text{ is large}}{\Longrightarrow} \zeta_m^* = \frac{\sqrt{E_m}}{\sqrt{K\mathbb{E}[x_{km}^2]\mathbb{E}\left[\frac{1}{h_{km}^2}\right]}} = \frac{\sqrt{E_m}}{\sqrt{K(\varphi^2 + \bar{x}^2)\varpi^2}} \tag{3.69}$$

$$\begin{aligned} e_k(\alpha_k^*, \{b_{km}^*\}) &= \mathbb{E}\left[\frac{\sum_{m=1}^M \zeta_m^* x_{km} + n_k}{\sum_{m=1}^M \zeta_m^*} - \frac{1}{M}\sum_{m=1}^M x_{km}\right] \\ &= \sum_{m=1}^M \left[\frac{\sqrt{E_m}}{\sum_{m=1}^M \sqrt{E_m}} - \frac{1}{M}\right] x_{km}. \end{aligned} \tag{3.70}$$

$\mathbb{E}_t[\epsilon_t]_k$ in Theorem 1 is expressed as $\|\mathbb{E}_t[\epsilon_t]\|_2^2 = \sum_{k=1}^K \mathbb{E}_t[\epsilon_t]_k^2 = \sum_{k=1}^K e_k^2$. The variance ν_k^2 then can be computed as:

$$\nu_k^2\left(\alpha_k^*, \{b_{km}^*\}\right) = \alpha_k^{*2}\sigma^2 = \left(\frac{1}{\sum_{m=1}^M \sqrt{\frac{E_m}{K\varpi(\varphi^2+\bar{x}^2)}}}\right)^2 \sigma^2 = \frac{K\varpi^2(\varphi^2 + \bar{x}^2)}{\left(\sum_{m=1}^M \sqrt{E_m}\right)^2}\sigma^2. \tag{3.71}$$

Finally, the MSE cost in Theorem 1 can be written as $\mathbb{E}_t[\|\epsilon_t\|_2^2] = \sum_{k=1}^K (\nu_k^2 + e_k^2)$.

3.9.4 Alternative Formulation to P2 (P3)

When K is large, channel coefficients $\{h_{km}\}$ and local gradients $\{x_{km}\}$ are independent and identically distributed (i.i.d.) across users and subchannels, the optimization problem **P2-1** can be further simplified by using law of large numbers:

$$\begin{aligned} \textbf{P2-2:}\ \min_{\tilde{b}}\ & \mathbb{E}\left[\frac{1}{\tilde{b}_{km}\tilde{h}_{km} + \tilde{x}_{km}}\right] \\ \text{s.t. } & \mathbb{E}[\tilde{b}_{km}/\tilde{x}_{km}] \le \bar{E}_m, \\ & \tilde{b}_{km} \ge 0. \end{aligned} \tag{3.72}$$

We here notice that general law of large numbers only applies if all $\tilde{b}_{km}$ are identically distributed. By further noting that $\tilde{b}_{km}$ will depend on $\tilde{x}_{km}$ and $\tilde{h}_{km}$, we can conclude that each objective term in (3.72) is identically distributed. Then, (3.72) simplifies as:

$$\begin{aligned} \textbf{P3:}\ \min_{\tilde{b}_{km}}\ & \mathbb{E}\left[\frac{1}{\tilde{b}_{km}\tilde{h}_{km}+\tilde{x}_{km}}\right] \\ \text{s.t. } & \mathbb{E}\left[\frac{\tilde{b}_{km}}{\tilde{x}_{km}}\right] \le \bar{E}_m, \\ & \tilde{b}_{km} \ge 0. \end{aligned} \tag{3.73}$$

Subsequently, KKT conditions can be written as:

$$\text{Lagrangian: } \mathcal{L} = \mathbb{E}\left[\frac{1}{\tilde{b}_{km}\tilde{h}_{km}+\tilde{x}_{km}}\right] + \lambda_m\left(\mathbb{E}\left[\frac{\tilde{b}_{km}}{\tilde{x}_{km}}\right]-\bar{E}_m\right) - \beta\tilde{b}_{km} \tag{3.74}$$

$$\text{Stationarity: } \frac{\partial\mathcal{L}}{\partial\tilde{b}_{km}} = \mathbb{E}\left[\frac{\lambda_m}{\tilde{x}_{km}} - \frac{\tilde{h}_{km}}{(\tilde{b}_{km}\tilde{h}_{km}+\tilde{x}_{km})^2}\right] = 0 \tag{3.75}$$

$$\Leftarrow \frac{\lambda_m^*}{\tilde{x}_{km}} = \frac{\tilde{h}_{km}}{(\tilde{b}_{km}\tilde{h}_{km}+\tilde{x}_{km})^2} \tag{3.76}$$

$$\text{Primal Feasibility: } \mathbb{E}\left[\frac{\tilde{b}_{km}}{\tilde{x}_{km}}\right] \le \bar{E}_m, \quad \tilde{b}_{km} \ge 0, \tag{3.77}$$

$$\text{Dual Feasibility: } \lambda_m \ge 0, \quad \beta \ge 0, \tag{3.78}$$

$$\text{Comp. Slackness: } \lambda_m\left(\mathbb{E}\left[\frac{\tilde{b}_{km}}{\tilde{x}_{km}}\right]-\bar{E}_m\right) = 0, \quad -\beta\tilde{b}_{km} = 0. \tag{3.79}$$

Finally, the solution is obtained as follows:

$$\tilde{b}_{km} = \left[\sqrt{\frac{\tilde{x}_{km}}{\lambda_m^* \tilde{h}_{km}}} - \frac{\tilde{x}_{km}}{\tilde{h}_{km}}\right]^+ \Rightarrow b_{km} = \sqrt{\left[\frac{\sigma}{h_{km}|x_{km}|\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2 h_{km}^2}\right]^+}$$

$$\lambda_m^* < \frac{\tilde{h}_{km}}{\tilde{x}_{km}} \Rightarrow \tilde{b}_{km} > 0 \text{ and } \lambda_m^* < \frac{h_{km}^2 x_{km}^2}{\sigma^2} \Rightarrow b_{km} > 0 \tag{3.80}$$

$$\mathbb{E}\left[\frac{\tilde{b}_{km}}{\tilde{x}_{km}}\right] = \int_0^\infty \int_0^\infty \frac{\tilde{b}_{km}}{\tilde{x}_{km}} p(\tilde{h}_{km}) p(\tilde{x}_{km}) d\tilde{h}_{km} d\tilde{x}_{km}$$

$$= \int_0^\infty \int_0^{\lambda_m \tilde{x}_{km}} \left(\sqrt{\frac{1}{\lambda_m^* \tilde{h}_{km} \tilde{x}_{km}}} - \frac{1}{\tilde{h}_{km}}\right) p(\tilde{h}_{km}) p(\tilde{x}_{km}) d\tilde{h}_{km} d\tilde{x}_{km} = \bar{E}_m$$

$$\mathbb{E}\left[b_{km}^2 x_{km}^2\right] = \int_{-\infty}^\infty \int_0^\infty \left[\frac{\sigma}{h_{km}|x_{km}|\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2 h_{km}^2}\right]^+ x_{km}^2 p(h_{km}) p(x_{km}) dh_{km} dx_{km}$$

$$= \int_{-\infty}^\infty \int_{\sqrt{\frac{\sigma^2 \lambda_m^*}{x_{km}^2}}}^\infty \left(\frac{|x_{km}|\sigma}{h_{km}\sqrt{\lambda_m^*}} - \frac{\sigma^2}{h_{km}^2}\right) p(h_{km}) p(x_{km}) dh_{km} dx_{km} = \bar{E}_m.$$

The bias term, i.e., e_k, then can be derived as follows:

$$e_k(\alpha_k^*, \{b_{km}\}) = \mathbb{E}\left[\alpha_k^* y_k\right] - \frac{1}{M}\sum_{m=1}^M x_{km} = \mathbb{E}\left[\sum_{m=1}^M \left(\alpha_k^* b_{km} h_{km} - \frac{1}{M}\right) x_{km} + \alpha_k^* n_k\right]$$

$$= \mathbb{E}\left[\sum_{m=1}^M \left(\frac{1}{M}\sqrt{\left[\frac{\sigma}{|x_{km}|h_{km}\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2 h_{km}^2}\right]^+} h_{km} - \frac{1}{M}\right) x_{km} + \frac{n_k}{M}\right]$$

$$= \frac{1}{M}\sum_{m \in \mathcal{S}^\dagger} \left(\sqrt{\frac{\sigma h_{km}}{|x_{km}|\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2}} - 1\right) x_{km}, \tag{3.81}$$

where $\mathcal{S}^\dagger$ represents the set of transmitters for which (3.80) is satisfied. The MSE cost can then be computed by first computing the variance $\nu^2(\alpha_k^*, \{b_{km}\})$:

$$\nu_k^2(\alpha_k^*, \{b_{km}\}) = \mathbb{E}\left[\left(\sum_{m=1}^M \alpha_k^* b_{km} h_{km} x_{km} + \alpha_k^* n_k - \mathbb{E}\left[\sum_{m=1}^M \alpha_k^* b_{km} h_{km} x_{km}\right]\right)^2\right]$$

$$= \mathbb{E}\left[\left(\sum_{m=1}^M \frac{1}{M}\left\{\sqrt{\left[\frac{\sigma h_{km}}{|x_{km}|\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2}\right]^+} x_{km} - \sqrt{\left[\frac{\sigma h_{km}}{|x_{km}|\sqrt{\lambda_m^*}} - \frac{\sigma^2}{x_{km}^2}\right]^+} x_{km}\right\} + \frac{n_k}{M}\right)^2\right]$$

$$= \frac{\sigma^2}{M^2}, \tag{3.82}$$

where the expectation is taken with respect to x_{km}. Subsequently, the MSE cost $\mathbb{E}_t\left[\|\epsilon_t\|_2^2\right]$ is computed as $\mathbb{E}_t[\|\epsilon_t\|_2^2] = \sum_{k=1}^{K}(e_k^2 + \nu_k^2)$.

Chapter 4

OPTIMIZING ELECTRIC VEHICLE CHARGING: FEDERATED LEARNING BASED DEMAND RESHAPING OF ELECTRIC VEHICLES

4.1 Introduction

The number of electric vehicles (EVs) had grown exponentially in the last decade [68, 72]. Though EVs are efficient in power consumption, the physical constraints on the battery and charging technologies have limited EVs to travel shorter distances in comparison to their fuel-using counterparts. Clearly, a denser construction of EV charging stations is necessary for hassle-free driving experience of EVs. Further, the predicted vast number of EVs in near future will also constitute further challenges for the power grid. To name a few challenges, the waiting time in EV charging and the overload of power grid during peak hours every day may lead to power shortages and a significant decrease in the quality of service for EV users.

From the perspective of EV users, EV charging is a time consuming procedure. A typical level 2 charger can charge for 15-25 miles of range in an hour, while a DC charger can charge for a range of 70 miles in an hour [76, 26]. Given that direct current (DC) charging stations are quite limited to metropolitan areas and to specific EV models, EV users may experience latency for charging their vehicles. To mitigate the latency, charging stations often deploy multiple charging ports. However, then the total charging capacity must be split across simultaneously charging EVs, extending the charging duration [64]. Since time is often the most valuable consideration for drivers, charging duration emerges to be one most significant obstacle against the widespread of EVs. Therefore, improvements on charging technology and efficient techniques for reshaping EV charging demand are needed.

On contrary to EV users, charging stations are profit-focused entities. The current pricing policies neither enforce a restricted scheduling of EV charging nor adopt incentive based pricing. Fixed pricing schemes might be attractive for both EV users and EV charging stations in the absence of charging queues, but in the presence of high number of charging requests, it leads to congestion. Planning EV charging ahead of time may increase the quality of service in the presence of congestion [153, 95]. However, forcing EV users to a strict charging schedule is not a plausible solution in comparison the flexibility of internal combustion engine (ICE) vehicles.

A plausible solution is to incentivize EV users with discounted charging prices to offload charging congestion during peak hours [144, 105]. In this study, charging stations announce incentivized charging prices based on the prediction of EV arrivals. Clearly, charging stations need to take into account the dynamic electricity prices and the availability of power in the grid as well, while maximizing their rewards. A good incentive mechanism clearly can even out the EV charging demand and increase the EV user satisfaction. Consequently, the profitability of the EV charging stations and the safety of the power grid are improved in the long term.

In light of the negative effects of EV charging congestion on waiting time and the stability of power grid, we devise an incentive based demand reshaping framework, consisting of a federation of charging stations and EV users. Each EV user possesses private information such as the distance to every charging station and the importance of charging price and charging duration. Each EV user selects a charging station and a time-frame to charge her/his EV based on her/his private information, the incentivized prices and the average charging duration. The maximized reward constitutes less charging duration, lower price and closer charging station (Figure 4.1).

EV users' charging station choices translates to EV arrivals for charging stations.

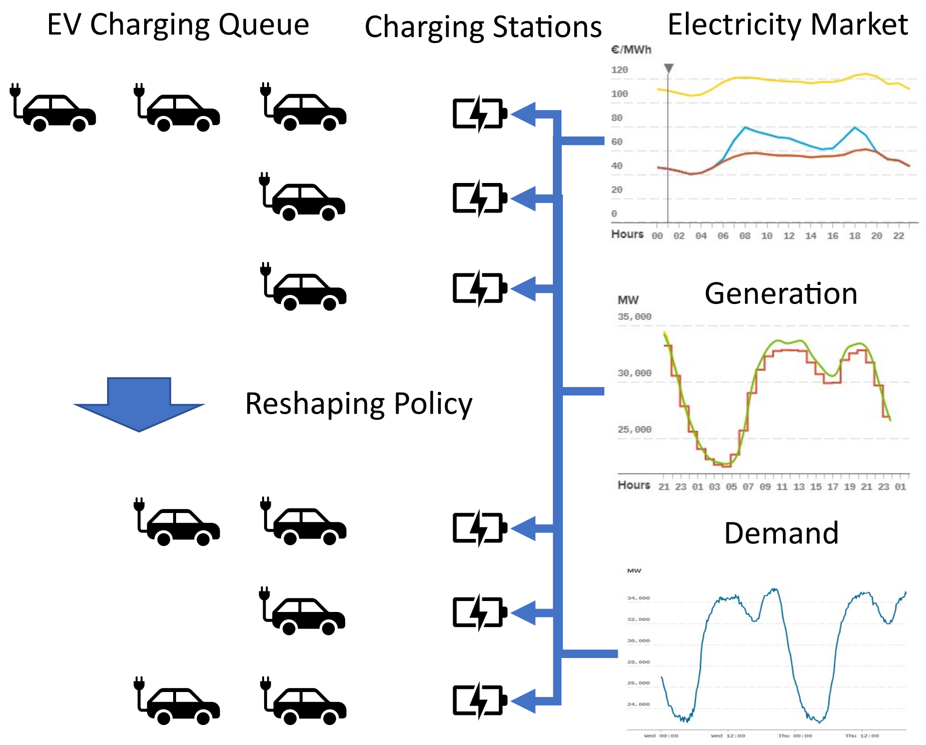


Figure 4.1: The Illustration of Demand Reshaping of EV Charging: Charging Stations May Incentivize EV Users to Request Charging at a Different Charging Station and at a Different Time Instant for Preventing Congestion.

The federation of charging stations target to maximize their reward by reshaping charging demand. To this end, charging stations must solve an optimization problem that is a function of charging prices. The optimization problem is affected by the power grid generation, demand, electricity price and quality of service as well as the EV users' private information.

The optimal charging prices are a function of EV users' charging station choices that depend on EV users' private information. Unfortunately, EV users' private information are not available to charging stations. On the bright side, the ease of access to EV charging data in charging stations enables the modelling of charging demand. This chapter leverages a learning approach to characterize the relationship between charging prices and EV arrivals. On contrary to the recent reinforcement learning based studies, the proposed technique only leverages offline data to train a deep neural network (DNN) [137, 88, 102]. This framework allows the numerical computation of optimal charging prices and is computationally much less demanding than deploying reinforcement learning.

The main contributions of this chapter are summarized as follows.

- *Synthetic Data Generation:* Despite ever-growing popularity of EVs, datasets encompassing the relationship between charging prices and EV arrivals are not easy to obtain because of the current fixed pricing policies for EV charging. To this end, this chapter designs a synthetic dataset generation platform that fuses different aspects of distinct real-world datasets. Specifically, *Colorado EV Charging Database* [47] and *Adaptive Charging Network Database* (ACN-Data) [84] are leveraged to generate distinct EV user types and charging related parameters.

- *Prospect Theory-based EV User Behaviour:* Utility maximization is often leveraged in literature to model human-based decision-making problems. Yet, the utility maximization approach relies on the strong assumption of rational users and assumes that each user targets to maximize her/his expected utility [145, 18]. The seminal work [129] showed that human behaviour rarely follows the rules as described in utility maximization framework. Prospect theory can realistically model human decision-making under uncertainty conditions. Since EV user behaviour is crucial in determining charging demand, prospect theory is utilized in this chapter, inspired by the recent studies [106, 67].

- *Learning-aided Optimization of Demand Reshaping Policy:* An optimization problem is constructed for computing the optimal demand reshaping policy that is a function of the optimal charging prices. Since the relationship between charging prices and EV arrivals is not known, a learning model is trained in place of the analytical model. Subsequently, the optimization problem is numerically solved via a gradient decent algorithm, which can back-propagate through the fixed DNN layers.

- *Federated Learning for Dispersed EV Data:* Instead of collecting all EV charging data in one edge server, the global learning model is directly trained using the local gradients from each charging station in federated manner. This approach mitigates the communication cost and proves to be as accurate as centralized training approach.

The remainder of the chapter is organized as follows. In Section 4.2, the literature review is presented. Section 4.3 illustrates the demand reshaping of EV charging and Section 4.4 presents a federated learning framework for the solution of profit maximization. Finally, Section 4.5 and 4.6 demonstrate the experimental results and provide discussions.

4.2 Related Work

Though EVs are more efficient than their internal combustion engine using counterparts, their reputation suffers from long charging duration. Given that EV charging may take more than an hour, it is natural to consider the congestion scenarios, where a charging station could be congested by EV charging demands. To mitigate these catastrophic scenarios, many papers studied the assignment of EVs to charging stations ahead of time. [144] proposes a scheduling algorithm for maximizing the charging station profit. The scheduling algorithm optimizes pricing, scheduling and admission control. To achieve these goals, [144] assumes that the utility functions of EV users are homogeneous and are known by the charging station. In [165], EV charging planning problem is solved for an EV fleet. The objectives included the placement of charging stations and hourly assignment of EVs. [28] explored and analyzed the applicability of queuing models in EV charging scheduling problems. In a more practical study, [27] analyzed EV charging congestion in Beijing based on the real-world data collected through taxis. Even though these studies presented deep insights and theoretical results, their success in real-world EV charging scenarios is at best questionable because of the underlying assumptions.

One essential information the analytical studies require is the availability of the EV users' private information, which makes it possible to analytically model their charging station choices given charging prices. To alleviate the problems arising from the lack of EV user information, some studies proposed the utilization of the data that are collectable at charging stations [51, 164, 154]. These studies utilized the underlying statistics to deduce EV user responses to different charging prices. Along a different line, more accurate techniques are also available to model human decision-making. Prospect theory is shown to accurately model human decision-making under uncertainty conditions. [106] leveraged the value and probability weight functions defined in [129] to model EV users' charging mode decisions. [67] uses prospect theory to predict when EV users charge their vehicles. Both studies corroborate more accurate predictions of their prospect theory based models in comparison to expected utility based counterparts.

The lack of accurate analytical models and the availability of huge amounts of EV charging data dictates the use of data-based approaches in computation of demand reshaping policies. To this end, reinforcement learning based techniques proved to be successful in solving difficult EV charging problems [114, 143, 137, 88, 102]. In [137] and [143], the real-time charging prices are modeled with a Markov Decision Process (MDP) with unknown transition probabilities. A model-free reinforcement learning approach is subsequently employed to determine the optimal strategy. [88] took this approach one-step further by adopting a constrained MDP to model EV arrivals. The resulting constrained MDP problem was solved using safe deep reinforcement learning. Along a similar line, [102] proposed the addition of the effect of smart grid to the cost of charging into the MDP problem. Despite huge success of reinforcement learning based techniques, a computationally efficient federated learning-based approach is adopted in this chapter.

4.3 Overview of Demand Reshaping of EV Charging

Charging is one of the biggest obstacles in the proliferation of EVs because of its long duration in comparison to its gasoline based competitors. Though the lack of EV charging infrastructure is tolerated by the small number of EVs in the traffic today, the number of EVs in the traffic grows every day. Smart allocation of EVs to limited number of charging stations may play a significant role in mitigating infrastructure costs and EV users' frustration in long queues at charging stations.

This chapter proposes a federated learning based framework for intelligent management of EV charging encompassing a federation of charging stations and selfish EV users. Charging stations share a common objective and target to maximize their profit whereas EV users selfishly aims to minimize their costs.

4.3.1 Overview of the Demand Reshaping Framework

We consider a federation of S charging stations $\mathcal{S} = \{1, 2, \ldots, S\}$ spread across an area $\mathcal{A}$. A charging station s encompasses a set of private parameters regarding the location g_s, number of chargers n_s, 24-hour ahead pricing $x_s(t)$ and hourly typical waiting duration $W_s(t)$. We further consider a set of E EV users $\mathcal{E} = \{1, 2, \ldots, E\}$ spread across the same area $\mathcal{A}$. An EV user e possesses distinct private information

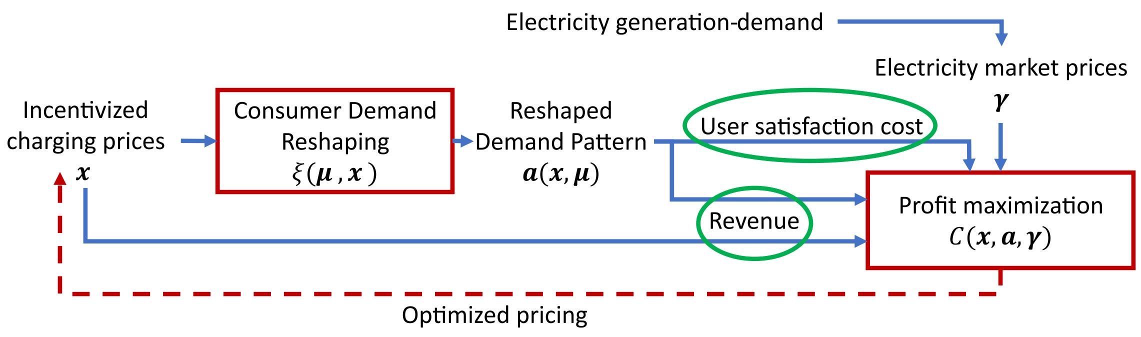

Electricity generation-demand
Electricity market prices
$\boldsymbol{\gamma}$
Incentivized charging prices
$\boldsymbol{x}$
Consumer Demand Reshaping
$\xi(\boldsymbol{\mu}, \boldsymbol{x})$
Reshaped Demand Pattern
$\boldsymbol{a}(\boldsymbol{x}, \boldsymbol{\mu})$
User satisfaction cost
Revenue
Profit maximization
$C(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\gamma})$
Optimized pricing

Figure 4.2: The Overview of the Proposed Learning Based Framework for Traffic Reshaping of EV Charging.

including her/his location ℓ_e and sensitivity to charging prices α_e^x, charging duration α_e^w and distances to charging stations α_e^d.

Charging stations share a joint cost function $C_{\mathcal{S}}$, which is a function of monetary cost due to dynamic electricity price and electricity demand-generation balance, revenue from EV charging and the quality of service due to waiting duration. Specifically, the EV charging revenue and waiting cost directly depend on EV charging demand, yet the prediction of these quantities requires the knowledge of EV users' private utility functions. On contrast to previous studies assuming the availability of EV users' utility functions, a learning-based approach is adopted in this chapter.

Charging stations are allowed to collect charging information containing the charging duration and charging time, but cannot access the private EV user information. Hence, charging stations cannot train distinct learning models for individual EV users, but can learn a single model for the overall EV charging demand given charging prices. Since different charging stations serve to a mutually exclusive set of EVs at a given hour, the EV charging data is spread across all charging stations and must be transferred to a central edge server for training. However, this approach constitutes high communication cost and further requires a large central data storage unit. To overcome these challenges, we adopt a federated learning approach, in which local gradients are transmitted to a central node. Subsequently, learning model is updated based on the collected local gradients.

As depicted in Figure 4.2, the objective of charging stations and the optimal solution rely on the accuracy of the learned model. The impact of the charging prices on cost function is two-fold: 1) the distribution and the amount of charging demand depend on charging prices, 2) charging prices directly effect the charging stations' revenue. Higher charging prices will decrease the charging demand and the revenue, whereas lower charging prices will increase the demand and waiting duration, and hence harm the quality of service. At optimal solution, high charging demand would be allocated to different charging stations to ensure minimum waiting duration and maximum revenue.

4.3.2 Modelling of EV Users' Charging Station Choices

An EV user $e \in \mathcal{E}$'s decision to charge at any charging station $s \in \mathcal{S}$ at any time instant t is determined by the solution of an expected value minimization problem, in which the expected values are computed using a distinct and private value function. The EV user e's value function V_e is expressed in terms of its distance to any charging station d_{es}, charging prices and waiting duration $W_s(t)$:

$$V_e(s,t,i)=\begin{cases} -(y_0(s,t)-y(s,t,i))^{\theta_e^-}; & y(s,t,i)<y_0(s,t) \\ \lambda_e(y(s,t,i)-y_0(s,t))^{\theta_e^+}; & y(s,t,i)\geq y_0(s,t) \end{cases} \tag{4.1}$$

where $y(s,t)$ and $y_0(s,t)$ are given as:

$$y(s,t,i) = \alpha_e^w W_s(t,i) + \alpha_e^x \int_{t+W_s(t,i)}^{t+W_s(t,i)+D_e} x_s(\tau)d\tau + \alpha_e^d d_{es} \tag{4.2}$$

$$y_0(s,t) = \alpha_e^w \beta_e^w + \alpha_e^x \beta_e^x D_e + \alpha_e^d \beta_e^d \tag{4.3}$$

In (4.1), the value function $V_e(s,t)$ is concave when $y > y_0$ and convex otherwise. This characterizes the risk-averse behaviour in cases of potential gains and risk-seeking behaviour in cases of potential losses. Observations in [129] suggest that human-beings are *more* risk-taking in cases of potential losses than in cases of potential gains, i.e., the private risk-taking parameter is selected as $\lambda_e > 2$. θ_e^- and θ_e^+ are user specific constants determining the convexity and concavity of the value function (See Figure 1 in [129]). In (4.2), the charging duration W_s is a random process and $W_s(t, i)$ denotes the ith outcome of W_s at time instant t. D_e represents the charging duration that EV user planned and d_{es} is the distance between EV e and charging station s. In prospect theory, $y_0(s, t)$ is referred to as a reference point. EV users tend to be risk-seeking above $y_0(s, t)$ and risk-averse below $y_0(s, t)$. Each EV user is assigned a different reference point for different parameters. β_e^w, β_e^x and β_e^d denote the private reference points for charging duration, charging price and distance to charging stations, respectively.

Prospect theory further proposes that probabilities are non-linearly interpreted by humans, e.g., a probability of 0.01 is perceived as much more than 0.01 whereas a probability of 0.99 is perceived as much less than 0.99 (See Figure 3 in [129]). Therefore, each EV user is assigned with a probability weighting function in the form:

$$\pi_e(p) = \frac{p^{\delta_e}}{(p^{\delta_e} + (1-p)^{\delta_e})^{1/\delta_e}} \tag{4.4}$$

where δ_e is a private parameter. Then, the expected cost induced on e by visiting s at time instant t is written as:

$$C_e(s,t) = \int \pi_e(p(i)) \left(V_e(s,t,i) + \int_t^{t+W_s(t,i)+D_e} h_e(\tau) d\tau \right) di \tag{4.5}$$

where p denotes the probability density function (pdf) of $W_s(t)$ and $h_e(t)$ denotes the cost associated with willingness of EV user e to visit s at time t, e.g., an EV user might be less willing to charge her/his car at midnight.

We here note that the EV user specific incentives can further increase the flexibility to reshape EV arrivals, but would lead to a computationally exorbitant scheduling problem on the charging station side due to the curse of dimensionality. Based on (4.5), the objective of an EV user can be stated as:

$$\{s_e^*, t_e^*\} = \underset{s\in\mathcal{S}, t\in\mathcal{T}}{\operatorname{argmin}}\ C_e(s,t) \tag{4.6}$$

where $\mathcal{T}$ denotes the continuous set of a future time-frame, e.g., next 24 hours. The optimization problem (4.6) is both analytically and computationally difficult because of the discrete variables s and since t appears in the limits of an integral without a closed-form solution. We here notice that these integrals could have been solved in closed form if the expressions $W_s(t,i)$ and $x_s(t)$ were analytically known. However, $W_s(t,i)$ also depends on the intrinsic values associated with all EV users and it is assumed to be global across EV users. This assumption is fair considering that the distribution of waiting duration can be announced online by charging stations. For instance, the popular visit times to many locations can easily be found by a simple online search. To this end, we assume EV users share the same waiting duration information of each charging station and maximize their reward accordingly. Secondly, $x_s(t)$ is announced by every charging station, and hence cannot be enforced to be in a specific function form. Therefore, we focus on the computational solution of (4.6) in this chapter.

For computationally solving (4.6), we first discretize the optimization problem as:

$$\{s_e^*, t_e^*\} = \underset{s\in\mathcal{S}, t\in\mathbb{T}}{\operatorname{argmin}}\ C_e[s,t] = \underset{s\in\mathcal{S}, t\in\mathbb{T}}{\operatorname{argmin}}\ \sum_{i\in\mathbb{I}} \pi_e(p_i)\left(V_e[s,t,i] + \sum_{\tau=t}^{t+W_s[t,i]+D_e} h_e[\tau]\right) \tag{4.7}$$

where we discretize $V_e[s,t,i]$ as:

$$V_e[s,t,i] = \begin{cases} -(y_0(s,t) - y[s,t,i])^{\theta_e^-}; & y[s,t,i] < y_0(s,t) \\ \lambda_e(y[s,t,i] - y_0(s,t))^{\theta_e^+}; & y[s,t,i] \geq y_0(s,t) \end{cases} \tag{4.8}$$

$y[s,t,i]$ is computed as:

$$y[s,t,i] = \alpha_e^w W_s[t,i] + \alpha_e^x \sum_{\tau=t+W_s[t,i]}^{t+W_s[t,i]+D_e} x_s[\tau] + \alpha_e^d d_{es} \tag{4.9}$$

We here notice that (4.7) converges to (4.6) when the duration between each discrete sample and the distance between each outcome of $W_s(t)$ goes to 0. We consider $\mathbb{T}$ to be a discrete set of time instants spanning 24 hours:

$$\mathbb{T} = \{0, \Delta t, 2\Delta t, \ldots, T\}, \tag{4.10}$$

where T denotes the time instant $23:59:59$, which is $24 \times 60 \times 60 - 1$ seconds when 0 second is taken as origin. Similarly, $\mathbb{W}_s[t]$ is the set of discrete outcomes of random variable $W_s(t)$:

$$\mathbb{W}_s[t] = \{0, \Delta\omega_s[t], 2\Delta\omega_s[t], \ldots, \Omega_s[t]\}. \tag{4.11}$$

$x_s[t]$ and $W_s[t]$ can then be defined on $\mathbb{T}$. However, it is more realistic to define $x_s[t]$ and $W_s[t]$ on $\mathbb{T}_{24}$, which is given as $\mathbb{T}_{24} = \{0, 1, 2, \ldots, 23\}$, since having EV charging pricing in a finer scale is not user-friendly.

We here recall that $W_s[t,i]$ is not an optimization variable, but rather is a realization of a random variable. On contrary, $x_s[t]$ is an optimization variable for charging stations and is announced to EV users ahead of time. Consequently, $x_s[t]$ and $W_s[t,i]$ do not necessarily follow a functional structure, e.g., $W_s[t,i]$ and $x_s[t]$ are not monotone increasing or decreasing. Therefore, each EV user performs an exhaustive search on $\mathbb{T}$ for solving (4.7).

4.3.3 Modelling of Charging Station Policies

The objective of charging stations is to maximize their profit by controlling the charging prices. Since charging prices are announced ahead of time, EV users will decide which charging station to choose, and hence may constitute congestion in particular charging stations. To this end, charging stations must optimize their charging prices to reshape the demand and to maximize their revenue from charging EVs:

$$\begin{aligned}
C(\{x_s[t]\}) &= \sum_{t=0}^{T}\sum_{s=1}^{S}(\gamma_s[t] - x_s[t])\min\{a_{st}(\{x_s[t]\}), n_s\} \\
&+\sum_{t=0}^{T}\sum_{s=1}^{S}\Bigg((n_s - \mathrm{mod}_{n_s}(r_s[t]))(e^{k_0} - 1) \\
&+ \sum_{k=k_0}^{\lfloor a_{st}+\mathrm{mod}_{n_s}(r_s[t])/n_s\rfloor + k_0} \min\{a_{st}(\{x_s[t]\}) + \mathrm{mod}_{n_s}(r_s[t]) - n_s k, n_s\}(e^{k+1} - 1)\Bigg) \\
&+\sum_{t=0}^{T}\sum_{s=1}^{S}\exp\left(x_s[t] - \zeta\right). \qquad (4.12)
\end{aligned}$$

where a_{st} is used in place of $a_s[t]$ for notation simplicity. In (4.12), $\gamma_s[t]$ denotes the market electricity price and $\{a_s[t]\}$ are the estimated arrivals for the announced charging prices $\{x_s[t]\}$ of all charging stations. ζ and the last term in (4.12) represent mean market charging price and the loss due to decrease in EV arrivals because of higher charging prices than the market average. Herein, discrete cost function is constructed since we had to use discrete utility functions for EV users. Furthermore, $r_s[t]$ represents the remainder of EVs waiting for charging from previous time instants and $k_0 = \lfloor r_s[t]/n_s \rfloor$.

Arrivals to all charging stations $a_s[t]$ can be expressed in terms of EV users' optimal solutions $\{s_e^*, t_e^*\}$ as:

$$a_s[t] = \sum_{e\in\mathcal{E}} \mathbb{1}_{\{s_e^*=s\wedge t_e^*=t\}}, \qquad \forall s \in \mathcal{S}, \forall t \in \mathbb{T}. \qquad (4.13)$$

Similarly, $r_s[t]$ is computed as:

$$r_s[t] = \sum_{\tau=1}^{t-1} a_s[\tau] + r_s[\tau] - \min\{a_s[\tau] + r_s[\tau], n_s\}. \qquad (4.14)$$

Clearly, (4.14) is a nested function, and hence it is challenging to compute a closed form solution for the optimal incentivized charging prices $\{x_s^*[t]\}$. Furthermore, computing the optimal solution via numerical methods will pose challenges when computing the gradients. Consequently, a less challenging cost function is constructed by simplifying the second term in (4.12) as:

$$\begin{aligned} C(\{x_s[t]\}) =& \sum_{t=0}^{T}\sum_{s=1}^{S}(\gamma_s[t] - x_s[t])a_{st}(\{x_s[t]\}) + \sum_{t=0}^{T}\sum_{s=1}^{S}\exp\left(a_{st}(\{x_s[t]\}) - n_s\right) \\ &+ \sum_{t=0}^{T}\sum_{s=1}^{S}\exp\left(x_s[t] - \zeta\right). \end{aligned} \tag{4.15}$$

Then the objective of charging stations becomes to solve for the optimal charging prices $\{x_s^*[t]\}$ simultaneously.

$$\{x_s^*[t]\} = \underset{\{x_s[t]\}}{\operatorname{argmin}}\ C(\{x_s[t]\}) \tag{4.16}$$

which can be rewritten in vector format as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ \big((\boldsymbol{\gamma} - \mathbf{x})^T\mathbf{a}(\mathbf{x}) + \mathbf{1}^T\exp\left(\mathbf{a}(\mathbf{x}) - \mathbf{n}\right) + \mathbf{1}^T\exp\left(\mathbf{x} - \boldsymbol{\zeta}\right)\big), \tag{4.17}$$

where $\mathbf{x}$, $\boldsymbol{\gamma}$, $\mathbf{a}$ and $\mathbf{n}$ are one-dimensional vector values and are given as:

$$\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_S^T]^T \tag{4.18}$$

$$\boldsymbol{\gamma} = [\boldsymbol{\gamma}_1^T, \boldsymbol{\gamma}_2^T, \ldots, \boldsymbol{\gamma}_S^T]^T \tag{4.19}$$

$$\mathbf{a} = [\mathbf{a}_1^T, \mathbf{a}_2^T, \ldots, \mathbf{a}_S^T]^T \tag{4.20}$$

$$\mathbf{n} = [n_0\mathbf{1}^T, n_1\mathbf{1}^T, \ldots, n_S\mathbf{1}^T]^T \tag{4.21}$$

$$\boldsymbol{\zeta} = [\zeta\mathbf{1}^T, \zeta\mathbf{1}^T, \ldots, \zeta\mathbf{1}^T]^T \tag{4.22}$$

The first term in both (4.12) and (4.15), quantifies the profit by charging $a_s[t]$ EVs. We here notice that the first term is negative when charging stations profit and is positive when charging stations are in loss. In (4.12), the number vehicles that can be charged simultaneously is capped at n_s, and hence a charging station cannot profit from EVs in the queue. On the other hand, (4.15) does not cap the maximum number of simultaneously charging EVs for the sake of differentiability. In a similar manner, (4.12) precisely computes the number of EVs waiting in the queue and can determine the EV waiting cost with high accuracy whereas (4.15) omits the cap n_s, but assigns a higher cost for exceeding n_s. Consequently, both cost functions attain their optimal solutions when the computed pricing constitutes exactly n_s number of EVs charging simultaneously. This solution does not incur any EV waiting cost, but maximizes the revenue assuming $x_s[t] > \gamma_s[t]$.

Clearly $\gamma_s[t]$ varies for different charging stations and for different time instants, which complicates the problem even further. The reason behind this construction is the varying nature of the demand across different regions in cities. Depending on the instantaneous power load, service providers may incentivize certain charging stations in pursuit of more power load.

(4.15) possesses an important differentiability advantage over (4.12). Hence, we can employ a gradient based numerical solver to solve for $\mathbf{x}$ if the relation between $\mathbf{a}$ and $\mathbf{x}$ could be established. To this end, the optimal solution $\mathbf{x}^*$ depends on $\mathbf{a}$ and the optimal solutions to (4.7). Unfortunately, charging stations cannot compute $\{s_e^*, t_e^*\}, \forall e \in \mathcal{E}$ as charging stations cannot access the private information of EV users. To overcome this challenge, we propose a learning-based approach, in which the mapping from charging prices $\mathbf{x}$ to EV arrivals $\mathbf{a}$ are learned from previous data. Once this mapping is learned from data, the mapping can be utilized to predict the EV arrivals for different selections of $\mathbf{x}$. The next section provides the complete details of the proposed framework and solution.

4.4 A Federated Learning Framework for Profit Maximization

In this section, we devise a learning framework for computing EV charging prices. Since a subset of EV charging stations are often governed by a single company, we assume the EV charging stations act collaboratively in this chapter. To this end, EV charging stations can solve the optimization problems (4.7) and (4.16) collectively. However, the former needs to be solved as a learning problem rather than a numerical optimization problem. A careful investigation reveals that the input and output of the learning problem are both of dimensions $T \times S$. Since the dimensions of the learning problem may increase significantly with the increase in charging station numbers, it is desirable to split the learning problem into smaller learning problems. To this end, we employ a federated learning approach.

4.4.1 Learning-based Modelling EV Arrivals

In order to solve (4.13) by using the available information to charging stations, a deep neural network (DNN) can be trained using a dataset $\mathcal{D}$ containing $(\mathbf{x}, \mathbf{a})$ pairs as data samples. Herein, the charging prices $\mathbf{x}$ and EV arrivals $\mathbf{a}$ serve as the input and outputs to the learning model, respectively. The learning model can be represented by $\xi(\mathbf{x}, \boldsymbol{\mu})$, where ξ and μ denote the model and the trainable parameters, respectively. Considering each entry in a single data sample $(\mathbf{x}_l, \mathbf{a}_l) \in \mathbb{R}$ and the objective is inference, mean-squared error (MSE) is selected as the cost function for training. The model ξ can be trained to minimize the MSE between the mini-batches of $\mathbf{a}$ and $\xi(\mathbf{x}, \boldsymbol{\mu})$ as:

$$\text{MSE}(\xi(\boldsymbol{\mu}, \mathbf{x}), \mathbf{a}) = \frac{1}{M \times L} \sum_{m=1}^{M} \sum_{l=1}^{L} (\mathbf{a}_l^{(m)} - \xi(\boldsymbol{\mu}, \mathbf{x}_l^{(m)}))^2 \qquad (4.23)$$

where $(.)_l^{(k)}$ denotes the kth sample of a mini-batch, $L = ST$ and M is the batch size. Clearly, the MSE cost is minimized when $\xi(\boldsymbol{\mu}, \mathbf{x}) = \mathbf{a}$ for any batch of data samples. To train ξ for obtaining optimal parameters $\boldsymbol{\mu}$, the stochastic gradient descent is performed in recursion using the following set of update rules:

$$\nabla_{(j)} = \frac{\partial \mathrm{MSE}(\xi(\boldsymbol{\mu}_{(j)}, \mathbf{x}), \mathbf{a})}{\partial \boldsymbol{\mu}_{(j)}} \tag{4.24}$$

$$\boldsymbol{\mu}_{(j+1)} = \boldsymbol{\mu}_{(j)} - \kappa \nabla_{(j)} \tag{4.25}$$

where κ and $\nabla_{(j)}$ denote the learning rate and the gradients at iteration j, respectively. Clearly, it is necessary to collect all data samples at a central node to perform (4.24) and (4.25). Unfortunately data samples are spread across different charging stations and it would impose high communication and time cost on the network.

Alternatively, the model update can be performed in a federated manner. Each charging station can compute a local gradient and transmit it to a central node, where all local gradients are used to update the global learning parameters:

$$\nabla_{(j),s} = \frac{\partial \mathrm{MSE}(\xi(\boldsymbol{\mu}_{(j)}, \mathbf{x}_{(s)}), \mathbf{a}_{(s)})}{\partial \boldsymbol{\mu}_{(j)}}, \qquad \forall s \in \mathcal{S} \tag{4.26}$$

$$\boldsymbol{\mu}_{(j+1)} = \boldsymbol{\mu}_{(j)} - \kappa \sum_{s=1}^{S} \nabla_{(j),s} \tag{4.27}$$

where $\mathbf{x}_{(s)}$ and $\mathbf{a}_{(s)}$ denote the data samples used in local charging station s. After training the model parameters for many iterations, the final model can predict EV arrivals at each charging station, $\mathbf{a}$, for the given charging prices, $\mathbf{x}$, with high accuracy.

4.4.2 Data Collection and Synthetic Database Generation

The accuracy of the learning model ξ heavily depends on the quality of the quality of the dataset used to train it. To this end, a simple data collection framework is devised in this section.

The objective of the learning model is to learn the relationship between charging prices and EV arrivals. Since data collecting needs to be cooperative, each charging station takes turns to randomly set a price for any time instant and for all charging stations. Subsequently, the selected charging station broadcasts all the charging prices for the next 24-hour. The next day, hour-by-hour EV arrivals are recorded at every charging station and this data is stored at the corresponding charging station. Since different charging stations take turns to set prices, different data samples are stored in different charging stations.

Unfortunately, this data collection phase may take too long for the scale of days. Therefore, we adopt a different strategy to test our framework. Two real datasets; 1) *Electric Vehicle Charging Station Data* and 2) *ACN-Data* are utilized to generate a synthetic dataset [47, 84].

Electric Vehicle Charging Station Data, contains the EV visit history across 24 different EV charging stations. The dataset includes the charging start, completion and end times, dates and the price. However, EVs are not assigned IDs, and hence it is not possible to track the charging record of each individual EV. On the other hand, it provides the essential information regarding the EV charging demand across different locations. Since the charging price is fixed across all charging stations and time, *Electric Vehicle Charging Station Data* can be used to estimate the locations of users. The second dataset, *ACN-Data*, consists of a single charging station in Caltech University and provides the charging history of EVs in the vicinity of Caltech. *ACN-Data* contains information regarding charging start, completion and end times as well as the user IDs of each EV. Therefore, the user EV charging characteristics can be extracted from *ACN-Data*.

The *synthetic database* is essentially the fusion of both databases. *ACN-Data* possesses 410 different, identified EV users and their EV charging history for almost 3 years of duration. These 410 users are used to define 410 different EV user types. Each owner type can be considered as a random variable with certain attributes, including EV charging duration and EV charging dates. EV charging duration is represented by a random variable ρ_T and its probability density function is determined according to the histogram of its history in *ACN-Data*. Similarly, EV charging dates is modeled as a Bernoulli random variable, ρ_V, with the success probability of p_v:

$$p_v = \frac{\text{number of EV charging station visits in 3 years}}{365 \times 3}, \tag{4.28}$$

where the information in nominator is obtained from *ACN-Data*.

Subsequently, charging station based statistics are also extracted from *Electric Vehicle Charging Station Data*. In particular, the number of average visits for each charging station and their locations are extracted as EV charging station attributes. Along with the random variables generated using *ACN-Data*, the number of average visits for each charging station and their locations are used to generate EV user samples. Algorithm 5 illustrates the method used to generate the *synthetic database*.

First a Poisson random variable is created for each charging station with mean value being equal to the number of average visits. Next, a sample ζ is drawn and ζ many EV users are generated. Each EV user is randomly assigned to an EV user type according to a discrete uniform distribution. To locate EV users, a Poisson Point Process (PPP) approach is adopted. Each EV user is randomly placed within a disk centered around the corresponding charging station. This process is repeated for all charging stations. Our experiments demonstrated that the average number of EV users generated using this process is around 5000 users.

To generate a single sample, $x_s[t], \forall s \in \mathcal{S}$ are generated randomly. The charging prices are also randomly generated by sampling from 576 i.i.d. uniform random variables. As described above, each user type is linked to two random variables, EV charging duration and EV charging dates. Subsequently, a single sample pair is drawn from these two random variables and assigned to the EV user. In the final step, the corresponding arrivals $\mathbf{a}$ are computed by solving (4.7) with exhaustive search. The whole process is repeated to generate more samples to form the *synthetic database*. More implementation details and a samples database are provided in [38].

4.4.3 Optimization of Charging Prices

Once a DNN model is trained, this model can be leveraged to compute optimal charging prices to constitute optimal EV arrivals. The solution to (4.17) can be obtained by plugging the DNN model in place of $\mathbf{a}(\mathbf{x})$:

$$\begin{aligned}\mathbf{x}^* =& \underset{\mathbf{x}}{\operatorname{argmin}} C(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} (\boldsymbol{\gamma} - \mathbf{x})^T \mathbf{a}(\mathbf{x}) + \mathbf{1}^T \exp\left(\mathbf{a}(\mathbf{x}) - \mathbf{n}\right) \\ =& \underset{\mathbf{x}}{\operatorname{argmin}}\ (\boldsymbol{\gamma} - \mathbf{x})^T \xi(\boldsymbol{\mu}, \mathbf{x}) + \mathbf{1}^T \exp\left(\xi(\boldsymbol{\mu}, \mathbf{x}) - \mathbf{n}\right) \end{aligned} \tag{4.29}$$

which can be numerically computed by recursive application of a gradient descent method:

$$\nabla_{(i)} = \frac{\partial C(\mathbf{x}_{(i)})}{\partial \mathbf{x}_{(i)}} \tag{4.30}$$

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \kappa^\dagger \nabla_{(i)} \tag{4.31}$$

where $\mathbf{x}_{(i)}$ and $\kappa^\dagger$ denote the charging prices at gradient descent iteration i and descent rate, respectively. Expanding (4.30), we obtain:

$$\nabla = \frac{\partial C(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \left((\boldsymbol{\gamma} - \mathbf{x})^T \xi(\boldsymbol{\mu}, \mathbf{x}) + \mathbf{1}^T \exp\left(\xi(\boldsymbol{\mu}, \mathbf{x}) - \mathbf{n}\right)\right)}{\partial \mathbf{x}}, \tag{4.32}$$

which contains the terms ${}^{\partial \xi(\boldsymbol{\mu}, \mathbf{x})}/_{\partial \mathbf{x}}$. We here notice that these gradients can be computed via back-propagation on $\mathbf{x}$ as we did on $\boldsymbol{\mu}$ during training the learning model. Therefore, gradient descent algorithm can compute the optimal $\mathbf{x}^*$ numerically.

4.5 Numerical Experiments

For illustrating the effectiveness and success of the proposed framework, we perform numerical experiments. In all experiments, 24 charging stations are used, and

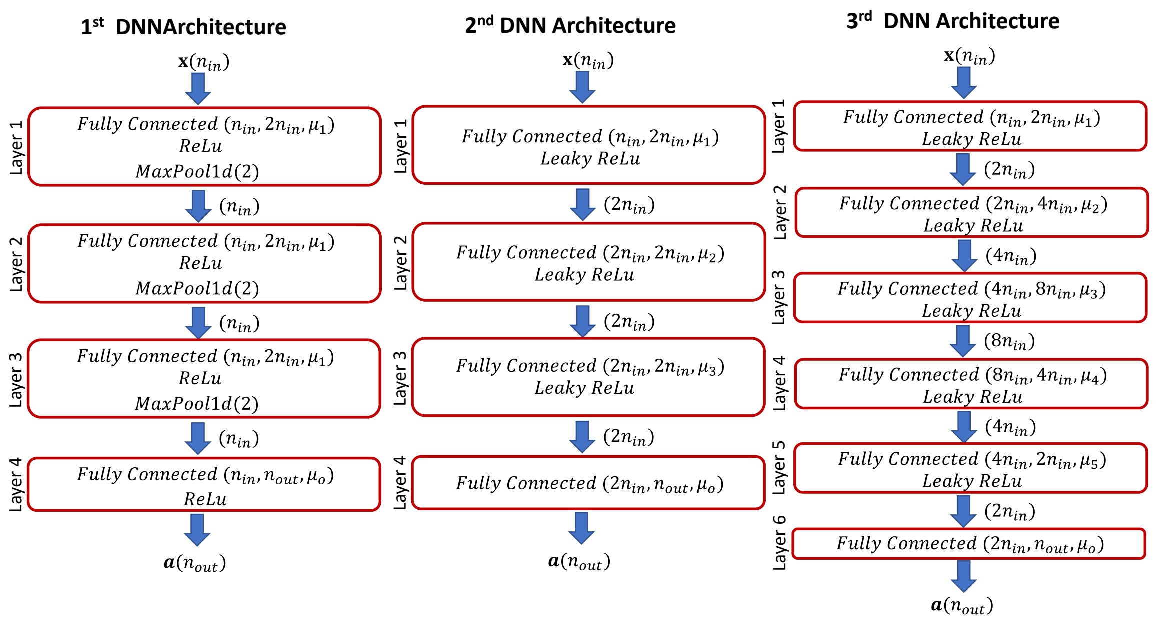

1st DNNArchitecture
2nd DNN Architecture
3rd DNN Architecture
$\mathbf{x}(n_{in})$
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Fully Connected $(n_{in}, 2n_{in}, \mu_1)$ ReLu MaxPool1d(2)
(n_{in})
Fully Connected (n_{in}, n_{out}, μ_o) ReLu
Fully Connected $(n_{in}, 2n_{in}, \mu_1)$ Leaky ReLu
Fully Connected $(2n_{in}, 2n_{in}, \mu_2)$ Leaky ReLu
Fully Connected $(2n_{in}, 2n_{in}, \mu_3)$ Leaky ReLu
Fully Connected $(2n_{in}, n_{out}, \mu_o)$
$(2n_{in})$
Fully Connected $(2n_{in}, 4n_{in}, \mu_2)$ Leaky ReLu
Fully Connected $(4n_{in}, 8n_{in}, \mu_3)$ Leaky ReLu
Fully Connected $(8n_{in}, 4n_{in}, \mu_4)$ Leaky ReLu
Fully Connected $(4n_{in}, 2n_{in}, \mu_5)$ Leaky ReLu
$(4n_{in})$
$(8n_{in})$
$\boldsymbol{a}(n_{out})$

Figure 4.3: Schematics of All 3 Neural Network Models Used in Experiments.

hence the size of charging prices and EV arrivals is selected to be 24×24. Learning rate for training learning models and the batch size are 5×10^{-5} and 256. We run experiments for 500 iterations. We used 8 charging stations for federated learning as default unless stated otherwise. In some experiments we compared different DNN models. To this end, we used 3 different models: 1) *1st DNN Architecture* is a 4-layer neural network with same parameters in first 3 layers and with ReLu and pooling activation; 2) *2nd DNN Architecture* is also a 4-layer neural network, but with different parameters at every layer, and with Leaky ReLu activation; 3) *3rd DNN Architecture* is a 6-layer neural network with Leaky ReLu activation. These architectures are demonstrated in Figure 4.3. Unless stated otherwise, *2nd DNN Architecture* is used in the experiments.

For the synthetic database generation, each EV user is randomly assigned with a set of parameters to describe their private value function. For a EV user $e \in \mathcal{E}$, we

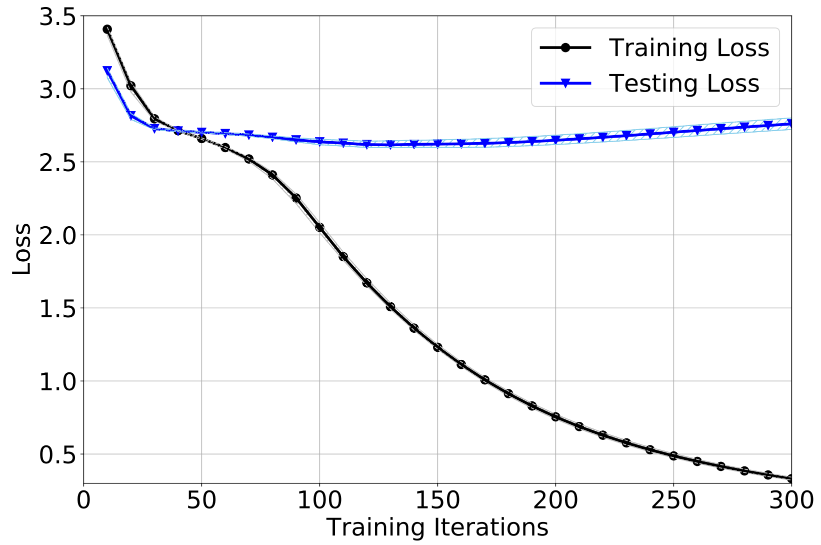


Figure 4.4: The Evolution of Training and Testing Loss During Model Training. Overfitting Occurs After 160 Iterations.

used the following random variables to randomly generate corresponding parameters:

$$\theta_e^- \sim U(0.6, 0.7) \qquad \theta_e^+ \sim U(\theta_e^-, 0.8) \qquad \lambda_e \sim U(2, 2.5)$$

$$\alpha_e^w \sim U(1, 3) \qquad \alpha_e^x \sim U(1, 3) \qquad \alpha_e^d \sim U(10, 30)$$

$$\beta_e^w \sim U(0, 0.5) \qquad \beta_e^x \sim U(0, 2) \qquad \beta_e^d \sim U(1000, 2000)$$

$$\delta_e \sim U(0.9, 1) \qquad h_e[t] \sim \mathcal{N}(\cos(24/90(t-2)) + 1, 1)$$

To simulate real-life electricity market conditions, the data from ESIOS is leveraged [49]. Accordingly, electricity market price is modeled as a random variable:

$$\gamma_s[t] \sim \mathcal{N}(\mathbb{E}[\gamma_s[t]], \mathrm{Var}[\gamma_s[t]]), \tag{4.33}$$

where $\mathbb{E}[\gamma_s[t]]$ and $\mathrm{Var}[\gamma_s[t]]$ are computed from a year long data collected from ESIOS.

Eliminating the Effects of Overfitting

In the first experiment setup, the training convergence of the learning model is explored. Overfitting may occur because of limited number of data samples used in experiments and large DNN model. To overcome this problem, we adopt an early stopping approach. To this end, we perform 10 independent training runs on the 1st DNN architecture and explore after which iteration overfitting occurs. As illustrated in Figure 4.4, overfitting is observed after 160 iterations, at which point the testing loss starts to grow while training loss continues to decrease. To overcome overfitting, early stopping is deployed and different DNN architectures are explored.

Effects of Number of Stations Participating in Federated Learning

The second experiment explores the convergence of federated learning under different number of participating charging stations. We test the convergence for 4, 8 and 24 participating charging stations. Again, 10 numerical experiments are performed for every setup and the results are illustrated in Figure 4.5. Results suggest that the performance of the learning model does not vary too much with respect to the number of charging stations participating in federated learning as expected. However, using 8 charging stations seems to provide slight benefit in terms of loss minimization. Consequently, the experiments used 8 charging stations herein for optimal performance unless stated otherwise.

Convergence Comparison for Different DNN Architectures

Different architectures can constitute superior performance in comparison to their counterparts and the relative performance may rely on the dataset itself. In this experiment setup, 3 different DNN architectures are trained to learn the relationship between $\mathbf{x}$ and $\mathbf{a}$ for exploring which DNN architecture best fits to the synthetic dataset. We have implemented the architectures shown at Figure 4.3 and the training convergence of these architectures are illustrated in Figure 4.6. The results indicate that deploying a larger DNN helps to improve both convergence rate and minimum

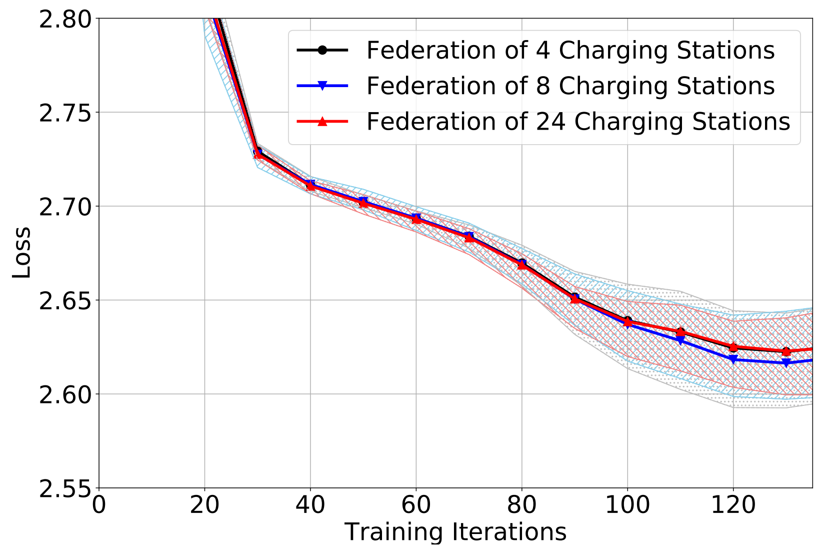


Figure 4.5: The Evolution of Testing Loss over Training Iterations for Different Numbers of Charging Station Participation.

loss at the cost of additional memory utilization and computational power. On the other hand, the convergence rate for the first DNN architecture is significantly slow in comparison to other 2 architectures even though the first and second DNN architectures have the same number of layers. However, the first neural network leverages the same parameters across its first three layers, and hence utilizes the minimum memory among all architectures.

Convergence of Numerical Optimization

This experiment setup targets to demonstrate the convergence of numerical optimization of charging prices. We expect to observe profit to stabilize as the optimization proceeds. Clearly, charging stations desire to increase charging prices to maximize their revenues while trying to minimize their electricity costs, waiting duration and long term customer loss due to overpricing.

We leverage the fully trained learning model based on 2nd DNN architecture and plug it in place of $\xi(\boldsymbol{\mu}, \mathbf{x})$ to numerically compute optimal charging prices. Descent

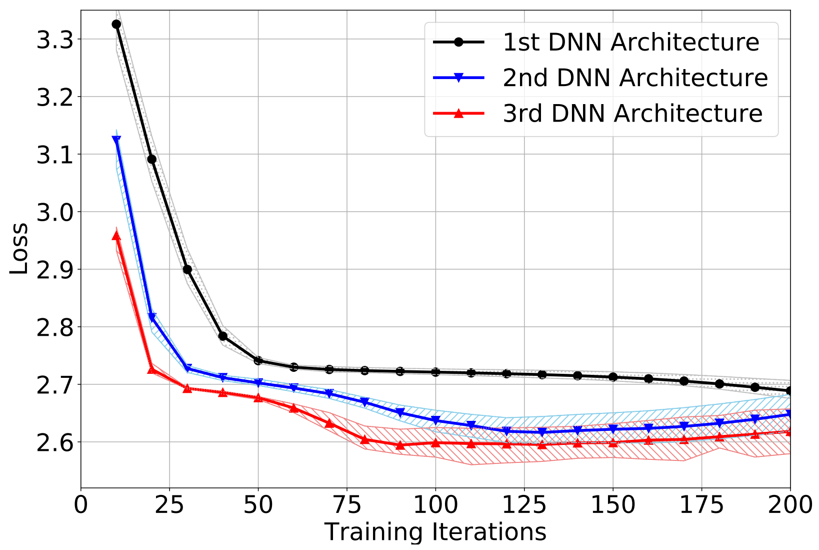


Figure 4.6: The Evolution of Testing Loss over Training Iterations for Different Neural Network Architectures.

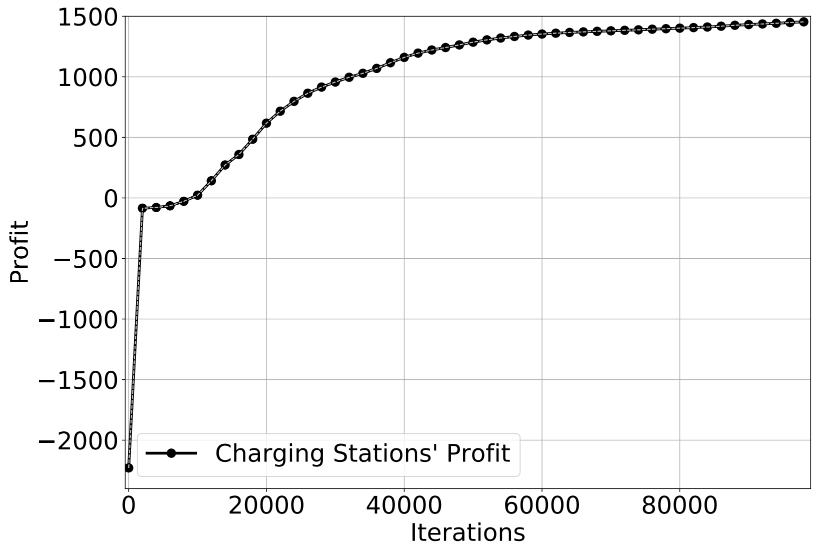


Figure 4.7: The Evolution of Charging Stations' Profit over Optimization Iterations.

rate $\kappa^{\dagger}$ is selected as 10^{-3}. The convergence of the numerical optimization is illustrated in Figure 4.7. As expected, total cost converges to a local optimal solution. After the local optimal point is reached, increasing charging prices incurs higher EV charging queues, and hence mitigates user experience.

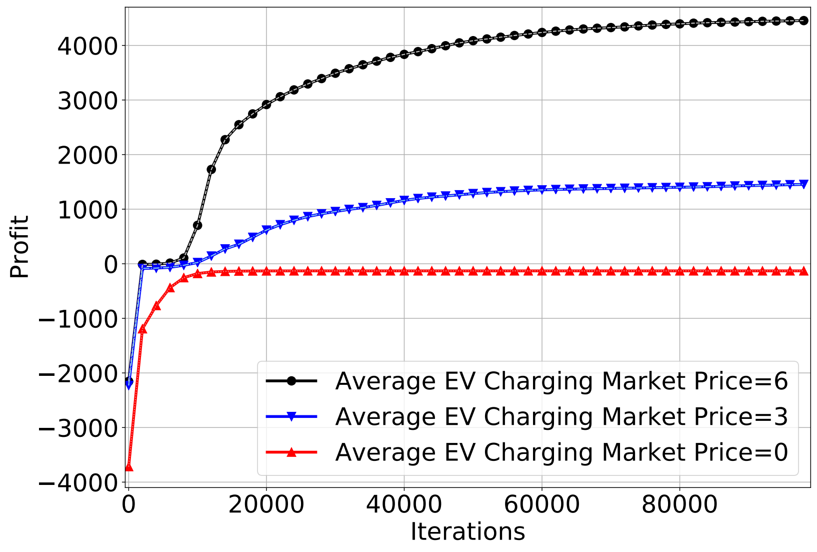


Figure 4.8: The Evolution of the Profit of Charging Stations for Different Average Market Prices of EV Charging.

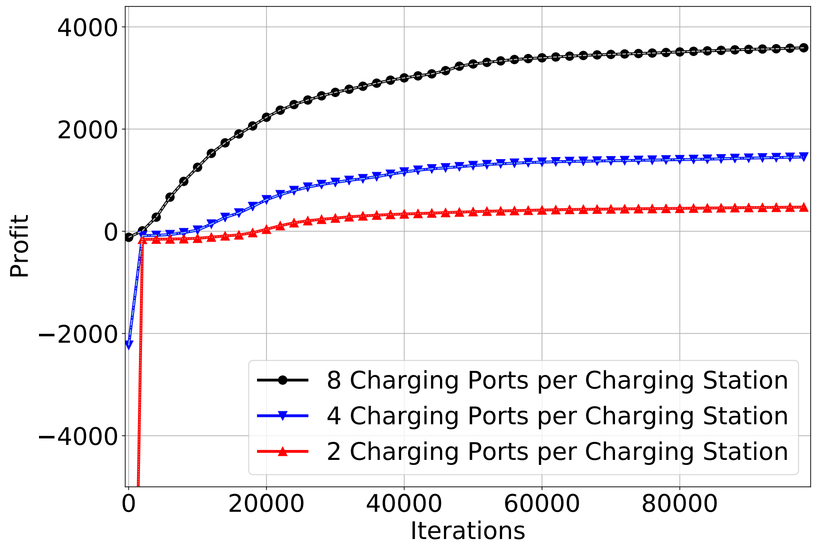


Figure 4.9: The Evolution of the Profit of Charging Stations for Different Numbers of EV Charging Ports at a Single Charging Station.

The Impact of EV Charging Market Prices on Optimal Solution

A federation of charging stations will want to collectively increase prices if there is not competition knowing that EVs must be charged at some point. However, in a real-world market, many federations of charging stations will compete. To incorporate

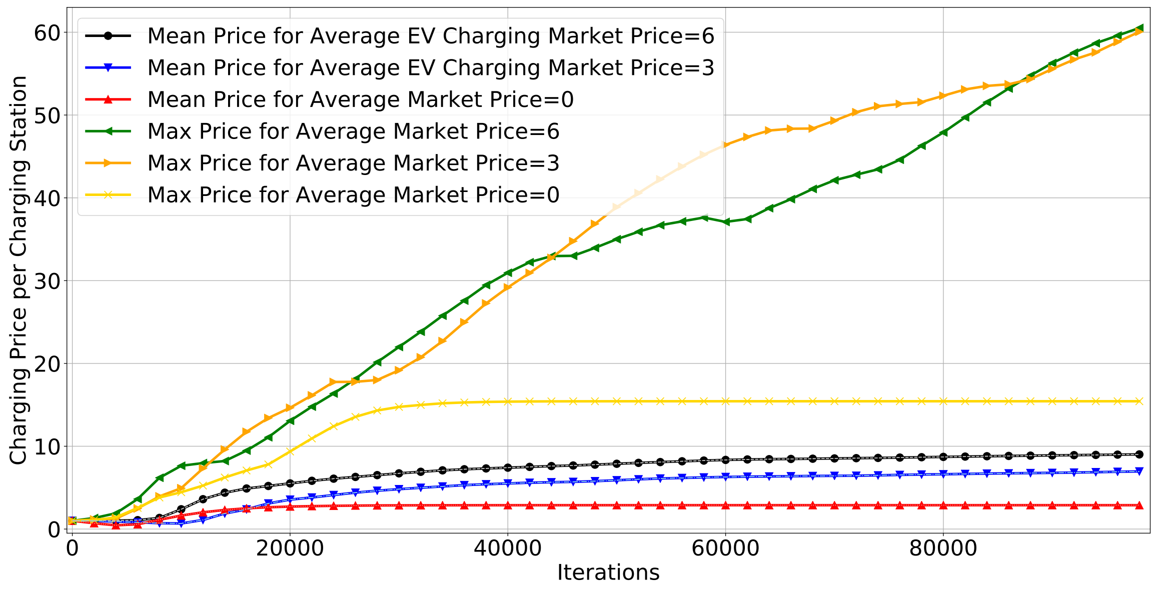


Figure 4.10: The Evolution of the Incentivized Charging Prices for Different Average Market Prices of EV Charging. Max and Mean Prices Denote the Maximum and Average Charging Prices Among All Charging Stations at Any Time of the Day.

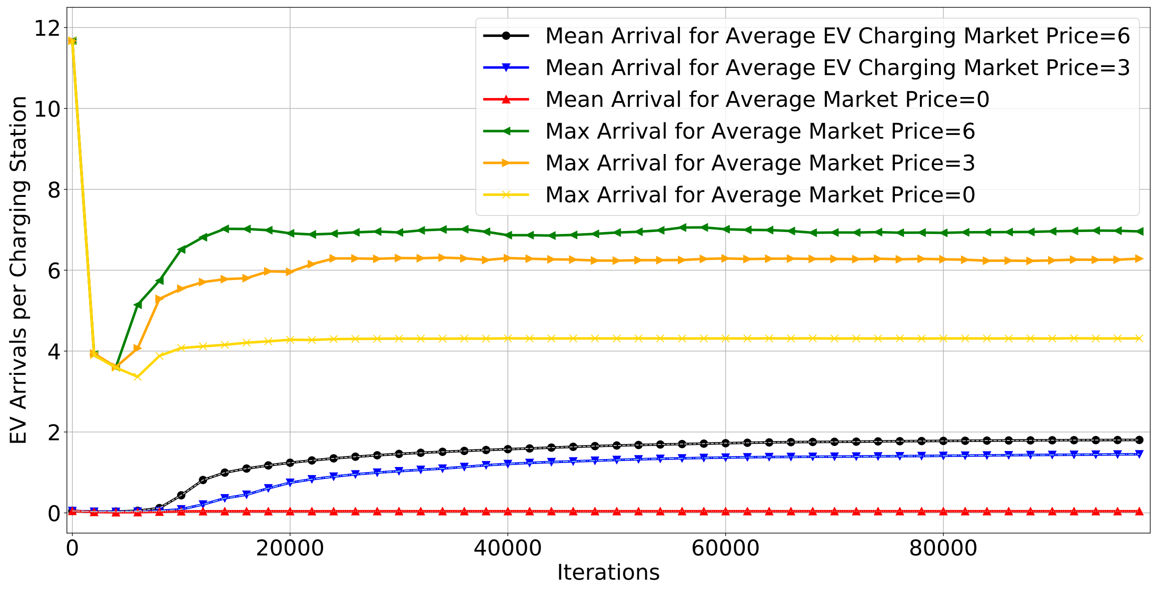


Figure 4.11: The Evolution of the EV Arrivals at a Single Charging Station for Different Average Market Prices of EV Charging. Max and Mean Arrivals Denote the Maximum and Average EV Arrivals Among All Charging Stations at Any Time of the Day.

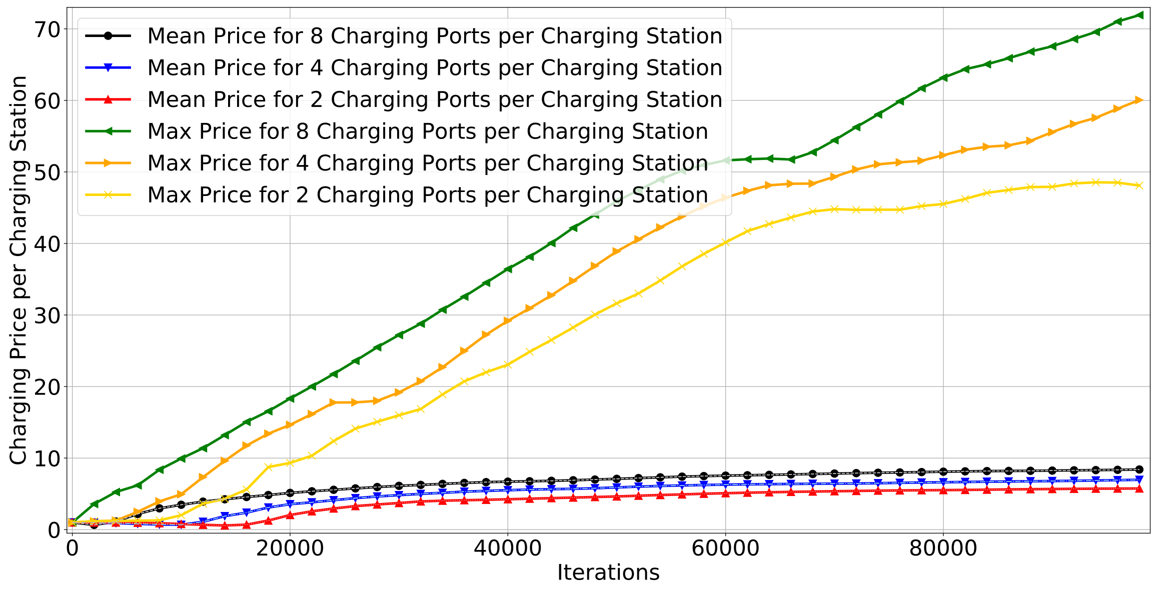


Figure 4.12: The Evolution of the Incentivized Charging Prices for Different Numbers of EV Charging Ports at a Single Charging Station. Max and Mean Prices Denote the Maximum and Average Charging Prices Among All Charging Stations at Any Time of the Day.

this effect in this chapter, overpricing cost is added to the cost of charging stations. To this end, increasing charging prices will decrease charging demand in long term. To prevent the loss in charging demand, charging stations should offer competitive prices with the market.

To illustrate the effect of overpricing, we run experiments with different average market prices. The demonstrated results in Figure 4.8, 4.10 and 4.11 indicate that the profit of the federation of charging stations significantly increase along with the average charging market price. In particular, charging stations try to minimize EV arrivals if the average charging market price is 0, indicating that charging stations are in loss and they try to minimize their losses rather than trying to maximize their profits.

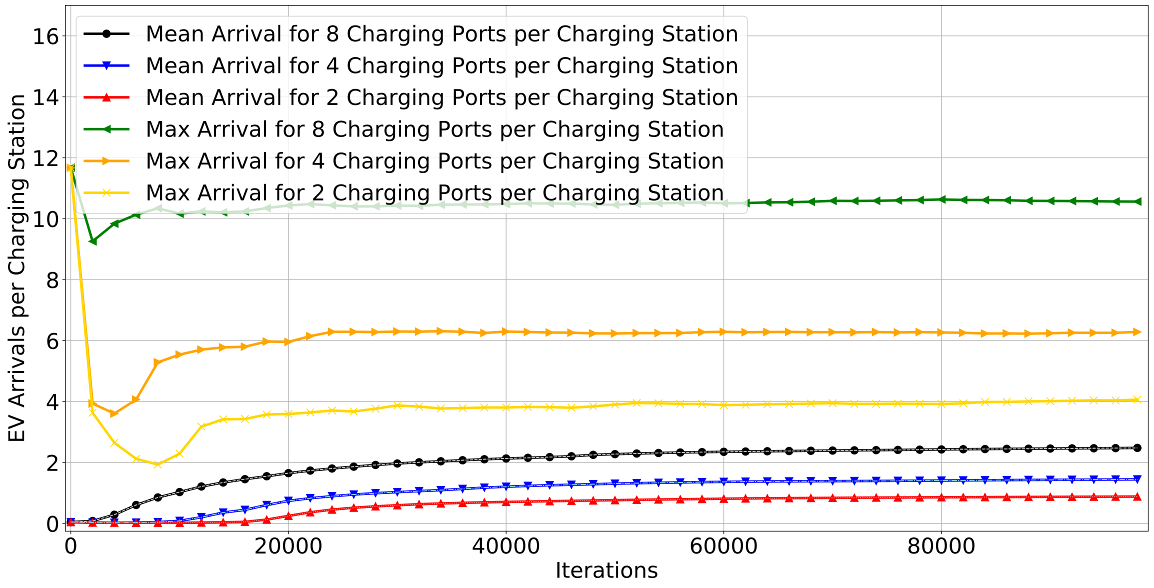


Figure 4.13: The Evolution of the EV Arrivals for Different Numbers of EV Charging Ports at a Single Charging Station. Max and Mean Arrivals Denote the Maximum and Average EV Arrivals Among All Charging Stations at Any Time of the Day.

The Impact of the Number of Charging Ports at Each Charging Station on Optimal Solution

Lastly, we illustrate the impact of the number of charging ports at each charging station on the optimal charging prices. To this end, we try 3 different scenarios with 2, 4 and 8 charging ports at every charging station. Clearly, the queue of EVs waiting for charging will significantly increase if there are more charging ports. Therefore, EV waiting cost will decrease significantly, which can be monetized by increasing charging prices, i.e., charging stations may profit from higher quality of service.

Results demonstrated in Figure 4.9, 4.12 and 4.13 show that charging stations profit significantly if they deploy more charging ports at every charging station. We observe that charging stations increases charging prices, but still attract more EVs because of the increased quality of service.

4.6 Conclusion

The problem of computing optimal demand reshaping policy under lack of EV user information is studied in this chapter. In the lack of private EV user information, a data based approach is adopted, in which a deep learning model is trained by using the collective information from charging station. Since data is spread across many charging stations, a federated learning framework is deployed in the wake of expensive computation and communication costs. In order to model EV user behaviour, the seminal Prospect Theory is leveraged alongside the real-life databases. Subsequently, a synthetic database generation algorithm is designed and implemented. The trained learning model on the synthetic database is used in optimizing the charging prices. The optimal prices ideally minimize the queue size and maximize the revenue of charging stations, constituting benefits for both EV users and charging stations. Numerical experiments corroborate the effectiveness of the proposed techniques.

Algorithm 5 Synthetic Data Generation

```
1: Inputs: Electric Vehicle Charging Station Data, ACN-Data;
2: Outputs: Synthetic database;
3: EV user types: $J = []$
4: for each EV user in ACN-Data, $e$ do
5:     Compute the pmf of charging duration random variable $\rho_{T_e}$
6:     Model charging probability as Bernoulli and compute $p_{v,e}$, and add to $J$
7: end for
8: EV user list: $K = []$, EV user locations: $L = []$
9: for each EV charging station in Colorado EV database, $s$ do
10:    $\rho_{\lambda_s}$ = number of average visits
11:    Sample the number of EV users $\zeta_s \sim \rho_{\lambda_s}$
12:    Uniformly sample a EV user type from $J$ and add it to $K$
13:    Randomly draw (PPP) EV location around the charging station and add to $L$
14: end for
15: Create the database $\mathcal{D} = []$
16: while Maximum number of samples is not reached do
17:    Sample $24 \times 24$ elements of $\mathbf{x}$ from $24 \times 24$ i.i.d. uniform random variables.
18:    Arrival vector $\mathbf{a} = \mathbf{0}$
19:    for every $k$ in $K$ do
20:        Draw a sample $j$ from $J[k]$
21:        Solve (4.7) using exhaustive search to obtain $s_k^*, t_k^*$
22:        $\mathbf{a}_{s_k^*, t_k^*} \leftarrow \mathbf{a}_{s_k^*, t_k^*} + 1$
23:    end for
24:    Add $\{\mathbf{x}, \mathbf{a}\}$ to $\mathcal{D}$
25: end while
```

Chapter 5

THE IMPACT OF TRAFFIC INFORMATION AGE ON CONGESTION MITIGATION

5.1 Introduction

Traffic engineering techniques have been developed for efficient use of network resources in conventional OSPF-based networks operating under tight quality of service (QoS) constraints [53, 19, 152, 160, 146, 60, 53, 52]. Traffic engineering policies can be determined based on the traffic measurements collected throughout the network by a centralized controller. Unfortunately, out-of-date traffic measurements and rapid fluctuations in traffic demand constitute an inaccurate global view of the network at the controller, diminishing the effectiveness of traffic engineering techniques. Further, finding optimal OSPF link weights is an NP-hard problem [36], and local search algorithms [53], which are slow and unresponsive to a changing network environment, often have to be implemented. For better utilization of network resources, contemporary networks need to quickly adapt to changing network environment and satisfy tight timing constraints while considering the ages of traffic measurements.

In the existing literature, it is often assumed that traffic demand is quasi-stationary and traffic measurements are consistent (see [53] and the references therein). Nevertheless, the explosive growth of mobile devices and bandwidth-hungry applications in the last decade have led networks to experience rapidly changing traffic demands. As a consequence, contemporary networks must be reconfigured frequently since otherwise the routing settings in use would be obsolete in a short span of time. On the other hand, reconfigurations incur routing instability in the network, and hence must be avoided if possible. Besides, the consistency assumption implies that the measurements are fresh and synchronously gathered, and hence the measurements accurately reflect the current traffic state of the network. Unfortunately, the measurements do not always have the same time-stamp since router clock synchronization cannot be achieved at all times [50]. In addition, the inter-arrival times of the traffic measurements may randomly range from several hundreds of milliseconds up to a hundred seconds [50]. In light of these practical challenges, traffic engineering techniques must take the ages of traffic measurements into account when computing new network settings.

Traffic engineering techniques often aim at network congestion minimization so as to improve network utilization through direct control of link flows [19, 60, 48]. Even though the corresponding traffic engineering problem is solvable in polynomial time, link flows cannot be implemented under conventional OSPF-based networking architecture [134, 135, 30]. On the other hand, the traffic engineering formulation for congestion minimization, in terms of OSPF link weights, is proven to be NP-hard [36]. Consequently, the applicability of traffic engineering in OSPF-based networks is either computationally too intensive and time consuming [53, 158] or require manipulations of OSPF routing to overcome the NP-hard nature of the traffic engineering problem [134, 48]. Along a different line, software defined networking (SDN) allows direct implementation of link flows in the network, and hence the congestion minimization problem can be solved under tight timing constraints [100, 6, 5]. However, centralized routing suffers from slow link fail-over, which degrades network robustness compared to distributed routing techniques [130, 6, 5].

In consideration of timing constraints and robustness, this chapter adopts centralized load-sensitive routing, and employs a flexible controller to assist distributed routing for improving network utilization. The employed controller iteratively updates OSPF weights and deploys these weights on legacy routers. The legacy routers then determine their own forwarding tables using the OSPF weight set announced by the controller and forward traffic accordingly. In consideration of tight timing constraints, the controller employs a lightweight, load-sensitive OSPF weight update algorithm, which cannot minimize, but only mitigates network congestion. Consequently, network benefits from robustness and flexibility at the cost of optimality, while satisfying tight timing constraints.

The main contributions of this chapter can be summarized as follows.

- *Traffic-adaptive network reconfiguration:* Under aforementioned observations, this chapter addresses the above problem in dynamic congestion mitigation for better network utilization. In particular, we formulate the iterative online decision-making problem for link weight reconfiguration, taking into account the impacts of network congestion, reconfiguration instability and measurement age. Subsequently, we execute a myopic policy to solve this problem and mitigate congestion over iterations.

- *Comprehensive measurement collection model:* We relax the widely adopted quasi-stationary traffic assumption on link state routing protocols to account for non-stationary traffic demands. In spite of this, we establish a tractable autoregressive model and further incorporate the notion of age of information to construct a comprehensive, yet flexible, traffic measurement collection model for a centralized controller that is capable of computing traffic engineering rules.

- *Measurement age-aware metrics:* In order to better quantify the impact of congestion and measurement age, we employ the following two distinct metrics, and derive their corresponding measurement and age-aware single stage predictors: First metric utilizes only first order statistics (mean of each component), whereas the second incorporates second order statistics (variances of components) to further encapsulate the impact of uncertainty due to measurement age. To the best of our knowledge, this is the first study that mathematically evaluates the impact of measurement ages on network utilization.

The rest of the chapter is organized as follows. We introduce the system overview and operation in Section 5.2, and explain the measurement age-aware traffic monitoring in Section 5.3. The employed load sensitive OSPF link weight update algorithm is presented in Section 5.4, and the decision-making problem for the adaptive link weight reconfigurations is formulated in Section 5.5. The overall system is implemented and tested for various experiment settings through simulations in Section 5.6. The chapter is concluded in Section 5.7.

5.2 System Model

We consider a dynamic routing system consisting of N routers and a logically centralized controller with traffic engineering capabilities. We model the network infrastructure as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ represent the set of routers and the set of directed links, respectively. Routers employ OSPF routing with equal cost multi-path (ECMP) to forward the traffic through them. Each link e_{ij}, from router i to j, has a limited capacity c_e and is assigned a link weight α_e.

A unicast traffic demand $w_{sd} \in \mathcal{W}$, where $\mathcal{W}$ represents the set of active traffic demands, originates from a source s and has to reach a destination d by flowing through a routing path in the network. Under OSPF routing, all routers maintain the same global view of the link weights, and hence can compute the same routing path for any traffic demand w_{sd}. For cost-efficient routing without cycles, it is sufficient for a router n to compute and employ a router-specific forwarding table, which only enlists the next router on the shortest path of a traffic demand w_{sd} to any destination

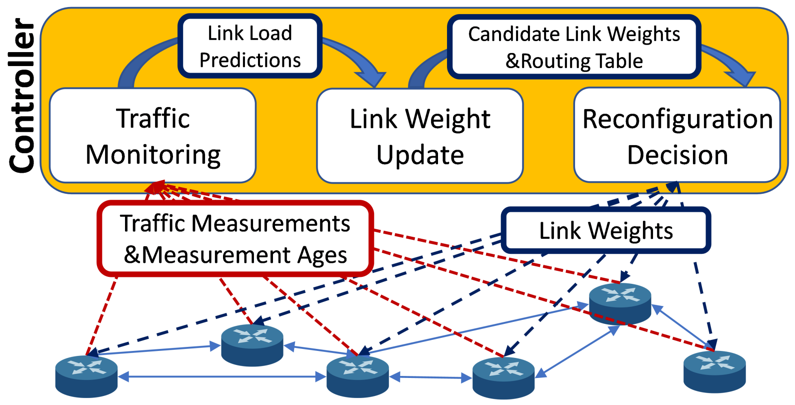


Figure 5.1: Illustration of the Controller-aided Link Weight Update.

$d \in \mathcal{N}$. Forwarding tables have to be recomputed whenever a link weight changes its value.

Clearly, one particular link can be traversed by multiple routing paths. Consequently, a link load x_e is evaluated as a weighted sum of all traffic demands, whose routing paths pass through the link e, normalized by link capacities:

$$x_e = \sum_{s=1}^{N} \sum_{d=1, d\neq s}^{N} \frac{g_{e,sd} w_{sd}}{c_e}, \tag{5.1}$$

where $g_{e,sd}$ is the routing coefficient that quantifies what ratio of traffic demand w_{sd} traverse through link e. Obviously, the link e is said to be congested if $x_e > 1$. In a congested link, some packets have to be dropped or put into queue, and hence congested links degrade network QoS. Routing coefficients are determined by the shortest path nature of OSPF routing, in a similar fashion forwarding tables are computed. Routing coefficients for every link and traffic demand pair can be stored in matrix form as a routing table. To this end, routing and forwarding tables at each router embodies the same routing paths in the network, and hence can interchangeably be used.

In interest of sustaining adequate levels of network utilization for longer periods of time, the centralized controller must employ fast and responsive traffic engineering techniques for adapting to dynamic traffic (Figure 5.1). For this purpose, adaptive traffic engineering operates in an iterative manner with each iteration lasting for Δt seconds. Every router measures the traffic demand originating from itself and sends these measurements to the controller at unique time instants. Subsequently, the most recent link load values can be computed using equation (5.1), traffic demand measurements and a routing table. To this end, the controller retains the *deployed routing table* corresponding to the currently *deployed link weights* in the network. A fast load-sensitive weight search algorithm can swiftly compute a new set of OSPF weights using the link load values. Clearly, inconsistent and obsolete measurements may mislead the controller when computing new *candidate link weights*, and hence newly computed candidate link weights, not necessarily, mitigate network congestion, while deploying them causes temporary routing instability.

In light of these observations, at the start of each iteration, the controller first predicts the link loads for the next iteration. Secondly, candidate link weights are computed by using the predicted link loads for better network utilization. At the final stage, the controller executes a myopic policy, in which it compares the costs implied by the candidate and the currently deployed OSPF weights to decide whether to reconfigure the network or not. To this end, network utilization cost accounts for network congestion, link load uncertainty due to age of traffic measurements and dynamic traffic, and routing instability during reconfiguration. The controller then broadcasts the candidate link weights to the network if they mitigate the network utilization cost. Routers then compute their unique forwarding tables using new link weights and route the traffic accordingly in the next iteration.

5.3 Traffic Model and Measurement Age

In the previous section, we demonstrated how dynamic routing operates in a network and how traffic engineering tasks could be handled at a centralized controller. Next, traffic demand monitoring is investigated in detail considering the implications of the age of measurements.

5.3.1 Traffic Demand Model

Conventional traffic engineering techniques are based on the strict assumption of quasi-stationary traffic. To this end, traffic demands are often modeled as a random process, whose distributional properties, e.g., mean and variance, vary on a longer time-scale. Nevertheless, the quasi-stationary assumption fails to reflect the rapidly changing contemporary traffic environment constituted by mobile devices and bandwidth-hungry applications. In this chapter, we do not restrict traffic to be quasi-stationary and we adopt a comprehensive yet simple stochastic model.

Let $w_{sd}(t)$ and $w_{sd}^k = w_{sd}(k\Delta t)$ represent the traffic demand from s to d as a function of time and the kth sample of traffic demand at time instant $t = k\Delta t$, respectively. Markovian approaches are broadly adopted in the literature to model recursive time-varying quantities [100, 43]. In particular, autoregressive models can mathematically represent random fluctuations on a parameter during each recursion as a stochastic additive term on the previous value of the parameter. In consideration of analytical simplicity, we cast traffic evolution over iterations as a first order autoregressive model:

$$w_{sd}^k = w_{sd}^{k-1} + \varepsilon_{sd}^k, \qquad \forall s, d \in \mathcal{N}, \tag{5.2}$$

where ε_{sd}^k is the random fluctuation on the traffic demand from s to d due to new and terminated sessions during the kth iteration. Note that the kth iteration here refers to the time interval $((k-1)\Delta t, k\Delta t)$. We do not assume any particular distribution for ε_{sd}^k and further allow its mean and standard deviation, μ_{sd}^k and σ_{sd}^k respectively, to change over recursions.

5.3.2 Age of Traffic Measurements

In consideration of synchronization overhead, routers asynchronously send their measurements, and hence the measurements arrive at the controller at separate time instants. Let τ_s^ℓ and t_s^ℓ represent the measurement instant (or equivalently transmission instant) of the ℓth traffic measurements of a router s and the arrival time of the corresponding measurements at controller, respectively. τ_s^ℓ and t_s^ℓ are separated by a random delivery duration, denoted by γ_s^ℓ. Further, considering communication overhead, routers may choose to wait some time before sending new measurements to the controller. On the other hand, rare measurement updates constitute inaccurate global view of the network at controller, and hence damage network utilization. For this cause, each router s employs a waiting time policy β_s^ℓ to adjust the trade-off between accuracy and communication overhead. Computing optimal waiting time policy is a stochastic control problem and has been studied in depth in [126, 74]. This problem is out of this chapter's scope, and hence we employ an adaptive waiting time policy, $\beta_s^\ell = \gamma_s^\ell$, i.e., s takes and sends a new measurement as soon as the latest measurement arrives at the controller:

$$\tau_s^\ell = \tau_s^{\ell-1} + \gamma_s^{\ell-1} = t_s^{\ell-1}, \qquad \forall s \in \mathcal{N}. \tag{5.3}$$

We here observe that some routers may deliver multiple traffic measurements during one iteration. Since controller operates at discrete time instants, it only utilizes the most recent measurements from each router. Traffic measurement age is then expressed as given in Definition 6. We here remind that the concept of *Age of Infor-*

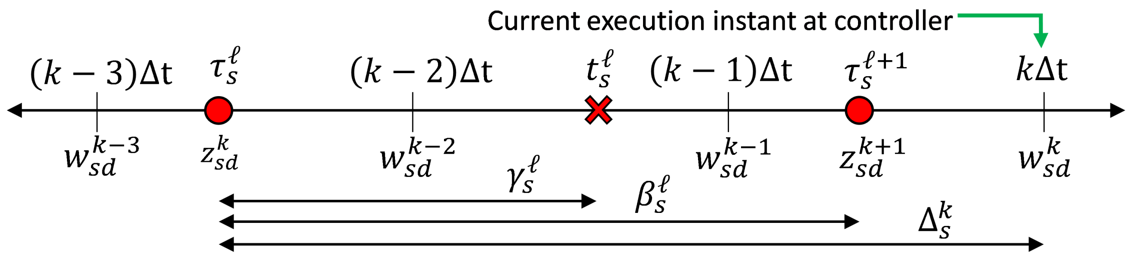


Figure 5.2: Time-line for Sampling Traffic Demand and Collecting Measurements.

mation is widely adopted in control theory literature [126, 74, 162].

Definition 6. *Age of the most recently received traffic measurements from a router s before the end of iteration k is defined as follows.*

$$\Delta_s^k = k\Delta t - \max_\ell(\tau_s^\ell : t_s^\ell \le k\Delta t), \qquad \forall s \in \mathcal{N}. \tag{5.4}$$

Clearly, the measurements do not perfectly reflect the current traffic conditions in the network and possess higher uncertainty regarding the traffic conditions as measurement ages surge. To this end, the controller has to exploit statistical mean and variance, μ_{sd}^k and $(\sigma_{sd}^k)^2$ respectively, and measurement age, Δ_s^k, to accurately estimate contemporary traffic conditions in the network. Recalling that the controller determines new traffic engineering rules for the $k+1$st iteration at the time instant $t = k\Delta t$, the next section describes how to predict traffic demands for the time instant $t = (k+1)\Delta t$ using the autoregressive model and the age of traffic measurements.

5.3.3 Traffic Demand Prediction at the Controller

Each delivered measurement message contains information regarding traffic demand and a time stamp pointing the time instant that the measurement was taken (Figure 5.2). Controller can compute the age of measurements by substituting the time stamp in (5.3) and (5.4). Let z_{sd}^k represent the measurement of traffic demand $w_{sd}(t)$ sampled at time instant $t = k\Delta t - \Delta_s^k$. If the measurements were fresh, i.e.,

$\Delta_s^k = 0$, the measurements then would exactly indicate the current traffic demand, i.e., $w_{sd}^k = z_{sd}^k$. On the other hand, non-zero age values impose uncertainty on traffic measurements since actual traffic demands have changed during this time period. In particular, this uncertainty increases along with the age of measurements and the rates of traffic demand fluctuations. Here, a specific traffic demand fluctuation from s to d is a continuous white random process, denoted by $\varepsilon_{sd}(t)$, and its relation to ε_{sd}^k is defined next.

Definition 7. *Over a single iteration, for any source destination pair, the change in discrete traffic demand is equivalent to overall change in continuous traffic demand.*

$$\varepsilon_{sd}^k = \int_{(k-1)\Delta t}^{k\Delta t} \varepsilon_{sd}(t)dt, \qquad \forall s, d \in \mathcal{N}. \tag{5.5}$$

In light of above definition, the uncertainty due to age of measurements can be quantified in terms of $\varepsilon_{sd}(t)$ by changing the lower limit of the integral.

$$w_{sd}^k - z_{sd}^k = \int_{k\Delta t - \Delta_s^k}^{k\Delta t} \varepsilon_{sd}(t)dt, \qquad \forall s, d \in \mathcal{N}. \tag{5.6}$$

A careful investigation reveals that (5.6) and the autoregressive model defined in (5.2) are equivalent:

$$w_{sd}^k = z_{sd}^k + \int_{k\Delta t - \Delta_s^k}^{(k-1)\Delta t} \varepsilon_{sd}(t)dt + \int_{(k-1)\Delta t}^{k\Delta t} \varepsilon_{sd}(t)dt = w_{sd}^{k-1} + \varepsilon_{sd}^k. \tag{5.7}$$

In general, distributional parameters of $\varepsilon_{sd}(t)$ evolve real time since $\varepsilon_{sd}(t)$ is non-stationary. However, estimation of distributional parameters of a continuous random process $\varepsilon_{sd}(t)$ requires extraordinary computational resources and traffic history even if modern learning techniques are employed. A simpler, but more effective approach is to assume $\varepsilon_{sd}(t)$ to be stationary for periods of time, e.g., during each iteration. Under this approach, distribution of $\varepsilon_{sd}(t)$ can only change its distributional properties at transition instants from one iteration to another, but can still satisfy the non-stationary traffic conditions captured in the autoregressive model (5.2). Partially stationary assumption along with (5.5) constitute a simpler expression for (5.6):

$$w_{sd}^k - z_{sd}^k = \left(\frac{\Delta_s^k}{\Delta t} - \delta_s^k\right) \varepsilon_{sd}^{k-\delta_s^k} + \sum_{i=0}^{\delta_s^k - 1} \varepsilon_{sd}^{k-i}, \qquad \forall s, d \in \mathcal{N}, \tag{5.8}$$

where $\delta_s^k = \lfloor \Delta_s^k / \Delta t \rfloor$ that is the largest integer value smaller than $\Delta_s^k/\Delta t$. We here observe that the most recent measurements are not necessarily sampled in the current iteration. Further we note that the second summation term in (5.8) vanishes if the most recent measurements are sampled during the current iteration, i.e., $\delta_s^k = 0$.

As described in above sections, controller first aims at predicting the traffic demand that will occur at the start of next iteration, i.e., $t = (k+1)\Delta t$. To this end, we employ a single stage predictor and a mean-squared estimator that utilize the autoregressive model (5.2), traffic measurements and their corresponding ages. We know from control theory that the estimator, which minimizes mean-squared error, is the conditional expectation of the variable of interest with respect to measurements.

Proposition 5. *Let $\hat{w}_{sd}^k$ denote the mean-squared estimator of traffic demand w_{sd}^k. Then, $\hat{w}_{sd}^k$ can be determined using* (5.2), *traffic measurements and corresponding ages:*

$$\hat{w}_{sd}^k = \mathbb{E}[w_{sd}^k | z_{sd}^k, \Delta_s^k] = \begin{cases} \hat{w}_{sd}^{k-1} + \mu_{sd}^k, & \text{if } \Delta_s^k \geq \Delta t, & (5.9\text{a}) \\ z_{sd}^k + \dfrac{\Delta_{sd}^k}{\Delta t} \mu_{sd}^k, & \text{if } \Delta_s^k < \Delta t. & (5.9\text{b}) \end{cases}$$

Proof. We utilize (5.7), the equivalent form of (5.2), and (5.8) to prove Proposition 5, and compute conditional expectations of w_{sd}^k for given z_{sd}^k and Δ_s^k.

$$\begin{aligned} \hat{w}_{sd}^k =& \mathbb{E}\left[z_{sd}^k + \left(\frac{\Delta_s^k}{\Delta t} - \delta_s^k\right) \varepsilon_{sd}^{k-\delta_s^k} + \sum_{i=0}^{\delta_s^k - 1} \varepsilon_{sd}^{k-i} \middle| z_{sd}^k, \Delta_s^k \right] \\ =& z_{sd}^k + \left(\frac{\Delta_s^k}{\Delta t} - \delta_s^k\right) \mu_{sd}^{k-\delta_s^k} + \sum_{i=0}^{\delta_s^k - 1} \mu_{sd}^{k-i}. \end{aligned} \tag{5.10}$$

Next, we evaluate two cases; 1) $\Delta_s^k \geq \Delta t$ and 2) $\Delta_s^k < \Delta t$. In the former case, $\delta_s^k = \left\lfloor {}^{\Delta_s^{k-1}}/_{\Delta t} + {}^{\Delta t}/_{\Delta t} \right\rfloor = \left\lfloor {}^{\Delta_s^{k-1}}/_{\Delta t} \right\rfloor + 1 = \delta_s^{k-1} + 1$, and hence (5.10) is further computed as:

$$\hat{w}_{sd}^k = z_{sd}^k + \left(\frac{\Delta_s^{k-1}}{\Delta t} - \delta_s^{k-1} \right) \mu_{sd}^{k-1-\delta_s^{k-1}} + \sum_{j=i-1=0}^{\delta_s^{k-1}-1} \mu_{sd}^{k-1-j} + \mu_{sd}^k = \hat{w}_{sd}^{k-1} + \mu_{sd}^k. \quad (5.11)$$

For the latter case, we have $\delta_{sd}^k = 0$, and hence (5.9b) is directly obtained by substituting $\delta_{sd}^k = 0$ in (5.10). □

In a similar way, single stage predictor predicts the traffic demand that will occur at the start of next iteration:

$$\hat{w}_{sd}^{k+1} = \mathbb{E}[w_{sd}^{k+1} | z_{sd}^k, \Delta_s^k] = \hat{w}_{sd}^k + \mu_{sd}^{k+1}. \quad (5.12)$$

Derivation of (5.12) follows the same steps we used in the proof of (5.9a).

5.4 Load Sensitive Link Weight Updates

In this chapter, we more focus on mitigating network congestion over time in quick iterations, and hence improving time-averaged network utilization in a dynamically changing traffic environment, rather than consuming huge amounts of time to minimize congestion in just one iteration. In spite of this, we employ a fast and responsive load-sensitive link weight update algorithm. In contrast to local search algorithms, which recursively search for the best link weight settings, load-sensitive routing can compute new link weights in a single iteration, and hence can adapt to rapidly changing traffic conditions. However, computed link weights are often far from being optimal solutions to well-known multi-commodity flow problem, which is shown to be NP-hard in terms of link weights. Despite this, load-sensitive link weight updates can outperform local search algorithms in improving time-averaged network utilization due to their adaptability to ever-changing traffic conditions.

5.4.1 Link Load Prediction

Load-sensitive weight update techniques rely on link load measurements to compute new link weights. For this cause, success of new link weights hinges on the accuracy of traffic demand measurements. Gathering precise traffic measurements is often a challenge for controllers, because the measurements of distinct routers often arrive at controllers with distinct ages. To overcome this problem, as described in above sections, we first collect traffic measurements and subsequently predict one-step ahead traffic conditions exploiting an auto-regressive traffic model. Predicted traffic demands can then be utilized to predict link loads by exploiting the routing matrix constituted by OSPF link weights.

$$\hat{x}_{e,\kappa}^{k+1} = \sum_{s=1}^{N} \sum_{d=1, d\neq s}^{N} \frac{g_{e,sd}^{\kappa} \hat{w}_{sd}^{k+1}}{c_e}, \qquad \forall e \in \mathcal{L}, \tag{5.13}$$

where $g_{e,sd}^{\kappa}$, for all $e \in \mathcal{L}$ and for all $s, d \in \mathcal{N}$, denote the routing coefficients constituted by the link weight set with index κ and $\hat{x}_{e,\kappa}^{k+1}$, for all $e \in \mathcal{L}$, represent the corresponding predicted link loads. We here observe that separate routing tables lead to distinct set of link loads, and hence a good choice of link weights can mitigate congestion.

5.4.2 Load-Sensitive Update of Link Weights

Let $g_{e,sd}^{0}$, for all $e \in \mathcal{L}$ and for all $s, d \in \mathcal{N}$, represent the routing coefficients established by the currently deployed OSPF link weights, α_e^0 for all $e \in \mathcal{L}$, in the network. A load-sensitive link weight update algorithm aims for obtaining new link weights, α_e^{κ} for all $e \in \mathcal{L}$, that potentially mitigate maximum link load under tight timing constraints. In this regard, controller must increase the costs of traversing congested links via increasing their link weights since traffic demands are routed through their min-cost, i.e., shortest, paths in the network. With this approach, total number of shortest paths passing through congested links can be reduced to mitigate corresponding link loads.

Load-sensitive routing algorithms often exploit a mapping from link loads to link weights that penalizes high link loads by increasing their link weights to control network congestion. In this chapter, we employ a simple, yet fast load-sensitive routing algorithm, LSAR [138]. LSAR adopts a piecewise linear mapping that is demonstrated to constitute routing stability under stationary traffic conditions [138].

$$f(y(\hat{x}_{e,\kappa}^{k+1})) = \begin{cases} 1, & \text{if } y(\hat{x}_{e,\kappa}^{k+1}) \leq 0.5, \\ 4y(\hat{x}_{e,\kappa}^{k+1}) - 1, & \text{if } y(\hat{x}_{e,\kappa}^{k+1}) > 0.5, \end{cases} \qquad \forall e \in \mathcal{L}, \tag{5.14}$$

where $y(\hat{x}_{e,\kappa}^{k+1})$ is a generic function of predicted link loads. At the moment, we employ a linear function $y(\hat{x}_{e,\kappa}^{k+1}) = \hat{x}_{e,\kappa}^{k+1}$ as a straightforward choice. Yet, a novel uncertainty-aware function is presented in the subsequent sections. In an effort to improve routing stability further, the average of new link weight and previous η link weights is computed. In the final stage, this average value is assigned as a candidate link weight if the difference between this value and the most recent link weight is larger than a threshold ξ.

$$\alpha_e^{\kappa+1} = \begin{cases} \theta = \frac{f(\hat{x}_{e,\kappa}^{k+1})}{\eta+1} + \sum_{i=0}^{\eta-1} \frac{\alpha_e^{\kappa-i}}{\eta+1}, & \text{if } |\alpha_e^{\kappa} - \theta| > \xi, \\ \alpha_e^{\kappa}, & \text{if } |\alpha_e^{\kappa} - \theta| \leq \xi. \end{cases} \tag{5.15}$$

$\alpha_e^{\kappa+1}$, $\forall e \in \mathcal{L}$, can then be used to compute candidate routing coefficients $g_{e,sd}^{\kappa+1}$ via executing a shortest path algorithm, e.g., Dijkstra's algorithm.

5.5 Adaptive Link Weight Reconfiguration

5.5.1 Uncertainty-Aware Weight Reconfigurations

The ultimate goal in this chapter is improving time-averaged network utilization and routing robustness under dynamic traffic conditions. Maximum link load, i.e., network congestion, is widely accepted as a proper measure of network utilization in the literature [52, 146, 75, 96]. Under dynamic traffic conditions, network congestion can be expressed in terms of predicted link loads, for which a single stage predictor is formulated in above sections. Despite being widely adopted, network congestion does not value the impact of uncertainty due to the age of measurements, and hence degrades the efficiency of traffic engineering techniques when traffic measurements possess non-zero age values.

In light of above concern, we further incorporate uncertainty on top of link loads by employing squared link loads as a measure for link weight updates and reconfiguration decisions. Besides, quadratic cost functions are often employed in control of recursive decision-making processes, i.e., linear quadratic regulators [21]. Next, a single stage predictor for squared link loads is formulated.

Proposition 6. *Let $P_{ee,\kappa}^{k+1}$ represent the variance of eth conditional link load computed for iteration $k+1$ under κth link weight set for given z_{sd}^{k} and Δ_s^k. Single stage predictor of $(x_{e,\kappa}^{k+1})^2$ then can be calculated in terms of single stage predictor of $x_{e,\kappa}^{k+1}$ and $P_{ee,\kappa}^{k+1}$:*

$$\widehat{(x_{e,\kappa}^{k+1})^2} = \mathbb{E}[(x_{e,\kappa}^{k+1})^2 | \mathbf{z}^k, \mathbf{\Delta}^k] = (\hat{x}_{e,\kappa}^{k+1})^2 + P_{ee,\kappa}^{k+1}, \tag{5.16}$$

where $\mathbf{z}^k$ and $\mathbf{\Delta}^k$ denote the vectors of traffic measurements and corresponding ages, respectively.

Proof. Proof is straightforward and follows:

$$\begin{aligned}\widehat{(x_{e,\kappa}^{k+1})^2} &= \mathbb{E}[(x_{e,\kappa}^{k+1} - \mathbb{E}[x_{e,\kappa}^{k+1}|\mathbf{z}^k, \mathbf{\Delta}^k])^2 | \mathbf{z}^k, \mathbf{\Delta}^k] \\ &\quad + \mathbb{E}^2[x_{e,\kappa}^{k+1}|\mathbf{z}^k, \mathbf{\Delta}^k] = P_{ee,\kappa}^{k+1} + (\hat{x}_{e,\kappa}^{k+1})^2,\end{aligned} \tag{5.17}$$

where the expression of $P_{ee,\kappa}^{k+1}$ is given in Lemma 4. □

Lemma 4. *Let Q_{sd}^{k+1} represent the conditional variance of traffic demand from s to d computed for iteration $k+1$ for given z_{sd}^k and Δ_s^k. $P_{ee,\kappa}^{k+1}$ is given in terms of κth routing coefficients and Q_{sd}^{k+1}:*

$$P_{ee,\kappa}^{k+1} = \sum_{s=1}^{N} \sum_{d=1,d\neq s}^{N} \left(\frac{g_{e,sd}^{\kappa}}{c_e}\right)^2 Q_{sd}^{k+1}. \tag{5.18}$$

Proof. Proof follows from the independence of distinct traffic demands:

$$\begin{aligned} P_{ee,\kappa}^{k+1} &= \mathrm{Var}[x_{e,\kappa}^{k+1}|\mathbf{z}^k, \mathbf{\Delta}^k] = \mathrm{Var}\left[\sum_{s=1}^{N} \sum_{\substack{d=1 \\ d\neq s}}^{N} \frac{g_{e,sd}^{\kappa}}{c_e} w_{sd}^{k+1} \middle| \mathbf{z}^k, \mathbf{\Delta}^k\right] \\ &= \sum_{s=1}^{N} \sum_{\substack{d=1 \\ d\neq s}}^{N} \left(\frac{g_{e,sd}^{\kappa}}{c_e}\right)^2 \mathrm{Var}\left[w_{sd}^{k+1}|z_{sd}^k, \Delta_s^k\right] = \sum_{n=1}^{N} \sum_{\substack{d=1 \\ d\neq n}}^{N} \left(\frac{g_{e,sd}^{\kappa}}{c_e}\right)^2 Q_{sd}^{k+1}. \end{aligned} \tag{5.19}$$

where Q_{sd}^k is computed in Lemma 5. □

Lemma 5. *Q_{sd}^k can be computed in terms of discrete traffic demand variances and measurement ages using* (5.2)*.*

$$Q_{sd}^k = \mathrm{Var}[w_{sd}^k|z_{sd}^k, \Delta_s^k] = \begin{cases} Q_{sd}^{k-1} + (\sigma_{sd}^k)^2, & \text{if } \Delta_s^k \geq \Delta t, & (5.20a) \\ \dfrac{\Delta_s^k}{\Delta t}(\sigma_{sd}^k)^2, & \text{if } \Delta_s^k < \Delta t. & (5.20b) \end{cases}$$

Proof. The proof follows computing conditional variance by utilizing (5.6) and independence of distinct traffic demands:

$$\begin{aligned} Q_{sd}^k =& \mathbb{E}\left[\left(z_{sd}^k + \int_{k\Delta t - \Delta_s^k}^{k\Delta t} \varepsilon_{sd}(t)dt - \mathbb{E}\left[z_{sd}^k + \int_{k\Delta t - \Delta_s^k}^{k\Delta t} \varepsilon_{sd}(t)dt\right]\right)^2\right] \\ =& \mathbb{E}\left[\left(\int_{k\Delta t - \Delta_s^k}^{k\Delta t} (\varepsilon_{sd}(t) - \mu_{sd}(t))dt\right)^2\right] \\ =& \mathbb{E}\left[\int_{k\Delta t - \Delta_s^k}^{k\Delta t} \int_{k\Delta t - \Delta_s^k}^{k\Delta t} (\varepsilon_{sd}(t_1) - \mu_{sd}(t_1))(\varepsilon_{sd}(t_2) - \mu_{sd}(t_2))dt_1 dt_2\right] \\ =& \mathbb{E}\left[\int_{k\Delta t - \Delta_s^k}^{k\Delta t} \int_{k\Delta t - \Delta_s^k}^{k\Delta t} K_{\varepsilon_{sd}\varepsilon_{sd}}(t_1, t_2)dt_1 dt_2\right] \end{aligned} \tag{5.21}$$

Following the steps in (5.21), Q_{sd}^{k} is represented as a double integral of auto-covariance matrix of $\varepsilon_{sd}(t)$. Since distinct traffic demands are independent of each other, auto-covariance matrix is diagonal, and hence above expression is simplifies to (5.20a) as:

$$\begin{aligned} Q_{sd}^{k} = \int_{k\Delta t-\Delta_s^k}^{k\Delta t} \mathrm{Var}[\varepsilon_{sd}(t)]dt = \int_{(k-1)\Delta t-\Delta_s^{k-1}}^{(k-1)\Delta t} \mathrm{Var}[\varepsilon_{sd}(t)]dt + \int_{(k-1)\Delta t}^{k\Delta t} \mathrm{Var}[\varepsilon_{sd}(t)]dt \\ = Q_{sd}^{k-1} + (\sigma_{sd}^{k})^2, \end{aligned} \tag{5.22}$$

when $\Delta_s^k \geq \Delta t$. If $\Delta_s^k < \Delta t$, (5.21) simplifies to (5.20b) as follows:

$$Q_{sd}^{k} = \int_{k\Delta t-\Delta_s^k}^{k\Delta t} \mathrm{Var}[\varepsilon_{sd}(t)]dt = \frac{\Delta_s^k}{\Delta t}(\sigma_{sd}^{k})^2, \tag{5.23}$$

which completes the proof. □

In a similar way to (5.20a), Q_{sd}^{k+1} is computed from Q_{sd}^{k}:

$$Q_{sd}^{k+1} = Q_{sd}^{k} + (\sigma_{sd}^{k+1})^2. \tag{5.24}$$

The above proposed measure can leverage the uncertainty information pertaining to the age of measurements. In this regard, a function of the second conditional moment of link loads can be fed into LSAR to compute new set of link weights with intent to mitigate link load uncertainty alongside link load values. Since LSAR is designed for the linear functions of link loads, we select this function as the square-root function for better adaptation to LSAR.

$$y(\hat{x}_{e,\kappa}^{k+1}) = \sqrt{\widehat{(x_{e,\kappa}^{k+1})^2}} = \sqrt{P_{ee,\kappa}^{k+1} + (\hat{x}_{e,\kappa}^{k+1})^2}. \tag{5.25}$$

We here observe that $y(\hat{x}_{e,\kappa}^{k+1})$ reduces to $\hat{x}_{e,\kappa}^{k+1}$ if measurements are instantly and synchronously collected. In the subsequent section, we finally present the measurement age-aware reconfiguration decision-making problem.

5.5.2 Adaptive Link Weight Reconfiguration Policy

Frequent reconfigurations cause instability in routing and impair QoS. On contrary, occasional reconfigurations conflicts with the interest of mitigating network congestion. Accordingly, the network reconfiguration problem, with each reconfiguration admitting addition routing cost, can be cast as a stochastic optimization problem with link load costs and reconfiguration constraints. In [100, 43], a dynamic programming solution to this problem is illustrated for the SDN architecture, in which the traffic engineering rules that are computed at each iteration strictly improves network utilization. Unfortunately, this approach can not be adopted under OSPF routing, in particular, due to two implications of the shortest path nature of link state routing: 1) in contrast to SDN, the weight update algorithms operating on OSPF routing can not always find better traffic engineering rules, and 2) future link weights and the corresponding future link loads can not be associated through a closed form cost function, and hence it is not possible to formulate a dynamic programming problem [21].

In light of complexity, we restrict our attention to myopic policies, in which a controller has to make a reconfiguration decision only considering one-step ahead predicted network utilizations costs. To this end, the controller decides to reconfigure the network with new candidate link weights if they not only mitigate network congestion and uncertainty but also recover the reconfiguration cost over iterations. The controller recursively solves the following myopic problem:

$$J^*(\mathbf{x}_\kappa^{k+1}, u^k) = \min_{u^k \in \{0,1\}} (\epsilon^\rho u^k + \max_{e \in \mathcal{L}} y^2(\hat{x}_{e,\kappa \cdot u^k}^{k+1})), \tag{5.26}$$

where $\mathbf{x}_\kappa^{k+1}$ is the vector of all predicted link loads under the κth link weight set. In (5.26), u^k is the control variable that takes value 1 if controller reconfigures the network with the most recent candidate weights, α_e^κ, $\forall e \in \mathcal{L}$, or takes value 0 oth-

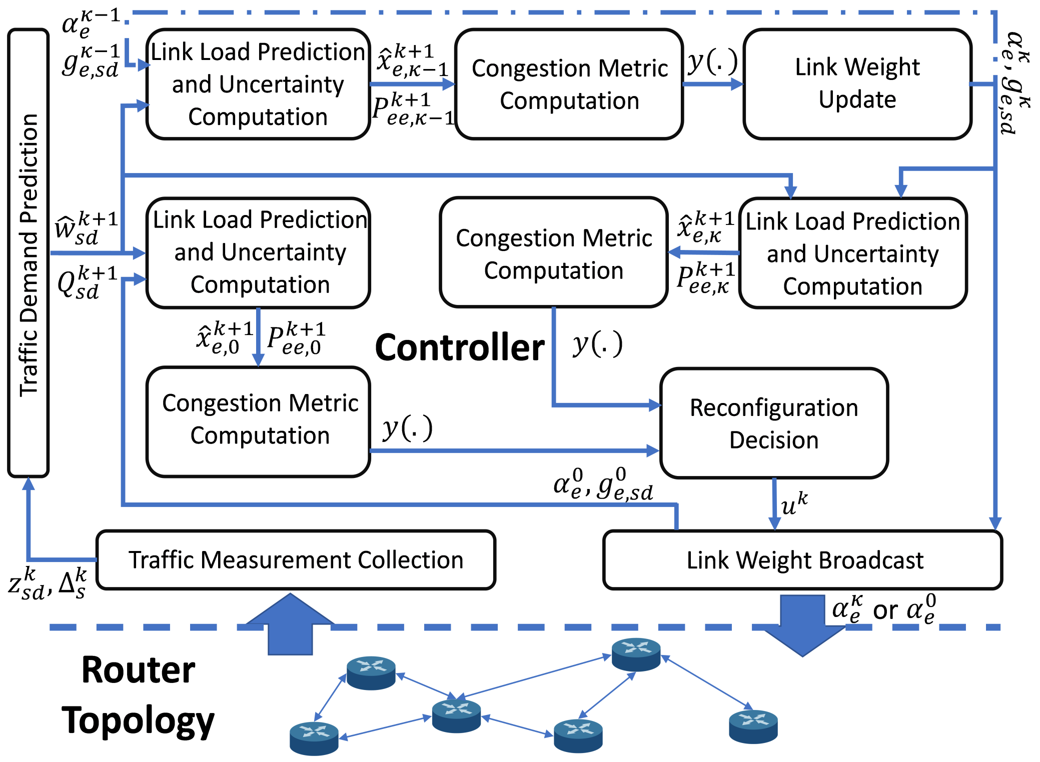

Traffic Demand Prediction
$\alpha_e^{\kappa-1}$
$g_{e,sd}^{\kappa-1}$
Link Load Prediction and Uncertainty Computation
$\hat{x}_{e,\kappa-1}^{k+1}$
$P_{ee,\kappa-1}^{k+1}$
Congestion Metric Computation
$y(.)$
Link Weight Update
$\alpha_e^{\kappa}, g_{e,sd}^{\kappa}$
$\hat{w}_{sd}^{k+1}$
Q_{sd}^{k+1}
Link Load Prediction and Uncertainty Computation
Congestion Metric Computation
$\hat{x}_{e,\kappa}^{k+1}$
$P_{ee,\kappa}^{k+1}$
Link Load Prediction and Uncertainty Computation
$\hat{x}_{e,0}^{k+1}$ $P_{ee,0}^{k+1}$
Controller
$y(.)$
Congestion Metric Computation
$y(.)$
Reconfiguration Decision
$\alpha_e^0, g_{e,sd}^0$
u^k
Traffic Measurement Collection
Link Weight Broadcast
z_{sd}^k, Δ_s^k
α_e^{κ} or α_e^0
Router Topology

Figure 5.3: A Sketch of Link Weight Updates and Routing Reconfiguration While Considering Traffic Measurement Ages.

erwise. ϵ^ρ represents the reconfiguration cost and it is monotonically decreasing in ρ, the number of iterations since the last reconfiguration. Hence, the reconfiguration cost reduces at each recursion until another reconfiguration occurs. Bearing in mind that the reconfiguration cost differs at distinct networks, we model ϵ^ρ as a proportion of network congestion cost that is constituted by the deployed set of link weights.

$$\epsilon^\rho = (\Phi/\rho)\max_{e\in\mathcal{L}} y^2(\hat{x}_{e,0}^{k+1}), \tag{5.27}$$

where Φ is a constant that depends on network structure. Next section demonstrates the simulation results illustrating the impact of measurement ages and dynamic traffic conditions. Further, the performances of the two single stage predictors proposed above sections are compared.

5.6 Simulation Results and Discussion

5.6.1 Simulation Setting

Since we are interested in mitigating time average network congestion under dynamic traffic conditions, the controller operates as follows. It first computes link load predictions and variances for collected measurements, while taking the ages of measurements into account. Subsequently, it computes candidate link weights using LSAR and link load predictions. In the final stage, the controller computes the link load, uncertainty and reconfiguration costs for deployed and candidate link weights, and reconfigures the network if profitable. The operation of the controller is demonstrated in Figure 5.3.

In our experiments, we use ARPANET, which consists of 48 nodes and 140 links, and randomly generated graphs as underlying network structure. Random graphs are generated using Waxman's method with Poisson process intensity, maximal link probability, and edge length parameters are chosen as 0.12, 0.7, and 0.7, respectively. For traffic generation, we adopt the technique used in [100] that implements the most resembling traffic conditions to our assumptions. Accordingly, we generate random traffic demands according to a Poisson process with distinct inter-arrival times at different iterations. Each unicast traffic request lasts for a fixed duration, ψ. Since we allow traffic demands to be non-stationary, we take samples from a uniform distribution to specify the inter-arrival times of Poisson process for each iteration. We model the random measurement delivery duration with a stationary exponential distribution and further assume that the delivery durations for distinct nodes are identically distributed and independent. Table 5.1 shows all the parameters used in all the experiments.

Table 5.1: The Summary of All Parameters Used in All Experiments.

Exp.	Traffic Type	Inter-arrival Time	Simulation Duration	ψ	$\mathbb{E}[\gamma_s^\ell]$
B	Constant	N/A	1 hour	N/A	Vary
C	Stationary	10ms	30 mins	1s	Vary
D	Non-stationary	Unif(1s, 3s)	1 hour	10s	1ms
E	Non-stationary	Unif(2s, 4s)	1 hour	17.5s	1ms
F	Non-stationary	Unif(3s, 5s)	1 hour	25s	1ms

Exp.	c_e	Init. Link Weights	Δt	Unicast Traffic Amt.	ξ	η	Φ
All	1 GB/s	1	30s	1 Mb/s	0.1	3	0.2

5.6.2 Congestion Mitigation with Oblivious Traffic Information

In this experiment, the 48 nodes in ARPANET are split into two groups, G_1 and G_2. For the time intervals $t_1 = [0\text{s}, 900\text{s}]$ and $t_3 = [1800\text{s}, 2700\text{s}]$, only the nodes in G_1, and for the time intervals $t_2 = [900\text{s}, 1800\text{s}]$ and $t_4 = [2700\text{s}, 3600\text{s}]$, only the nodes in G_2 transmit a constant traffic of 5 Mb/s. Because of limited space, we only present the results obtained using the linear link load metric.

Figure 5.4 demonstrates maximum link load evolution over iterations for 2 different scenarios. As illustrated, the proposed system mitigates maximum link load in quick iterations when the measurements are fresh. On contrary, the out-of-date measurements either slow down the controller's traffic engineering operation or even mislead the controller to implement worse routing rules at certain iterations. Yet, the congestion is better mitigated in the latter scenario at some iterations due to the nonlinear nature of OSPF routing.

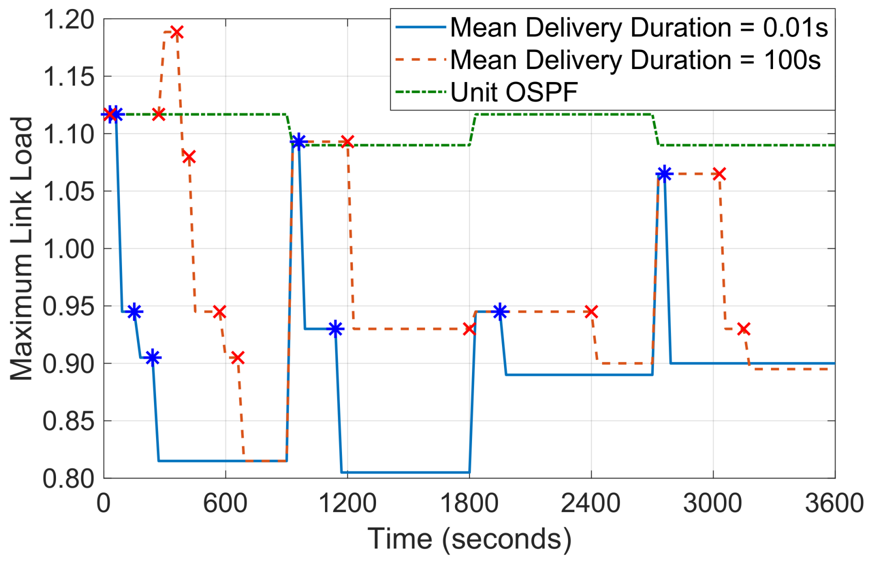


Figure 5.4: The Evolution of Maximum Link Load with Oblivious Traffic Information. $*$ and $+$ Represent the Time Instances for Reconfiguration. Under Unit OSPF Settings, All Link Weights Are Assigned to Unit Value of 1 at All Iterations. (B)

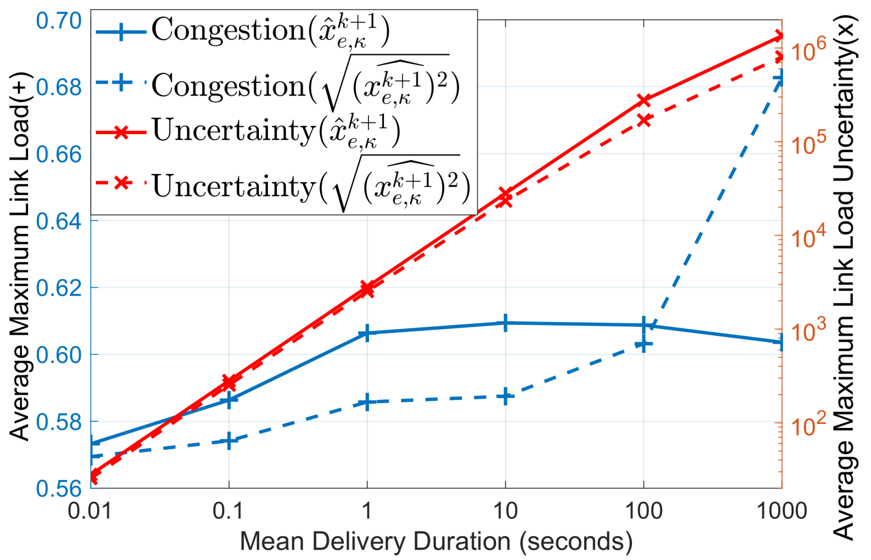


Figure 5.5: The Average Maximum Link Load (Left Axis) and Average Maximum Link Load Uncertainty (Right Axis) Comparisons of Linear and Squared Link Load Metrics over Various Delivery Duration Scenarios. Note that the Increments on the Right Vertical Axis are Logarithmic. (C)

5.6.3 Impact of Measurement Age

In this experiment, we compare the average performances of the predictors derived in above sections under different delivery duration scenarios. For this cause, we randomly generate 10, at least 2-connected, networks consisting of 10 nodes using Waxman's technique. For each network, we sample 50 traffic realizations and run 50 simulations. At each simulation, we only use the measurements from the final iteration for evaluation purposes. In consideration of the nonlinearity of OSPF routing, we here solve the linear program (LP) formulation of routing problem in terms of link flows, and hence obtain the optimal routing for both predictors to ensure a fair comparison. To this end, we use the formulation presented in [146], which is generalized as:

$$\begin{aligned}
\min_{g^{\kappa}_{e,sd}, \forall e,s,d} \quad & \Gamma, \\
\text{s.t.} \quad & \sum_{e \in \mathcal{O}(i)} g^{\kappa}_{e,sd} - \sum_{e \in \mathcal{I}(i)} g^{\kappa}_{e,sd} = 1, && \forall s, d, \forall i = s, \\
& \sum_{e \in \mathcal{O}(i)} g^{\kappa}_{e,sd} - \sum_{e \in \mathcal{I}(i)} g^{\kappa}_{e,sd} = 0, && \forall s, d, \forall i \neq s, d, \\
& \sum_{s} \sum_{d} y\left(\hat{x}^{k+1}_{e,\kappa}\right) \leq \Gamma, && \forall e \in \mathcal{L}, \\
& 0 \leq g^{\kappa}_{e,sd} \leq 1, \quad \Gamma \geq 0, && \forall e \in \mathcal{L}, \forall s, d,
\end{aligned} \tag{5.28}$$

where $\mathcal{O}(i)$ and $\mathcal{I}(i)$ represent the sets of egress and ingress edges of a node i, respectively. We solve (5.28) using the solver SDPT3 of the convex optimization package CVX [57, 56].

Average congestion and average congestion uncertainties are shown in Figure 5.5. We point out 2 important observations here: 1) as expected, the squared link load metric better mitigates uncertainty for all average ages of measurements in comparison to the linear link load metric, and hence improves the worst case network utilization; and 2) the squared link load metric constitutes lower average congestion until the mean delivery duration reaches 100 seconds. Intuitively, when measurement ages are too large, predicted congestion costs becomes negligible in comparison to uncertainty costs, and hence measurement value information can not be utilized in

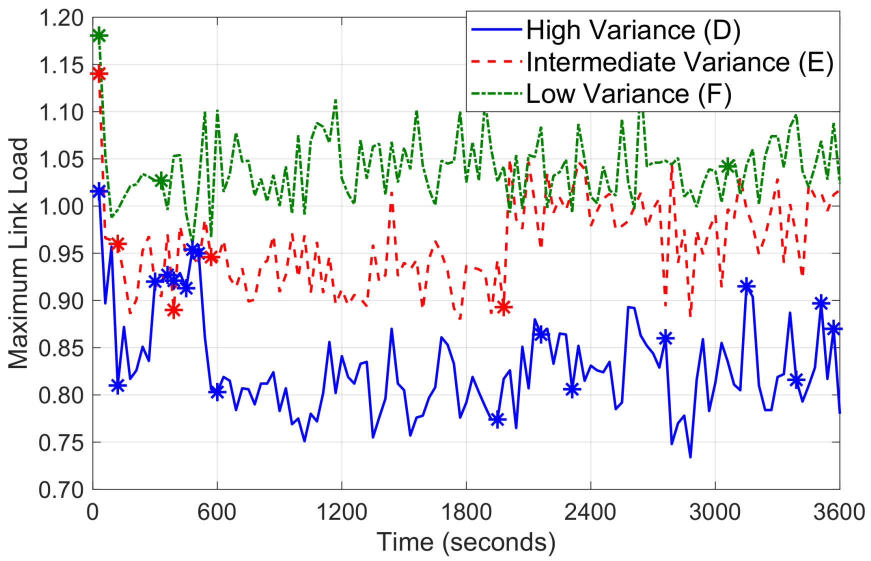


Figure 5.6: The Maximum Link Load Performance of the Proposed System Under Different Traffic Inter-arrival Times. Observe that the Generated Traffic Amounts Are Not Equal for Distinct Scenarios, and Hence Maximum Link Load Does Not Constitute a Fair Performance Comparison. (D,E,F)

computing routing rules. Consequently, the linear metric yields lower average congestion for higher ages of measurements.

5.6.4 Impact of Traffic Non-Stationarity

We compare the reconfiguration frequencies under low, intermediate and high variance, non-stationary traffic conditions. Because of limited space, we only present the simulations obtained using linear link load metric. As illustrated in Figure 5.6, the employed system successfully mitigates network congestion and sustain network utilization against non-stationary traffic fluctuations in all 3 experiments. We further observe that the reconfiguration frequency increases along with the increase in traffic inter-arrival time (See Table 5.1). Clearly, a rapidly changing traffic environment enforces the controller to reconfigure the network more often, since the gains achieved by deploying the candidate link weights will surpass the monotonically decreasing reconfiguration cost more often.

5.7 Conclusions

The traffic engineering problem had been extensively studied in the literature as a robust and effective way of routing traffic in conventional networks. Yet, the impact of measurement age on the performance of traffic engineering techniques under rapidly changing traffic environments has not been investigated. To this end, this chapter presents a first step to mathematically define the impact of measurement age on distributed routing by introducing a model, which incorporates dynamic traffic demands and measurement ages. We formulate 2 single stage predictors for 2 distinct link load metrics, which takes into account the ages of measurements. Finally, we formulate a decision-making problem for reconfiguring the network with new link weights and employ a myopic policy to iteratively solve this problem. We demonstrate, through simulations, the performance of our system and the trade-offs between link load metrics under different measurement age scenarios.

Chapter 6

CONCLUSION

In this dissertation, edge networks with focus on distributed learning and adaptive algorithms are studied. Chapter 2 demonstrated a technique for the coalescence of multiple generative models, which finds applications in the accumulation of data on an edge network. Chapter 3 proposed a federated learning framework with power allocation constraints in wireless setting. The problem at hand is effectively solved by decomposition of the optimization problem. Chapter 4 presented a federated learning based technique for profit maximization of charging stations in the absence of private EV owner information. In the following, the main contributions and possible future directions are discussed. Chapter 5 studied a time varying traffic engineering problem with considerations of age of measurements. Designed NP-hard optimization problem is solved using an adaptive load-sensitive technique.

Generative models often find applications in computer vision and image generation. However, a key advantage of generative models is that they can compress the distributional information of a dataset into a deterministic model. This property can be advantageous in edge learning problems, since an important challenge in edge learning is the transfer of empirical information across edge devices. In addition to compression, generative models further constitute the privacy of personal data, since the original dataset could almost never be reproduced by the generative model. Instead, a generative model generates unique data samples with the same set of features of the original data samples. The key question addressed in Chapter 2 is whether a more compact expression of multiple datasets or generative models be obtained by a single generative model. Such a compact expression is possible via the use of analytical tools from Optimal Transport theory. In particular, the Wasserstein barycenter problem is formulated in terms of generative models and a Wasserstein-1 cost can be minimized. The minimizer is a barycenter model of multiple generative models and comprises the empirical distribution information residing in initial generative models. Experimental studies confirms analytical findings. The use of barycenter model in edge networks for faster training of new generative models is envisaged. More use cases of the developed fast adaptation technique is yet to be explored.

Federated learning has gained popularity over the last decade. However, the application of federated learning on wireless medium with power allocation constraints is still an open research field. In an attempt to narrow this gap, the impacts of wireless environment and power constraints on federated learning are investigated in this dissertation. A joint optimization problem in terms of these constraints is formulated. Due to the entanglement of gradient information and power constraints at each edge user, the optimization problem is shown to be NP-hard. However, a novel decomposition of the optimization problem, which can be efficiently solved, is formulated at the cost of additional bandwidth. Experimental results confirm our analytical results and show that the decomposed optimization problem and the accompanying algorithm minimizes the bias and variance of the received gradient signal. Therefore, the proposed algorithm facilitates a decent convergence to minimum cost.

The biggest obstacle for widespread of electric vehicles is the long charging duration in comparison to gasoline fueled vehicles. EV charging duration may further increase if the charging demand concentrates at certain charging stations at certain times of the day. To this end, efficient demand reshaping policies tailored for EV charging has to be developed and deployed to mitigate catastrophic effects of charging congestion. However, the lack of private EV owner information poses a significant challenge to predict EV charging demand. In this dissertation, a federated learning based technique is developed to predict EV charging demand across charging stations by leveraging the data collected at charging stations. Trained model is subsequently employed to compute optimal pricing policy, which will ensure maximum profit for charging stations as well as shorter queues for EV charging, improving the EV charging experience for users.

The traffic engineering problem had been extensively studied in the literature as a robust and effective way of routing traffic in conventional networks. Yet, the impact of measurement age on the performance of traffic engineering techniques under rapidly changing traffic environments has not been investigated. To this end, Chapter 5 mathematically defines the impact of measurement age on distributed routing by introducing a model, which incorporates dynamic traffic demands and measurement ages. The proposed link weight update algorithm takes into account the dynamic traffic demands as well as age of traffic measurements to ensure robustness. Experimental studies illustrate that the proposed algorithm provides more robust traffic engineering in comparison to the state-of-the-art techniques with high efficiency. The expansion of this technique to software defined networks (SDN) is possible and, indeed, SDN constitutes a less demanding linear optimization problem. However, the solution of this linear program still needs convex numerical solvers and requires the development of age-aware control of link weights, which needs further exploration.

BIBLIOGRAPHY

[1] Abari, O., H. Rahul and D. Katabi, “Over-the-air function computation in sensor networks”, arXiv preprint arXiv:1612.02307 (2016).

[2] Agueh, M. and G. Carlier, “Barycenters in the wasserstein space”, SIAM Journal on Mathematical Analysis **43**, 2, 904–924 (2011).

[3] Ahn, J.-H., O. Simeone and J. Kang, “Wireless federated distillation for distributed edge learning with heterogeneous data”, in “2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)”, pp. 1–6 (IEEE, 2019).

[4] Aji, A. F. and K. Heafield, “Sparse communication for distributed gradient descent”, in “Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing”, pp. 440–445 (2017).

[5] Akyildiz, I. F., A. Lee, P. Wang, M. Luo and W. Chou, “A roadmap for traffic engineering in SDN-openflow networks”, Comput. Netw. **71**, 1–30 (2014).

[6] Akyildiz, I. F., A. Lee, P. Wang, M. Luo and W. Chou, “Research challenges for traffic engineering in software defined networks”, IEEE Network **30**, 3, 52–58 (2016).

[7] Alistarh, D., D. Grubic, J. Li, R. Tomioka and M. Vojnovic, “Qsgd: Communication-efficient sgd via gradient quantization and encoding”, in “Advances in Neural Information Processing Systems”, pp. 1709–1720 (2017).

[8] Alistarh, D., T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat and C. Renggli, “The convergence of sparsified gradient methods”, in “Advances in Neural Information Processing Systems”, pp. 5973–5983 (2018).

[9] Ambrosio, L., “Lecture notes on optimal transport problems”, in “Mathematical aspects of evolving interfaces”, pp. 1–52 (Springer, 2003).

[10] Ambrosio, L., N. Gigli and G. Savaré, *Gradient flows: in metric spaces and in the space of probability measures* (Springer Science & Business Media, 2008).

[11] Amiri, M. M. and D. Gündüz, “Federated learning over wireless fading channels”, arXiv preprint arXiv:1907.09769 (2019).

[12] Amiri, M. M. and D. Gündüz, “Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air”, arXiv preprint arXiv:1901.00844 (2019).

[13] Amiri, M. M. and D. Gündüz, “Over-the-air machine learning at the wireless edge”, in “2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)”, pp. 1–5 (IEEE, 2019).

[14] Anderes, E., S. Borgwardt and J. Miller, "Discrete wasserstein barycenters: Optimal transport for discrete data", Mathematical Methods of Operations Research **84**, 2, 389–409 (2016).

[15] Andreev, S., A. Pyattaev, K. Johnsson, O. Galinina and Y. Koucheryavy, "Cellular traffic offloading onto network-assisted device-to-device connections", IEEE Commun. Mag. **52**, 4, 20–31 (2014).

[16] Arjovsky, M., S. Chintala and L. Bottou, "Wasserstein gan", arXiv preprint arXiv:1701.07875 (2017).

[17] Arora, S., R. Ge, Y. Liang, T. Ma and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (gans)", in "Proceedings of the 34th International Conference on Machine Learning-Volume 70", pp. 224–232 (JMLR. org, 2017).

[18] Bae, S., T. Zeng, B. Travacca and S. Moura, "Inducing human behavior to alleviate overstay at pev charging station", in "2020 American Control Conference (ACC)", pp. 2388–2394 (IEEE, 2020).

[19] Banner, R. and A. Orda, "Multipath routing algorithms for congestion minimization", IEEE/ACM Trans. Netw. **15**, 2, 413–424 (2007).

[20] Bernstein, J., Y.-X. Wang, K. Azizzadenesheli and A. Anandkumar, "Signsgd: Compressed optimisation for non-convex problems", in "International Conference on Machine Learning", pp. 559–568 (2018).

[21] Bertsekas, D. P., *Dynamic programming and optimal control*, vol. 1 (Athena scientific Belmont, MA, 2005).

[22] Boissard, E., T. Le Gouic, J.-M. Loubes *et al.*, "Distribution's template estimate with wasserstein metrics", Bernoulli **21**, 2, 740–759 (2015).

[23] Breiman, L., "Bagging predictors", Machine learning **24**, 2, 123–140 (1996).

[24] Brenier, Y., "Polar factorization and monotone rearrangement of vector-valued functions", Communications on Pure and Applied Mathematics **44**, 4, 375–417 (1991).

[25] Cao, X., G. Zhu, J. Xu and K. Huang, "Optimal power control for over-the-air computation in fading channels", arXiv preprint arXiv:1906.06858 (2019).

[26] ChargePoint, "Driver's Checklist: A Quick Guide to Fast Charging", URL `https://www.chargepoint.com/files/Quick_Guide_to_Fast_Charging.pdf` (Accessed online, 2021).

[27] Chen, H., H. Zhang, Z. Hu, Y. Liang, H. Luo and Y. Wang, "Plug-in electric vehicle charging congestion analysis using taxi travel data in the central area of beijing", arXiv preprint arXiv:1712.07300 (2017).

[28] Choi, D. I. and D.-E. Lim, "Analysis of the state-dependent queueing model and its application to battery swapping and charging stations", Sustainability **12**, 6, 2343 (2020).

[29] Chong, M. J. and D. Forsyth, "Effectively unbiased fid and inception score and where to find them", (2019).

[30] Cisco, "Configuring OSPF", (1997).

[31] Cisco, "Cisco visual networking index: Forecast and methodology, 2016–2021", Tech. rep. (2017).

[32] Cuturi, M., "Sinkhorn distances: Lightspeed computation of optimal transport", in "Advances in Neural Information Processing Systems 26", pp. 2292–2300 (Curran Associates, Inc., 2013).

[33] Cuturi, M. and A. Doucet, "Fast computation of wasserstein barycenters", in "International Conference on Machine Learning", pp. 685–693 (2014).

[34] Cuturi, M. and G. Peyré, "A smoothed dual approach for variational wasserstein problems", SIAM Journal on Imaging Sciences **9**, 1, 320–343 (2016).

[35] De Lange, M., R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks", arXiv preprint arXiv:1909.08383 (2019).

[36] Dean H., L., O. Ariel, R. Danny and Y. Shavitt, "How good can IP routing be?", Tech. rep., DIMACS (2001).

[37] Dedeoglu, M., T.-C. Chiu and J. Zhang, "The impact of traffic information age on congestion mitigation", in "2019 IEEE Global Communications Conference (GLOBECOM)", pp. 1–6 (2019).

[38] Dedeoglu, M., S. Lin and J. Zhang, "Synthetic EV Database", Data can be retrieved from the link below, `https://www.dropbox.com/sh/6suckq6xepl7y8k/AADGm5Ch9kYztT4BBLwx9exWa?dl=0` (2021).

[39] Dedeoglu, M., S. Lin, Z. Zhang and J. Zhang, "Continual learning of generative models with limited data: From wasserstein-1 barycenter to adaptive coalescence", (2021).

[40] Dedeoglu, M., J. Zhang and R. Liang, "Emotion classification based on audiovisual information fusion using deep learning", in "2019 International Conference on Data Mining Workshops (ICDMW)", pp. 131–134 (2019).

[41] Demestichas, P., A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong and J. Yao, "5G on the horizon: Key challenges for the radio-access network", IEEE Veh. Technol. Mag. **8**, 3, 47–53 (2013).

[42] Deng, L., "The mnist database of handwritten digit images for machine learning research", IEEE Signal Processing Magazine **29**, 6, 141–142 (2012).

[43] Destounis, A., S. Paris, L. Maggi, G. S. Paschos and J. Leguay, "Minimum cost SDN routing with reconfiguration frequency constraints", IEEE/ACM Trans. on Netw. **26**, 4, 1577–1590 (2018).

[44] Dhar, P., R. V. Singh, K.-C. Peng, Z. Wu and R. Chellappa, "Learning without memorizing", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 5138–5146 (2019).

[45] Dong, J., Y. Shi and Z. Ding, "Blind over-the-air computation and data fusion via provable wirtinger flow", arXiv preprint arXiv:1811.04644 (2018).

[46] Durugkar, I., I. Gemp and S. Mahadevan, "Generative multi-adversarial networks", arXiv preprint :1611.01673 (2016).

[47] ElectricVehicleChargingStationData, URL `https://services.arcgis.com/ePKBjXrBZ2vEEgWd/arcgis/rest/services/ElectricVehicleChargingStationData/FeatureServer`, accessed Sep. 10, 2021 (????).

[48] Elwalid, A., C. Jin, S. Low and I. Widjaja, "MATE: MPLS adaptive traffic engineering", in "Proc. IEEE INFOCOM", vol. 3, pp. 1300–1309 (2001).

[49] ESIOS, "Esios electricity market data", in "Proceedings of the Tenth International Conference on Future Energy Systems", e-Energy '19 (2019).

[50] Feldmann, A., A. Greenberg, C. Lund, N. Reingold, J. Rexford and F. True, "Deriving traffic demands for operational ip networks: methodology and experience", IEEE/ACM Trans. Netw. **9**, 3, 265–279 (2001).

[51] Flammini, M. G., G. Prettico, A. Julea, G. Fulli, A. Mazza and G. Chicco, "Statistical characterisation of the real transaction data gathered from electric vehicle charging stations", Electric Power Systems Research **166**, 136–150 (2019).

[52] Fortz, B., J. Rexford and M. Thorup, "Traffic engineering with traditional IP routing protocols", IEEE Commun. Mag. **40**, 10, 118–124 (2002).

[53] Fortz, B. and M. Thorup, "Internet traffic engineering by optimizing OSPF weights", in "Proc. IEEE INFOCOM 2000", vol. 2, pp. 519–528 (2000).

[54] Goldenbaum, M. and S. Stanczak, "Robust analog function computation via wireless multiple-access channels", IEEE Transactions on Communications **61**, 9, 3863–3877 (2013).

[55] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", in "Advances in neural information processing systems", pp. 2672–2680 (2014).

[56] Grant, M. and S. Boyd, "Graph implementations for nonsmooth convex programs", in "Recent Advances in Learning and Control", edited by V. Blondel, S. Boyd and H. Kimura, Lecture Notes in Control and Information Sciences, pp. 95–110 (Springer-Verlag Limited, 2008), `http://stanford.edu/~boyd/graph_dcp.html`.

[57] Grant, M. and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1", `http://cvxr.com/cvx` (2014).

[58] Grnarova, P., K. Y. Levy, A. Lucchi, N. Perraudin, I. Goodfellow, T. Hofmann and A. Krause, "A domain agnostic measure for monitoring and evaluating gans", in "Advances in Neural Information Processing Systems 32", pp. 12092–12102 (Curran Associates, Inc., 2019).

[59] Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin and A. C. Courville, "Improved training of wasserstein gans", in "Advances in Neural Information Processing Systems 30", edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, pp. 5767–5777 (Curran Associates, Inc., 2017).

[60] Guo, Y., Z. Wang, X. Yin, X. Shi and J. Wu, "Traffic engineering in SDN/OSPF hybrid network", in "Proc. IEEE ICNP", pp. 563–568 (2014).

[61] Han, S., H. Mao and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding", arXiv preprint arXiv:1510.00149 (2015).

[62] Hardy, C., E. Le Merrer and B. Sericola, "Md-gan: Multi-discriminator generative adversarial networks for distributed datasets", in "2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)", pp. 866–877 (IEEE, 2019).

[63] He, Z. and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 11438–11446 (2019).

[64] Henri Overbeek, "Why is (DC) fast charging sometimes slow?", URL `https://www.linkedin.com/pulse/why-dc-fast-charging-sometimes-slow-henri-overbeek-/` (Accessed online, 2021).

[65] Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium", in "Advances in Neural Information Processing Systems 30", edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, pp. 6626–6637 (Curran Associates, Inc., 2017).

[66] Hossain, E., M. Rasti, H. Tabassum and A. Abdelnasser, "Evolution toward 5G multi-tier cellular wireless networks: An interference management perspective", IEEE Wireless Commun. **21**, 3, 118–127 (2014).

[67] Hu, L., J. Dong and Z. Lin, "Modeling charging behavior of battery electric vehicle drivers: A cumulative prospect theory based approach", Transportation Research Part C: Emerging Technologies **102**, 474–489 (2019).

[68] Hu, Z., K. Zhan, H. Zhang and Y. Song, "Pricing mechanisms design for guiding electric vehicle charging to fill load valley", Applied Energy **178**, 155–163 (2016).

[69] IHS Markit, "Internet of things: a movement, not a market, *Technical Report*", URL `[Online].Available:{https://cdn.ihs.com/www/pdf/IoT_ebook.pdf}.[Accessed{Apr.}26,2020]` (2017).

[70] IHS Markit, "Internet of Things: a movement, not a market, White paper", URL `https://cdn.ihs.com/www/pdf/IoT_ebook.pdf` (2017).

[71] Ivkin, N., D. Rothchild, E. Ullah, V. Braverman, I. Stoica and R. Arora, "Communication-efficient distributed sgd with sketching", arXiv preprint arXiv:1903.04488 (2019).

[72] K. Bradsher, "China Outlines Plans for Making Electric Cars", New York:New York Times URL `https://www.nytimes.com/2009/04/11/business/energy-environment/11electric.html` (Accessed online, 2021).

[73] Karimireddy, S. P., Q. Rebjock, S. U. Stich and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes", arXiv preprint arXiv:1901.09847 (2019).

[74] Kaul, S., R. Yates and M. Gruteser, "Real-time status: How often should one update?", in "Proc. IEEE INFOCOM", pp. 2731–2735 (2012).

[75] Kaur, H. T., T. Ye, S. Kalyanaraman and K. S. Vastola, "Minimizing packet loss by optimizing OSPF weights using online simulation", in "Proc. IEEE/ACM MASCOTS", pp. 79–86 (2003).

[76] Kevin Doyle, "Level Up Your EV Charging Knowledge", URL `https://www.chargepoint.com/blog/level-your-ev-charging-knowledge/` (Accessed online, 2021).

[77] Kirkpatrick, J., R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks", Proceedings of the national academy of sciences **114**, 13, 3521–3526 (2017).

[78] Konečný, J., H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh and D. Bacon, "Federated learning: Strategies for improving communication efficiency", arXiv preprint arXiv:1610.05492 (2016).

[79] Korotin, A., V. Egiazarian, A. Asadulaev, A. Safin and E. Burnaev, "Wasserstein-2 generative networks", in "International Conference on Learning Representations", (2021), URL `https://openreview.net/forum?id=bEoxzW_EXsa`.

[80] Krizhevsky, A., "Learning multiple layers of features from tiny images", Tech. rep. (2009).

[81] Le Gouic, T. and J.-M. Loubes, "Existence and consistency of wasserstein barycenters", Probability Theory and Related Fields **168**, 3-4, 901–917 (2017).

[82] Ledig, C., L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network", (2017).

[83] Lee, S.-W., J.-H. Kim, J. Jun, J.-W. Ha and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching", in "Advances in neural information processing systems", pp. 4652–4662 (2017).

[84] Lee, Z. J., T. Li and S. H. Low, "ACN-Data: Analysis and Applications of an Open EV Charging Dataset", in "Proceedings of the Tenth International Conference on Future Energy Systems", e-Energy '19 (2019).

[85] Lei, N., K. Su, L. Cui, S.-T. Yau and D. X. Gu, "A geometric view of optimal transportation and generative model", (2017).

[86] Leontev, M. I., V. Islenteva and S. V. Sukhov, "Non-iterative knowledge fusion in deep convolutional neural networks", Neural Processing Letters **51**, 1, 1–22 (2020).

[87] Leygonie, J., J. She, A. Almahairi, S. Rajeswar and A. C. Courville, "Adversarial computation of optimal transport maps", CoRR **abs/1906.09691** (2019).

[88] Li, H., Z. Wan and H. He, "Constrained ev charging scheduling based on safe deep reinforcement learning", IEEE Transactions on Smart Grid **11**, 3, 2427–2439 (2020).

[89] Lin, S., M. Dedeoglu and J. Zhang, *Accelerating Distributed Online Meta-Learning via Multi-Agent Collaboration under Limited Communication*, p. 261–270 (Association for Computing Machinery, New York, NY, USA, 2021), URL `https://doi.org/10.1145/3466772.3467055`.

[90] Lin, S., G. Yang and J. Zhang, "A collaborative learning framework via federated meta-learning", arXiv preprint arXiv:2001.03229 (2020).

[91] Liu, H., X. Gu and D. Samaras, "Wasserstein gan with quadratic transport cost", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 4832–4841 (2019).

[92] Liu, W. and X. Zang, "Over-the-air computation systems: Optimization, analysis and scaling laws", arXiv preprint arXiv:1909.00329 (2019).

[93] Makkuva, A., A. Taghvaei, S. Oh and J. Lee, "Optimal transport mapping via input convex neural networks", in "International Conference on Machine Learning", pp. 6672–6681 (PMLR, 2020).

[94] McCann, R. J., “A convexity principle for interacting gases”, Advances in mathematics **128**, 1, 153–179 (1997).

[95] Moschella, M., P. Ferraro, E. Crisostomi and R. Shorten, “Decentralized assignment of electric vehicles at charging stations based on personalized cost functions and distributed ledger technologies”, IEEE Internet of Things Journal (2021).

[96] Murphy, J., R. Harris and R. Nelson, “Traffic engineering using OSPF weights and splitting ratios”, in INTERWORKING pp. 277–287 (2002).

[97] Neyshabur, B., S. Bhojanapalli and A. Chakrabarti, “Stabilizing GAN training with multiple random projections”, CoRR **abs/1705.07831** (2017).

[98] Osia, S. A., A. S. Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane and H. Haddadi, “A hybrid deep learning architecture for privacy-preserving mobile analytics”, IEEE Internet of Things Journal **7**, 5, 4505–4518 (2020).

[99] Ostapenko, O., M. Puscas, T. Klein, P. Jahnichen and M. Nabi, “Learning to remember: A synaptic plasticity driven framework for continual learning”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 11321–11329 (2019).

[100] Paris, S., A. Destounis, L. Maggi, G. S. Paschos and J. Leguay, “Controlling flow reconfigurations in SDN”, in “Proc. IEEE INFOCOM”, pp. 1–9 (2016).

[101] Qi, M., Y. Wang, J. Qin and A. Li, “Ke-gan: Knowledge embedded generative adversarial networks for semi-supervised scene parsing”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 5237–5246 (2019).

[102] Qian, T., C. Shao, X. Wang and M. Shahidehpour, “Deep reinforcement learning for ev charging navigation by coordinating smart grid and intelligent transportation system”, IEEE Transactions on Smart Grid **11**, 2, 1714–1723 (2020).

[103] Radford, A., L. Metz and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks”, arXiv preprint arXiv:1511.06434 (2015).

[104] Rebuffi, S.-A., A. Kolesnikov, G. Sperl and C. H. Lampert, “icarl: Incremental classifier and representation learning”, in “Proceedings of the IEEE conference on Computer Vision and Pattern Recognition”, pp. 2001–2010 (2017).

[105] Ren, Y., Y. Zhou and L. Shi, “Decision-making approach in charging mode for electric vehicle based on cumulative prospect theory”, in “2012 China International Conference on Electricity Distribution”, pp. 1–4 (IEEE, 2012).

[106] Ren, Y., Y. Zhou and L. Shi, “Decision-making approach in charging mode for electric vehicle based on cumulative prospect theory”, in “2012 China International Conference on Electricity Distribution”, pp. 1–4 (IEEE, 2012).

[107] Riemer, M., T. Klinger, D. Bouneffouf and M. Franceschini, “Scalable recollections for continual lifelong learning”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 1352–1359 (2019).

[108] Rolnick, D., A. Ahuja, J. Schwarz, T. Lillicrap and G. Wayne, “Experience replay for continual learning”, in “Advances in Neural Information Processing Systems”, pp. 350–360 (2019).

[109] Schapire, R. E., “A brief introduction to boosting”, in “Ijcai”, vol. 99, pp. 1401–1406 (1999).

[110] Schwarz, J., J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu and R. Hadsell, “Progress & compress: A scalable framework for continual learning”, arXiv preprint arXiv:1805.06370 (2018).

[111] Seguy, V., B. B. Damodaran, R. Flamary, N. Courty, A. Rolet and M. Blondel, “Large-scale optimal transport and mapping estimation”, arXiv preprint arXiv:1711.02283 (2017).

[112] Sery, T. and K. Cohen, “On analog gradient descent learning over multiple access fading channels”, arXiv preprint arXiv:1908.07463 (2019).

[113] Shafiee, M. J., F. Li, B. Chwyl and A. Wong, “Squishednets: Squishing squeezenet further for edge device scenarios via deep evolutionary synthesis”, arXiv preprint arXiv:1711.07459 (2017).

[114] Shahriar, S., A. R. Al-Ali, A. H. Osman, S. Dhou and M. Nijim, “Machine learning approaches for ev charging behavior: A review”, IEEE Access **8**, 168980–168993 (2020).

[115] Shi, S., Q. Wang, K. Zhao, Z. Tang, Y. Wang, X. Huang and X. Chu, “A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks”, in “2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)”, pp. 2238–2247 (IEEE, 2019).

[116] Shin, H., J. K. Lee, J. Kim and J. Kim, “Continual learning with deep generative replay”, in “Advances in Neural Information Processing Systems”, pp. 2990–2999 (2017).

[117] Shrivastava, A., T. Pfister, O. Tuzel, J. Susskind, W. Wang and R. Webb, “Learning from simulated and unsupervised images through adversarial training”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 2107–2116 (2017).

[118] Simon, D. and A. Aberdam, “Barycenters of natural images constrained wasserstein barycenters for image morphing”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 7910–7919 (2020).

[119] Singh, S. P. and M. Jaggi, “Model fusion via optimal transport”, arXiv preprint arXiv:1910.05653 (2019).

[120] Sinha, A., H. Namkoong, R. Volpi and J. Duchi, "Certifying some distributional robustness with principled adversarial training", arXiv preprint arXiv:1710.10571 (2017).

[121] Smith, J. and M. Gashler, "An investigation of how neural networks learn from the experiences of peers through periodic weight averaging", in "2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)", pp. 731–736 (IEEE, 2017).

[122] Srivastava, S., V. Cevher, Q. Dinh and D. Dunson, "Wasp: Scalable bayes via barycenters of subset posteriors", in "Artificial Intelligence and Statistics", pp. 912–920 (2015).

[123] Staib, M., S. Claici, J. M. Solomon and S. Jegelka, "Parallel streaming wasserstein barycenters", in "Advances in Neural Information Processing Systems", pp. 2647–2658 (2017).

[124] Stich, S. U., "Local sgd converges fast and communicates little", in "ICLR 2019 International Conference on Learning Representations", (2019).

[125] Stich, S. U., J.-B. Cordonnier and M. Jaggi, "Sparsified sgd with memory", in "Advances in Neural Information Processing Systems", pp. 4447–4458 (2018).

[126] Sun, Y., E. Uysal-Biyikoglu, R. Yates, C. E. Koksal and N. B. Shroff, "Update or wait: How to keep your data fresh", in "Proc. IEEE INFOCOM", pp. 1–9 (2016).

[127] Taghvaei, A. and A. Jalali, "2-wasserstein approximation via restricted convex potentials with application to improved training for gans", arXiv preprint arXiv:1902.07197 (2019).

[128] Thrun, S., "A lifelong learning perspective for mobile robot control", in "Intelligent robots and systems", pp. 201–214 (Elsevier, 1995).

[129] Tversky, A. and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty", Journal of Risk and uncertainty **5**, 4, 297–323 (1992).

[130] v. Adrichem, N. L. M., B. J. v. Asten and F. A. Kuipers, "Fast recovery in software-defined networks", in "Third European Workshop on Software Defined Networks", pp. 61–66 (2014).

[131] Venkataramani, S., A. Ranjan, S. Banerjee, D. Das, S. Avancha, A. Jagannathan, A. Durg, D. Nagaraj, B. Kaul, P. Dubey *et al.*, "Scaledeep: A scalable compute architecture for learning and evaluating deep networks", in "Proceedings of the 44th Annual International Symposium on Computer Architecture", pp. 13–26 (2017).

[132] Villani, C., *Topics in optimal transportation*, no. 58 (American Mathematical Soc., 2003).

[133] Villani, C., *Optimal transport: old and new*, vol. 338 (Springer Science & Business Media, 2008).

[134] Vissicchio, S., O. Tilmans, L. Vanbever and J. Rexford, “Central control over distributed routing”, in “Proc. ACM SIGCOMM”, pp. 43–56 (2015).

[135] Vissicchio, S., O. Tilmans, L. Vanbever and J. Rexford, “Central control over distributed routing (extended version)”, Tech. rep. (2015).

[136] Volpi, R., H. Namkoong, O. Sener, J. C. Duchi, V. Murino and S. Savarese, “Generalizing to unseen domains via adversarial data augmentation”, in “Advances in Neural Information Processing Systems”, pp. 5334–5344 (2018).

[137] Wan, Z., H. Li, H. He and D. Prokhorov, “Model-free real-time ev charging scheduling based on deep reinforcement learning”, IEEE Transactions on Smart Grid **10**, 5, 5246–5257 (2019).

[138] Wang, H. and M. R. Ito, “Dynamics of load-sensitive adaptive routing”, in “Proc. IEEE ICC 2005”, vol. 1, pp. 213–217 Vol. 1 (2005).

[139] Wang, J., W. Bao, L. Sun, X. Zhu, B. Cao and S. Y. Philip, “Private model compression via knowledge distillation”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 1190–1197 (2019).

[140] Wang, J., J. Zhang, W. Bao, X. Zhu, B. Cao and P. S. Yu, “Not just privacy: Improving performance of private deep learning in mobile cloud”, in “Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining”, pp. 2407–2416 (2018).

[141] Wang, J., W. Zhou, G.-J. Qi, Z. Fu, Q. Tian and H. Li, “Transformation gan for unsupervised image synthesis and representation learning”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 472–481 (2020).

[142] Wang, Q., Y. Li, B. Shao, S. Dey and P. Li, “Energy efficient parallel neuromorphic architectures with approximate arithmetic on fpga”, Neurocomputing **221**, 146–158 (2017).

[143] Wang, S., S. Bi and Y. A. Zhang, “Reinforcement learning for real-time pricing and scheduling control in ev charging stations”, IEEE Transactions on Industrial Informatics **17**, 2, 849–859 (2021).

[144] Wang, S., S. Bi, Y.-J. A. Zhang and J. Huang, “Electrical vehicle charging station profit maximization: Admission, pricing, and online scheduling”, IEEE Transactions on Sustainable Energy **9**, 4, 1722–1731 (2018).

[145] Wang, S., S. Bi, Y.-J. A. Zhang and J. Huang, “Electrical vehicle charging station profit maximization: Admission, pricing, and online scheduling”, IEEE Transactions on Sustainable Energy **9**, 4, 1722–1731 (2018).

[146] Wang, Y. and Z. Wang, “Explicit routing algorithms for internet traffic engineering”, in “Proc. ICCN’99”, pp. 582–588 (1999).

[147] Wang, Y., C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia and B. Raducanu, “Transferring gans: generating images from limited data”, in “Proceedings of the European Conference on Computer Vision (ECCV)”, pp. 218–234 (2018).

[148] Wen, D., G. Zhu and K. Huang, “Reduced-dimension design of MIMO over-the-air computing for data aggregation in clustered iot networks”, IEEE Transactions on Wireless Communications **18**, 11, 5255–5268 (2019).

[149] Wen, W., C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen and H. Li, “Terngrad: Ternary gradients to reduce communication in distributed deep learning”, in “Advances in Neural Information Processing Systems”, pp. 1509–1519 (2017).

[150] Wu, C., L. Herranz, X. Liu, J. van de Weijer, B. Raducanu *et al.*, “Memory replay gans: Learning to generate new categories without forgetting”, in “Advances in Neural Information Processing Systems”, pp. 5962–5972 (2018).

[151] Wu, S., A. G. Dimakis, S. Sanghavi, F. X. Yu, D. Holtmann-Rice, D. Storcheus, A. Rostamizadeh and S. Kumar, “Learning a Compressed Sensing Measurement Matrix via Gradient Unrolling”, URL `http://arxiv.org/abs/1806.10175` (2018).

[152] Xu, D., M. Chiang and J. Rexford, “Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering”, IEEE/ACM Trans. Netw. **19**, 6, 1717–1730 (2011).

[153] Xu, X., D. Niu, Y. Li and L. Sun, “Optimal pricing strategy of electric vehicle charging station for promoting green behavior based on time and space dimensions”, Journal of Advanced Transportation **2020** (2020).

[154] Xu, X., D. Niu, Y. Li and L. Sun, “Optimal pricing strategy of electric vehicle charging station for promoting green behavior based on time and space dimensions”, Journal of Advanced Transportation **2020** (2020).

[155] Yang, K., T. Jiang, Y. Shi and Z. Ding, “Federated learning via over-the-air computation”, arXiv preprint arXiv:1812.11750 (2018).

[156] Yang, T.-J., Y.-H. Chen and V. Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, pp. 5687–5695 (2017).

[157] Ye, J., P. Wu, J. Z. Wang and J. Li, “Fast discrete distribution clustering using wasserstein barycenter with sparse support”, IEEE Transactions on Signal Processing **65**, 9, 2317–2332 (2017).

[158] Ye, T., H. T. Kaur, S. Kalyanaraman, K. S. Vastola and S. Yadav, “Dynamic optimization of OSPF weights using online simulation”, in “Proc. IEEE INFOCOM”, (2002).

[159] Yonetani, R., T. Takahashi, A. Hashimoto and Y. Ushiku, "Decentralized learning of generative adversarial networks from non-iid data", arXiv preprint arXiv:1905.09684 (2019).

[160] Younis, O. and S. Fahmy, "Constraint-based routing in the internet: Basic principles and recent research", IEEE Commun. Surveys Tuts. **5**, 1, 2–13 (2003).

[161] Yu, F., Y. Zhang, S. Song, A. Seff and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop", CoRR **abs/1506.03365**, URL `http://arxiv.org/abs/1506.03365` (2015).

[162] Yuan, Z., B. Li and J. Liu, "Can we improve information freshness with predictions in mobile crowd-learning?", in "IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)", pp. 702–709 (IEEE, 2020).

[163] Zeng, Q., Y. Du, K. K. Leung and K. Huang, "Energy-efficient radio resource allocation for federated edge learning", arXiv preprint arXiv:1907.06040 (2019).

[164] Zeng, T., H. Zhang and S. Moura, "Solving overstay and stochasticity in pev charging station planning with real data", IEEE Transactions on Industrial Informatics **16**, 5, 3504–3514 (2019).

[165] Zhang, H., C. J. Sheppard, T. E. Lipman and S. J. Moura, "Joint fleet sizing and charging system planning for autonomous electric vehicles", IEEE Transactions on Intelligent Transportation Systems **21**, 11, 4725–4738 (2019).

[166] Zhang, J., N. Li and M. Dedeoglu, "Federated learning over wireless networks: A band-limited coordinated descent approach", in "IEEE INFOCOM 2021 - IEEE Conference on Computer Communications (INFOCOM 2021)", (Vancouver, Canada, 2021).

[167] Zhang, T., X. Chen, B. Wu, M. Dedeoglu, J. Zhang and L. Trajkovic, "Stochastic modeling and analysis of public electric vehicle fleet charging station operations", IEEE Transactions on Intelligent Transportation Systems pp. 1–14 (2021).

[168] Zhang, Z., S. Lin, M. Dedeoglu, K. Ding and J. Zhang, "Data-driven distributionally robust optimization for edge intelligence", in "IEEE INFOCOM 2020 - IEEE Conference on Computer Communications", pp. 2619–2628 (2020).

[169] Zhang, Z., S. Lin, M. Dedeoglu, K. Ding and J. Zhang, "Data-driven distributionally robust optimization for edge intelligence", in "IEEE INFOCOM 2020-IEEE Conference on Computer Communications", pp. 2619–2628 (IEEE, 2020).

[170] Zhou, Z., X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing", Proceedings of the IEEE **107**, 8, 1738–1762 (2019).

[171] Zhu, G. and K. Huang, "MIMO over-the-air computation for high-mobility multi-modal sensing", IEEE Internet of Things Journal (2018).

[172] Zhu, G., D. Liu, Y. Du, C. You, J. Zhang and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning", arXiv preprint arXiv:1809.00343 (2018).

[173] Zhu, G., Y. Wang and K. Huang, "Broadband analog aggregation for low-latency federated edge learning", IEEE Transactions on Wireless Communications (2019).

[174] Zhu, J.-Y., T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks", in "Proceedings of the IEEE international conference on computer vision", pp. 2223–2232 (2017).