Learning at the Edge under Resource Constraints

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Jayanth Reddy Regatti, M.S, B.Tech

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2023

Dissertation Committee:

Prof. Abhishek Gupta, Advisor

Prof. Ness Shroff, Advisor

Prof. Yingbin Liang

Prof. Eylem Ekici

Abstract

Recent decades saw a huge increase in the number of personal devices, wearables, edge devices, etc which led to increased data collection and increased connectivity at the edge. This collected data can be used to make insights about health, the economy, and business and help us make better decisions at the individual, organizational and global levels. With the proliferation of these devices, there are also numerous challenges associated with making use of these devices and the data to train useful models. The challenges could be due to privacy regulations or other constraints determined by the particular learning setup. These constraints make it difficult to extract the required insights from the data and the edge systems. The goal of this thesis is to understand these challenges or resource constraints and develop efficient algorithms that enable us to train models while adhering to the constraints. This thesis makes the following contributions

- Propose an efficient algorithm FedCMA for model heterogeneous Federated Learning under resource constraints, showed the convergence and generalization properties, and demonstrated the efficacy against state-of-the-art algorithms in the model heterogeneity setting.

- Proposed a two-timescale aggregation algorithm that does not require the knowledge of the number of adversaries for defending against Byzantine adversaries in

the distributed setup, proved the convergence of the algorithm, and demonstrated the defense against state-of-the-art attacks.

- We highlight the challenges posed by resource constraints in the Offline Reinforcement Learning setup where the observation space during inference is different from the observation space during training. We propose a simple algorithm STPI (Simultaneous Transfer Policy Iteration) to train the agent to adapt to the changes in the observation space and demonstrated the effectiveness of the algorithm on MuJoCo environments against simple baselines.

This thesis is dedicated to my parents and my wife Shreya.

Acknowledgments

I consider myself genuinely fortunate to be advised by Prof. Abhishek Gupta and Prof. Ness Shroff. Without their constant support, this thesis would have been far from complete. They both gave me the freedom to explore my interests and choose my research problems. My time at the grad school overlapped with the COVID-19 pandemic and I am incredibly grateful to my advisors for letting me work remotely so that I can stay with my wife. I can not imagine how much more stressful it would have been for me without their support. I can't thank Prof. Gupta enough for his guidance. He has been extremely patient in training me throughout my Ph.D. and I have learned a great deal about research and life in general from him. I found in him a friend from whom I can seek advice about my professional and personal life. Prof. Shroff is the epitome of dedication when it comes to research and teaching. I'll remember one particular situation for the rest of my life; despite feeling unwell for a few days, Prof. Shroff was ready and actively asked questions during our weekly meeting as enthusiastic as ever. This makes me ask myself, "What is my excuse" whenever I feel I am slacking at my work.

I am forever grateful to my parents, sister, in-laws, and my wife especially for supporting me through all my lows and making me believe in myself. I still remember the discussion I had with my parents before enrolling in Ph.D. They encouraged me to go back to the university saying "you can earn money anytime but the time to

pursue passions is now". Their life would have been so much smoother had I continued working, but they put my interests ahead of theirs. This thesis wouldn't have come this far without the ceaseless support of my wife. Despite having one of the most coveted jobs, she uprooted her life in India to come to the USA and enrolled in grad school to stay with me. I lack words to thank my wife for staying by my side despite fighting her own battles. Despite being away from the university campus during the Covid-19 pandemic, I was able to stay sane while pursuing research, thanks to my amazing friends. I would like to especially thank Kalyan, Anusha, Harini, Bhargav, Monika, Tejaswi, Sruti, Adyasha, Avilash, Sravya for making the last three years a pleasant ride. I also thank Irene and Fernando for making us feel at home during my stay at Roseville.

I am deeply indebted to the funding agencies (ARPA-E, ARL, NSF) that funded my Ph.D. research. I am also grateful to J.N.TATA Endowment for the gift scholarship. I would like to thank my mentors at Microsoft, Aniket Anand Deshmukh, Urun Dogan, Eren Manavoglu, and Denis Charles for helping me make the best out of my internships at Microsoft, and especially Aniket and Urun for also guiding me throughout my Ph.D. I would like to thank all my collaborators, Dr. Songtao Lu, Prof. Yingbin Liang, Prof. Yi Zhou, Dr. Frank Cheng, Dr. Young Hun Jung, Eren Manavoglu, Gaurav Tendolkar, and Hao Chen. I would like to thank my committee members Prof. Liang and Prof. Ekici for sparing their time and providing me with valuable suggestions to improve my thesis. Finally, I would like to thank the ECE graduate program coordinator, Patricia Toothman, and COEIT admin Alex Morgan for helping me out with all the administrative and technical hurdles I faced.

Vita

2011 - 2015 ..... B.Tech. Electrical Engineering, Indian Institute of Technology Hyderabad.

2016 - 2018 ..... M.S. Electrical and Computer Engineering, The Ohio State University.

2019 - 2022 ..... Graduate Research Associate, PhD student, Electrical and Computer Engineering,The Ohio State University.

Publications

Research Publications

J. Regatti, H. Chen, and A. Gupta, "Byzantine Resilience with Reputation Scores". *In Proc 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2022.

J. Regatti and A. Gupta, "Finite sample analysis of minmax variant of offline reinforcement learning for general MDPs". *IEEE Open Journal of Control Systems*, vol. 1, pp. 152–163, 2022.

A. A. Deshmukh, **J. Regatti**, E. Manavoglu, and U. Dogan, "Representation learning for clustering via building consensus". *Machine Learning*, pp. 1–38, 2022.

J. Regatti, A. A. Deshmukh, E. Manavoglu, and U. Dogan, "Consensus clustering with unsupervised representation learning". *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2021.

J. Regatti, H. You, H. Wang, J. Hall, A. Schnabel, B. Hu, J. Zhang, J. Wang, and A. Gupta, "A discussion of artificial intelligence applications in sic mosfet device operation". *International Electric Machines & Drives Conference (IEMDC), IEEE*, pp. 1–5, 2021.

J. Regatti and A. Gupta, "Traffic-Aware Adaptive Routing for Minimizing Fuel Consumption". *American Control Conference (ACC)*, pp. 4818–4825, IEEE, 2019.

J. Regatti, G. Tendolkar, Y. Zhou, A. Gupta, and Y. Liang, "Distributed SGD Generalizes Well Under Asynchrony". *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, pp. 863–870, 2019.

Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

Page

List of Tables

List of Figures

Chapter 1: Introduction

In the past two decades, there has been a steady increase in the use of digital devices, such as cell phones, wearable devices, watches, IoT (internet of things) devices, smart home appliances, remote sensing devices, etc. Some estimates say that there are around 7 billion IoT devices and 3 billion smartphones in the world [2]. With the use of these devices coupled with increased internet connectivity, there is an abundant collection of data that captures several aspects of human behavior, and human interactions with one another or with systems. Various facets of human lives be it work, education, entertainment, or leisure, are highly intertwined with these digital systems. It is next to impossible to imagine spending one's day without accessing any of these "smart" devices, to the extent that *digital detox* became a thing.

With this abundance of data comes a huge business opportunity to extract patterns of human behavior/physical phenomenon and monetize such patterns. For example, Fig 1.1 shows the tremendous increase in online advertising revenue over the last two decades. This became possible due to the increasing interaction of humans on the internet, with each other on digital platforms, etc. This trend is only expected to increase, with newer technologies like 5G, social media, self-driving vehicles, and blockchain mechanisms showing the potential to integrate even more tightly into human lives. Extracting patterns from data is not just an opportunity for monetization, but

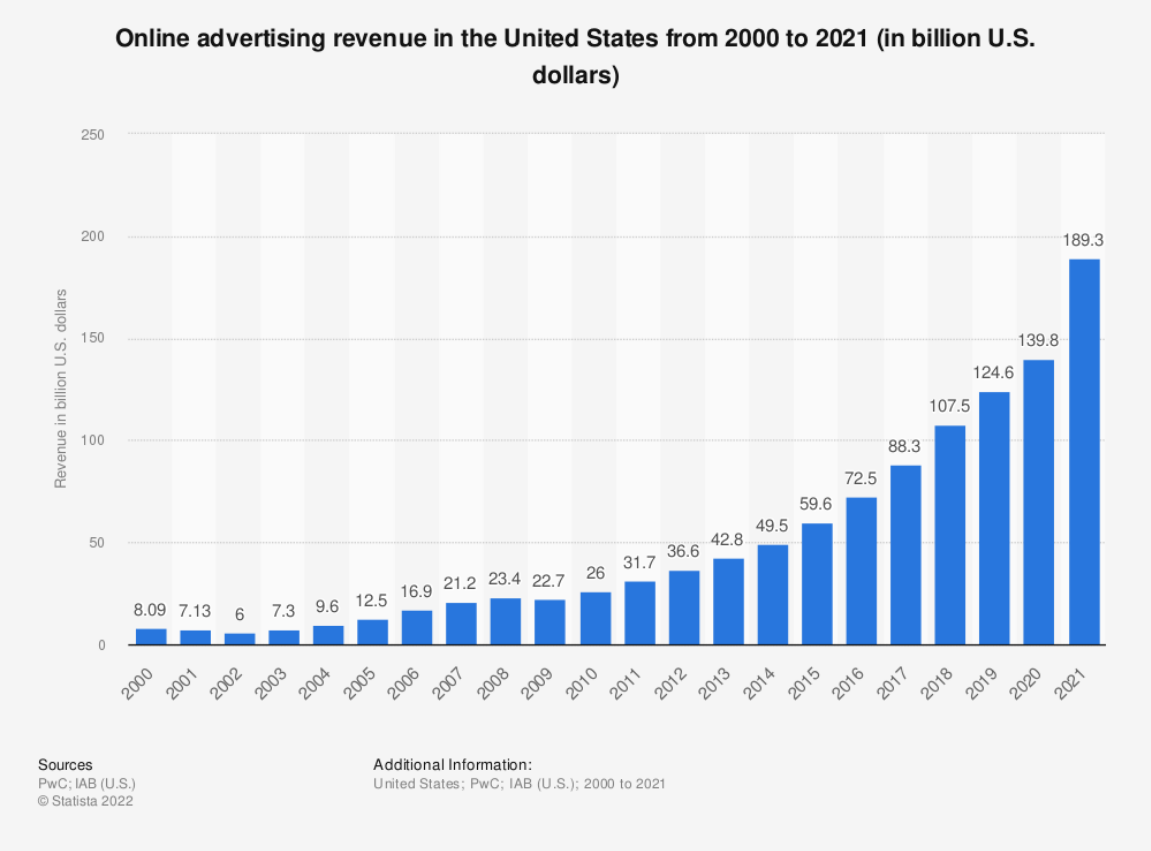


Figure 1.1: Online advertising revenue in the USA [1]

it also impacts (positively or negatively) several other fields such as health care [3], climate change [4], elections [5], etc. The techniques used to extract these patterns and make predictions have been popularly dubbed as *Artificial Intelligence (AI)*.

One straightforward approach to extracting these patterns using the data across scores of such edge devices is to pool the data at a central location or server and train the algorithms using the aggregated data. However, this is not a practical solution, due to several reasons. Two of the most prominent reasons are

1. Sending the data from the edge devices to the central server may involve huge communication costs, or may not be feasible at all times due to intermittent connectivity.

2. Sharing the data from the edge devices with the central server may not be safe for the users concerned.

While the first reason is relatively straightforward, the second one demands more discussion. Monetizing user data is very lucrative and drove innovation so fast that the regulations to control the impact of such businesses (including safeguarding the privacy of the users) could not catch up. Several popular cases showed catastrophic and wide-ranging effects of data breaches, from stolen trade secrets [6] to election interference [7]. Only recently did regulations such as General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) come into effect. The idea of GDPR or CCPA is to collect only data that is consented to by the user and absolutely necessary for the functionality, to safeguard the personal data such that the user can not be identified by using the data for any application.

Studying user/data privacy dates back to the seminal works on secure multi-party computing (MPC) [8] in the 1980s. Yao (1982) [8] poses the *Millionaire problem* where two millionaires want to know which of them is richer, without revealing their wealth to the other millionaire. More generally, each user i has private data x_i. Each user wants to query a function $f(x_1, \cdots, x_i, \cdots, x_n)$ that uses the private data of the other users. Differential privacy [9] is a popularly used theoretical notion to study and provide privacy guarantees. While data privacy was studied earlier for practical applications, such as information retrieval, safeguarding census data, and data mining

[10], it became more important than ever with the success of Machine Learning (ML) or Artificial Intelligence (AI) algorithms in the 2000s and 2010s.

The alternative to storing all the data at the central server is to communicate other useful information such as aggregate statistics of the data, noisy data, model parameters, etc. Machine Learning algorithms require a lot of data and recent works show that the models leak information about training data [11]. With the increasing penetration of ML algorithms into several aspects of business decision making such as credit card, mortgage, and loan approvals, the risk of data leakage has become a real threat to privacy. This problem has been further exacerbated due to the increasing deployment of edge devices that collect, store and process user information.

The focus of this thesis is to study the various aspects of learning involving the edge devices (without sharing the data), with particular emphasis on obeying several resource constraints some of which are generic to edge deployment, and some are problem-dependent constraints.

1.1 The edge

Edge is a term used to describe any device that is capable of collecting data directly, interacting with the end user directly and is far away from a cloud or a centralized server. Edge devices can be ranging from mobile phones, smart watches, to self-driving cars, drones, nanosatellites, etc. See Fig 1.2 for an illustration. Edge deployment is crucial to integrate smart algorithms into real-world applications since they perform key tasks such as collecting data, making predictions, and serving the users.

However, edge devices come with several inherent constraints such as limited memory, computation, power, connectivity to the cloud, and the requirement to

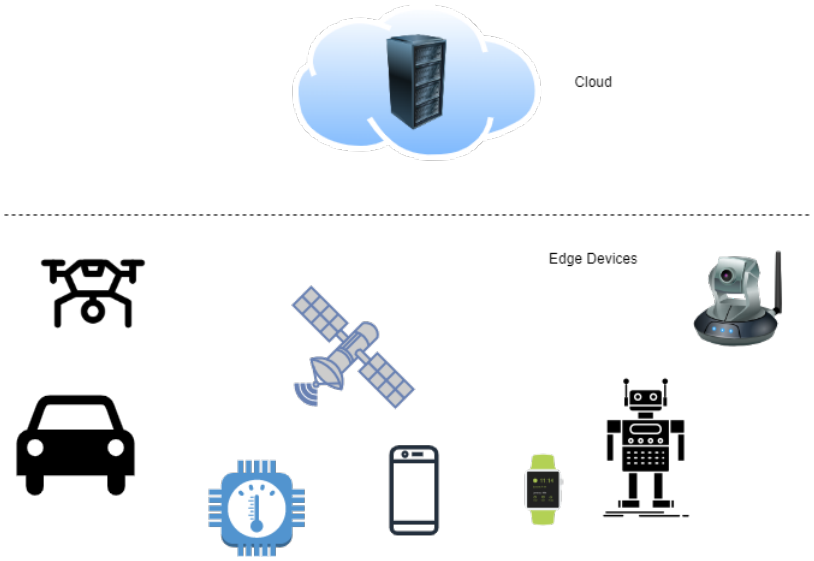


Figure 1.2: Illustration of Edge devices

make faster predictions. Depending on different applications, some or more of these constraints become more important and the learning algorithms need to be explicitly curated to obey these constraints.

Earlier studies provide a detailed survey on edge computing [2], resource constraints [12] and categorize challenges into several different ways. For the purpose of this thesis, we shall use the following broad categories to illustrate the challenges associated with learning at the edge:

1. **Training data**: non-iid, limited in size, corrupted observations
2. **Hardware**: limited capacity and power, heterogeneous and private computational resources
3. **Communication**: poor connectivity, limited communication bandwidth
4. **Security and privacy**: prone to attacks and can not share data.

We collectively dub these challenges as *Resource Constraints* that affect learning at the edge. It is possible that these categories may be interdependent on each other. For example, an edge device with a poor sensor (hardware) may lead to corrupted observations that affect the quality of the training data. Similarly, an edge device with poor communication may not be able to update security patches frequently and is therefore more prone to attacks. Later, in the thesis, we will discuss each of the resource constraints in detail, with motivating examples.

1.1.1 Learning at the edge

While we use the term learning at the edge to collectively refer to all types of computing at the edge, it is important to make an important but nuanced differentiation between (i) inference at the edge and (ii) training at the edge.

Challenges with inference at the edge: Inference or making predictions is one of the most important jobs of an edge device. For example, due to poor hardware (sensors), the observations made by the edge device may be corrupted, and therefore the input data may be from a different distribution than what the algorithm is trained for. This can cause degradation in the performance of the algorithm. Similarly, consider the case where a cloud server trains a huge model and sends it to the edge device for inference. However, due to hardware constraints, the edge device may not have enough memory or computing resources to load the model or to provide real-time predictions (which is a key aspect of edge devices). Therefore, the server must also train a model that fits the hardware of the edge device, which may lead to performance trade-offs.

Challenges with training at the edge: Due to strict regulations to keep the data private, edge devices are often required to train a model on the device. In this case, the device communicates its trained model to a central server or to other partnering edge devices to get a better-aggregated model. This leads to additional challenges such as data heterogeneity where the training data distributions at the different devices are different. In such cases, the training algorithm may not converge properly. In addition to statistical heterogeneity, the training may also be affected by stale updates, adversarial users, etc. Moreover, if the edge device is also tasked to make predictions, the above challenges are compounded by the challenges related to inference.

Please note that in this thesis, unless otherwise specified, training at the edge always refers to training at the edge where there are multiple users that take part in the training. On the other hand, inference at the edge refers to only a single user deployment to make decisions.

Examples of Learning at the edge Let us now list down a few example applications of edge-based learning along with the type of constraints that these applications may encounter.

1. Auto-correct, auto-complete, or next word prediction: Consider an auto-complete feature for an application such as social media messenger, mobile keyboard, email, or document editor that greatly enhances the user experience of using the application. In such cases, efforts are made to tailor the predictions for the user, such as the language, context, etc. For example, WhatsApp and Telegram are two famous messaging apps in India, and many users type regional language words (for example Telugu) typed in English to communicate instead

of changing to the native language on the mobile keyboard. In many cases, the auto-corrected word for an actual English prompt may be completely different from the auto-corrected result for a Telugu word prompt. In such cases, the mobile application greatly benefits from the user's past typing data and similar users' typing data since a single user may have very little data. Moreover, there is a wide range of mobile devices with varying computing power and therefore the app designer needs to take this heterogeneity into consideration.

2. Keystroke detection for mobile keyboard: Popular mobile keyboards such as Microsoft's SwiftKey keyboard, Google's Gboard, and Swype enable a feature to predict a word based on the swiping pattern on the keyboard which enables faster typing on tablets and smartphones. Similar to auto-correct, keystroke detection also varies with language, and slang and is therefore personalized for each user using the user's past data and other similar users' behaviors. The constraints faced are similar to the example above.

3. Photo organizer applications: A key challenge for any photographer is to properly organize the photographs for fast retrieval. A less sophisticated approach is to tag every photograph with the time, date, and other keywords that the user enters so that the photos can be retrieved based on the tags. However, it is too tiresome to tag each of the 100s of photos that a regular smartphone user captures every day. Therefore, object recognition algorithms and image description algorithms are used to auto-generate tags for every photograph, thereby enabling faster retrieval. Since photographs are considered private data, it is often required that the images do not leave the smartphone or computer,

and therefore fast algorithms that can run on mobile devices are required. The heterogeneity of the computing power on mobile devices is also an important constraint in this application.

4. Health diagnostics applications on smart watches, or wearable devices: Smartwatches are used increasingly to track user vitals and these vitals are used to detect or warn a user of any abnormal pattern that may be a symptom of a health ailment. The limited power and memory on a smartwatch constrain the type of models/algorithms that one can use to raise such warnings. Moreover, each user has very limited data and may not have experienced any abnormal patterns in the past. Therefore, it is important for such apps to maintain the privacy and security of the user data while also learning useful models to make accurate predictions.

5. Autonomous vehicles: An autonomous vehicle must make several decisions every second, such as whether to brake or to accelerate, to stop at a signal or to yield to a pedestrian, to accelerate or to slow down at a speed breaker, etc. At every instant, the sensors on the vehicle provide information about the surroundings and the onboard algorithms determine what course of action to take. A fraction of a second delay in making a decision can be fatal, therefore low latency is a key requirement. Moreover, it is important that these systems are secure since an adversarial attack on the algorithm of one autonomous vehicle can cause an accident affecting several more vehicles on the road. Note that, while the real-world adoption of fully autonomous and self-driving cars is not yet a certainty (due to lack of appropriate regulations), level 2 and 3

autonomous driving systems are already widely used that use sophisticated radar and computer vision algorithms to detect obstacles on the road.

6. Diagnostics and Prognostics for mechanical and electronic devices: Detecting and predicting faults is an important way to make devices more reliable and robust to failures. Since the on-device computation is limited and the devices may not always be connected to the internet, the algorithms for fault detection need to fit on the limited memory chips.

7. UAVs (Unmanned aerial vehicles) and Remote sensing: UAVs are often used to map a geographical area and to collect sensor readings where it is difficult for humans to be present and record data. In addition to collecting data, UAVs may also be tasked to execute some actions based on the observed data, such as extinguishing fires. Due to the remoteness of the tasks, the UAVs may not always communicate frequently, which leads to several agents having older versions of the models.

8. Online advertising: A search platform displays ads relevant to queries entered by a user by allowing advertisers to bid on each query. An auto-bidding agent is an automated algorithm that determines a bid (dollar value) depending on the relevance of the user query to the advertiser's choice of bidded keywords. Whenever an ad is clicked by a user, the advertiser pays some amount to the search platform that is determined by the auction mechanism. The goal of the agent is to maximize the number of ad clicks on a given day where the spending is constrained by a fixed daily budget. The agent must therefore balance between aggressively bidding for the current search query and saving

budget for future search queries. The agent must perform this optimization in the order of milliseconds since the users expect the search results to be displayed almost instantly.

9. Nanosatellites: Consider the example of deep space probes and nanosatellites that are extremely limited in their sensors and on-device power. They are required to collect useful data or execute actions within these constraints.

10. Robot vacuum cleaners and food delivery robots: It is very interesting to watch a food delivery robot trying to navigate through a busy traffic intersection. This use case is very similar to the autonomous vehicle example, however, the impact of a crash is less severe due to the low speeds at which these robots move. The computing power is much lesser in this case, however, the latency is not as much of an issue.

11. Industrial robots: Industrial tasks are increasingly automated to increase the pace of manufacturing and also since it is a cheaper form of labor. Some examples of the tasks are identifying objects, picking and placing them in their respective piles, etc. The algorithms are often trained on simulation environments that allow the collection of large amounts of data which is otherwise not possible using the actual robots. This leads to a shift between the data distribution (simulated) used at training and the data distribution observed during deployment. This shift affects the performance of the algorithms.

Please note that each of these examples can involve either training at the edge, inference at the edge, or both depending on the application. For example, mobile applications (auto-correct, keystroke detection, photo organizer, health diagnostics)

may need to collaboratively train with other users since each user's data is very limited and therefore not sufficient to train a good model. For a new user using these applications, the app may default to performing inference using a pre-trained model until enough data is collected to personalize it to the particular user. For robots, UAVs, nanosatellites, and autonomous vehicles, the devices must be able to perform inference despite some sensor failures which requires training the algorithm to cater to the challenges that arise during inference.

In this thesis, we shall discuss the resource constraints from the lens of two aspects: (i) distributed training at the edge and (ii) inference at the edge. We consider the two broad categories of learning: supervised and reinforcement learning. The resource constraints affect both of them in many similar ways and some unique ways particular to the problem.

1.2 Distributed Supervised Learning

Traditional centralized machine learning assumes a central server/cloud where all the data is aggregated/pooled. The learning algorithm is trained on this centralized data and the trained model is communicated to the edge devices/user devices for inference/deployment. This paradigm does not ensure the privacy of the user data and has other drawbacks such as high communication costs. The alternative is to train the models at the edge devices [13, 14] through a central server that coordinates training. Sometimes in the absence of a central server, fully decentralized learning methods are also used where the users communicate with each other directly [15]. A usual requirement in these algorithms is that every user is available for aggregation at every time step. This is not true in practice, since the compute times of each client

vary and the connectivity of the clients also varies. This need for synchronization offsets the speedup offered by having multiple users in the setup as it leads to stale users or asynchronous updates [16] that lead to poor convergence.

Distributed optimization has been studied for a long duration with applications to control systems, wireless sensor systems, network routing, communication systems, etc., [17, 18]. Distributed learning in the presence of practical challenges like non-iid data at the users, asynchronous updates/client unavailability [19, 20, 21, 22] are studied under *Federated Learning* (FL).

1.2.1 Background of Federated Learning

In the FL setup, each client has $D_i := \{x_j, y_j\}_{j=1}^{N_i}$, such that $x_j \sim \mathcal{D}_i,\ y_j = c(x_j)$ where c is a ground truth labeling function. Let $p_i \in [0, 1]$ such that, $N_i = p_i N$, where $N = \sum_{i=1}^{M} N_i$. The model space is denoted by $\mathcal{W}$ and the function $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$ is used to make the prediction.

We denote $\mathbf{w}_{(i,t)}$ as the weights at client i at time t. We simply write $\mathbf{w}_i$ when time is clear from the context and $\mathbf{w}$ when worker and time are clear from the context. Given a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we overload the loss function definition $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for each worker as $l(\mathbf{w}_i, x, y) = l\big(f(\mathbf{w}_i, x), y\big)$ and when we do not need to mention x, y, we simply use $\xi = (x, y)$ as $l(\mathbf{w}_i, x, y) = l(\mathbf{w}_i, \xi)$. Given a finite dataset $D_i = \{(x_j, y_j)\}_{j=1}^{N_i}$, we define the population loss L_i and the empirical loss $\widehat{L}_i$ as $L_i(\mathbf{w}_i) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\big[l(\mathbf{w}_i, x, y)\big]; \widehat{L}_i(\mathbf{w}, D_i) = \frac{1}{N_i}\sum_{j=1}^{N_i} l\Big(\mathbf{w}, x_j, y_j\Big)$. The objective of FL is to learn $\mathbf{w}^*$ such that

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}} \sum_{i=1}^{M} \alpha_i L_i(\mathbf{w}_i). \tag{1.1}$$

where $\alpha_i \geq 0$, $\sum_{i=1}^{M} \alpha_i = 1$. Popular choices for α_i are $\frac{1}{M}$ or p_i. Some popularly used algorithms are Federated Averaging (FedAvg) [19], and FedProx [23].

1.2.2 Challenges due to Resource Constraints

When learning at the edge, each user may have a different setup that is used for data collection, learning, communication, and inference. This variation in these properties leads to a mismatch when the user needs to participate in a distributed or federated setup. Most commonly studied challenges are due to mismatches in the data distribution of the edge devices, infrequent communication updates to the server, user scheduling, insufficient computing power at the users, compression of models, heterogeneous hardware, security and privacy [12, 24, 25]. These constraints broadly fall into the four types of constraints we discussed in Section 1.1.

In this work, we particularly focus on the challenges that arise due to model heterogeneity, i.e., each user model lies in a different space. Federated Averaging algorithms require some aggregation of the parameters of the users, but for this to work, all the users' parameters must lie in the same model space. This is a very important criterion for any of the previously proposed algorithms to work, however, realistic constraints make this condition often difficult to satisfy. For example, some users have very little computation power and can afford only a smaller model as compared to powerful users with powerful hardware. These situations are examples of model heterogeneity that make existing algorithms futile.

In addition to this, distributed learning setups also suffer from attacks on the learning system which can derail the training process. For example, a user can send arbitrarily malicious data instead of sending the actually computed parameters. In

such cases, the aggregation of the users' parameters can be offset from the actual value of the parameters without attacks and thus lead to slow convergence or divergence of the model. In this thesis, we shall consider situations where such attacks are possible when especially an estimate of the number of adversaries is not known in the learning system.

1.3 Reinforcement Learning

1.3.1 Background of Reinforcement Learning

Let us define a Markov Decision Process (MDP) by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the state space (which can be a finite state space or a general state space), $\mathcal{A}$ be the action space (finite or general). Let η_{init} be the distribution of the starting state. At a state $s \in \mathcal{S}$, the set of feasible actions is given by $\Gamma(s) \subseteq \mathcal{A}$. We use $\mathcal{B}$ to denote the feasible state-action pairs: $\mathcal{B} = \{(s, a) \in \mathcal{S} \times \mathcal{A} \mid a \in \Gamma(s)\}$. The reward function is denoted by $R : \mathcal{B} \to [0, R_{\max}]$ (most practical applications have a bounded reward). The transition kernel of the MDP which determines the state dynamics is denoted by $P : \mathcal{B} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the set of all probability distributions over $\mathcal{S}$. We use $\gamma \in (0, 1]$ to denote the discount factor.

Let $V^\pi : \mathcal{S} \to \mathbb{R}$ denote the value function defined by

$$V^\pi(s) = \mathbb{E}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R(s_h, a_h) \Big| s_1 = s, a_h \sim \pi(\cdot|s_h)\right]$$

The goal is to learn a stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes $v^\pi := \mathbb{E}_{s\sim\eta_{\text{init}}}[V^\pi(s)]$. Let $Q^\pi : \mathcal{B} \to \mathbb{R}$ denote the state-action value function (also called Q function) as

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R(s_h, a_h) | s_1 = s, a_1 = a, a_h \sim \pi(\cdot|s_h), h \geq 2\right]$$

The objective can also be defined as,

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{(s,a)}[Q^\pi(s,a)]$$

where Π is the set of feasible stationary policies.

1.3.2 Challenges due to Resource Constraints

The resource constraints that we consider for RL in this thesis are slightly different from those that we considered for FL. In particular, we focus on the inference problem, where the goal is to deploy one agent at the edge which faces resource constraints.

In the RL setting, the data takes the form of interactions between the agent and the environment ($\{(s_t, a_t, r_t, s_{t+1})\}_t$). A key element to the success of Reinforcement Learning in the game-playing framework is the availability of high-fidelity simulators that can mimic the transition dynamics of the RL environments. With access to such simulators, the agents can interact millions of times with the environment during training. This is often not practical for other real-world tasks. Even for the case of robotic tasks where building such simulators is actively pursued, it is often difficult to transfer the learned policies to the real world (also called Sim2Real problem) [26, 27].

On-policy algorithms require continual interaction between the agent and the environment and require fresh samples at each iteration. One such example is SARSA (or semi-gradient SARSA in the function approximation case). The convergence properties of on-policy methods require that the data comes from the on-policy distribution.

Off-policy algorithms on the other hand do not need fresh samples at every iteration as is required by the on-policy methods. Instead, off-policy algorithms learn from past interactions between the agent and the environment. Deep Q Learning is an

example of an off-policy method where a memory replay buffer is stored that collects interactions between the agent and the environment on a rolling basis. The algorithm samples interactions from the replay buffer and uses them in the update to the Q function parameters. The convergence of off-policy algorithms is not as robust as the on-policy algorithms, especially in the case of function approximation. While on-policy RL offers robust convergence guarantees, several applications of RL to real-world settings require an off-policy approach. It is impractical to collect the data in the on-policy setting due to the lack of a simulator and safety concerns with deploying a non-optimal policy in real-world operations. To collect data without access to a simulator, it is possible to deploy an arbitrary policy in real-time operation and collect data from the interactions. While deploying a random policy is possible, it may not be desirable primarily due to safety concerns, and it is a waste of resources to deploy such policies due to the cost of procuring data (device failure). Off-policy learning also requires collecting on-policy data that goes into the replay buffer. Data collected from off-policy learning often comprises trajectories of samples from a mixture of policies encountered during the training process.

Contrarily, there are scenarios where it is not possible to get more training data, and one only has access to a fixed dataset of interactions with the environment using some fixed behavior policy. This is often referred to as **Batch RL or offline RL** [28]. An important problem studied in the offline setting is off-policy evaluation (which is also relevant in the off-policy setting).

There are several challenges with efficiently learning from offline data, and the most important one is the data distributional shift. This distributional shift occurs because the data is collected from a different fixed policy, whereas the updates are

made to the value function of a different policy. This issue is addressed by several techniques such as Importance Sampling (IS), policy constraints, and uncertainty estimation methods. It is possible that if the dataset coverage is not sufficient, then function approximation extrapolates the action values of unseen state-action pairs, thereby incorrectly overestimating the values of such state-action pairs. Recent works [29], [30] take a pessimistic approach to address this issue, whereas [31] show that by increasing the size of the dataset conservative approaches are not required. In this thesis, we consider the challenges faced by training an offline RL agent that needs to be deployed in a resource-constrained edge environment.

1.4 Organization of the thesis

The rest of the thesis is organized as follows. Please refer to Fig 1.3 for the focus area of each chapter.

The focus of Chapter 2 is model heterogeneous Federated Learning (FL) for classification where different clients have different model architectures. Unlike existing works on model heterogeneity, we neither require access to a public dataset nor do we impose constraints on the model architecture of clients. We ensure that the clients' model architectures and data are private. We propose a theoretically grounded algorithm **Fed**erated **C**onditional **M**oment **A**lignment (Fed-CMA) that aligns conditional distributions of each client in the feature space and achieves a consensus on the final layer classification weights. The proposed algorithm requires communicating only the mean embeddings of the feature space and the classification layer weights, resulting in very little communication overhead. Under mild assumptions, we derive convergence and generalization properties of the algorithm to understand

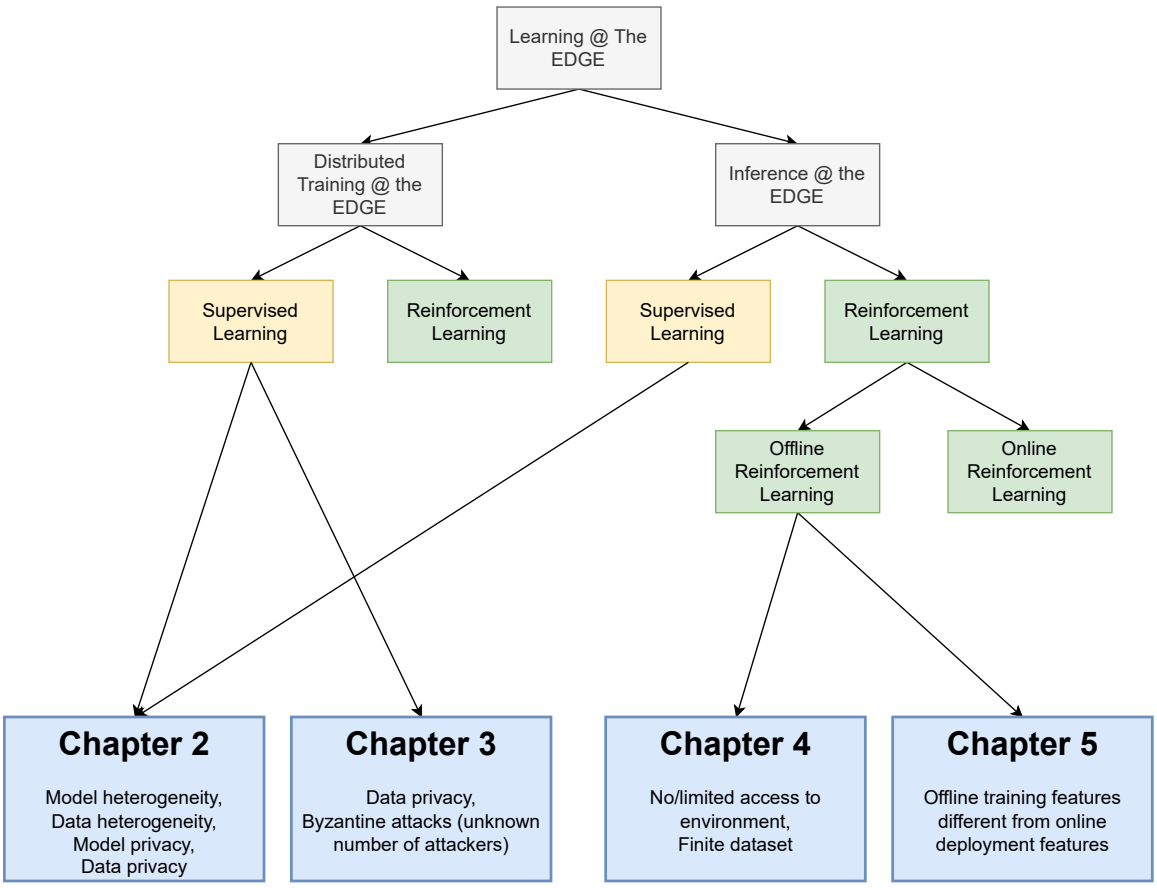


Figure 1.3: Flow chart describing the organization of the thesis

when participation in FL benefits the client. We provide fundamental insights into the role of the representations in the convergence and generalization of the algorithm. Empirical results on CIFAR-10, MNIST, EMNIST, FEMNIST, and synthetic datasets illustrate the efficacy of the proposed method.

In Chapter 3, the issue of security while learning at the edge is studied. Training distributed machine learning algorithms is prone to Byzantine attacks where the adversarial workers send corrupted model updates to derail the training. In this work, a reputation score-based gradient aggregation is proposed as a possible solution. We

introduce a class of novel stochastic gradient descent algorithms, ByGARS (Byzantine Gradient Aggregation using Reputation Scores) that involve computing reputation scores (of workers) using an auxiliary dataset at the server. These reputation scores are then used for aggregating the gradients (model updates) at the server. Under reasonable assumptions, we show that using these reputation scores is robust to any number of adversaries and prove the convergence of a representative algorithm, ByGARS++ for strongly convex objective functions using results from two-timescale stochastic approximation theory. The computational complexity of ByGARS++ is the same as the usual distributed stochastic gradient descent method with only an additional inner product computation in every iteration. We also demonstrate the effectiveness of the algorithms for non-convex learning problems using MNIST and CIFAR-10 datasets against almost all state-of-the-art Byzantine attacks.

In Chapter 4, we introduce the offline reinforcement learning problem and study the finite sample complexity bounds for offline reinforcement learning with general state, general function space, and state-dependent action sets. The algorithm analyzed does not require knowledge of the data-collection policy as compared to earlier works. We show that one can compute an ϵ-optimal Q function (state-action value function) using $O(1/\epsilon^4)$ i.i.d. samples of state-action-reward-next state tuples.

In Chapter 5, Offline reinforcement learning is used to train policies in scenarios where real-time access to the environment is expensive or impossible. As a natural consequence of these harsh conditions, an agent may lack the resources to fully observe the online environment before taking an action. This leads to situations where the offline dataset (available for training) can contain fully processed features (using powerful language models, image models, complex sensors, etc.) that are not available

when actions are actually taken online. In this resource-constrained setting, this disconnect leads to an interesting and unexplored problem in offline RL: Is it possible to use a richly processed offline dataset to train a policy that has access to fewer features in the online environment? In other words, this problem is an instantiation of the challenges described with inference at the resource-constrained edge. In this work, we introduce and formalize this novel resource-constrained problem setting. We highlight the performance gap between policies trained using the full offline dataset and policies trained using limited features. We advocate the use of transfer learning to address this performance gap by training a teacher agent using the offline dataset where features are fully available, and simultaneously distilling the knowledge to a student agent that only uses the resource-constrained features through a transfer loss. We evaluate the proposed approach on three diverse set of tasks for the MuJoCo suite (continuous control). Our analysis shows the proposed approach improves over the considered baselines and unlocks interesting insights.

Chapter 2: Model Heterogeneous Federated Learning

2.1 Introduction

In the paradigm of federated learning (FL), it is quite often that the client models have different architectures due to the heterogeneity of computational hardware devices (such as GPU memory, and smartphones). In such cases, naive parameter aggregating as in federated averaging (FedAvg) [19] or FedProx [23] might be no longer possible for achieving satisfactory generalization performance as in the homogeneous case. Even though some knowledge distillation approaches use a public dataset to address this issue, this type of dataset may not always be available and the role of the public dataset on the generalization performance is not yet understood well. Also, which type of architecture is used in deep learning has a huge impact on performance. The clients may consider the architecture as a trade secret (often due to the number of resources spent in designing it) and may not be willing to share the model architecture because of intellectual property concerns. We term this as *model privacy*. This also implies that the users are free to choose any type of (agnostic) model architecture without sharing it.

In this work, we consider the data and model heterogeneous FL scenario with facing the following possible additional constraints: (i) **model privacy**, (ii) **no public**

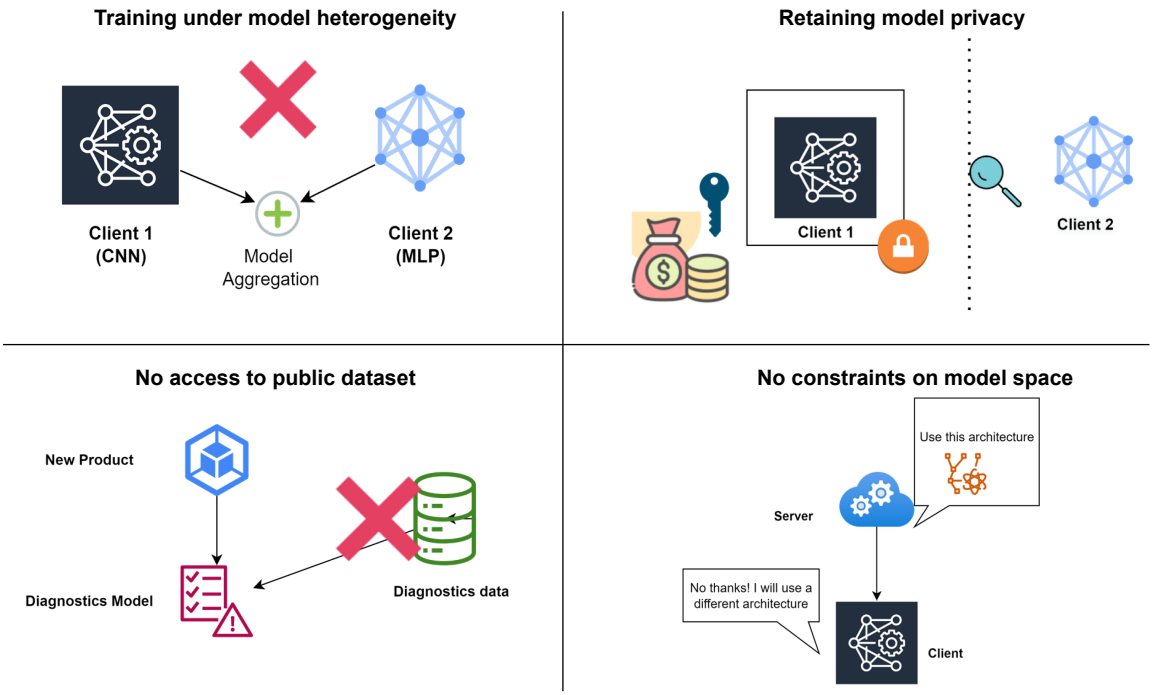


Figure 2.1: Constraints considered in this work

dataset, and (iii) **unrestricted model space**. See Fig 2.1 for an illustration of the constraints and see Table 2.1 for a summary of relevant works. Note that, while model heterogeneity refers to a more general case where the users can have different model spaces such as random forests, decision trees, and neural networks, we restrict our study to the use of neural networks (since the only way to learn in the general case is with distillation techniques which require a public dataset). Below, we provide a motivating example for this work.

(Motivating use case) Consider the following application where a product manufacturer works with multiple original equipment manufacturers (OEMs) from different countries (due to regulatory restrictions) for developing a new product line. The product manufacturer intends to improve the diagnostics and prognostics which is

often a supervised classification task. Since it is a new product line, data is scarce at every OEM (no public dataset), thus requiring the OEMs to collaborate. The OEMs may differ in the type of sensors leading to *data heterogeneity* and cannot share the data due to the *data privacy* constraint. They may also have the *model heterogeneity* issue due to various distributed computing resources that restrict the type of model architectures they would deploy. Moreover, in such scenarios, the OEMs may not be willing to share the model architecture as they may consider it as a trade secret and due to the number of resources spent in designing it manually or using techniques such as Neural Architecture Search [32].

Work	Model Heterogeneity	Model Privacy	Public Dataset	Restricted Model Space	Convergence
[33]	✓	✓	✓	×	×
[34] [35]	✓	×	✓	×	×
[36]	✓	×	×	✓	×
[37] [38]	✓	×	×	✓	✓
[39]	✓	✓	×	✓	×
[40]	✓	✓	×	×	×
Ours	✓	✓	×	×	✓

Table 2.1: Summary of previous work in model heterogeneous FL. The desirable properties are highlighted in green.

In this work, we propose a fundamentally different approach than the works in Table 2.1 to address model heterogeneity with a new algorithm that is motivated by theoretical results in the *domain adaptation* (DA) literature. In our case since the

client models are heterogeneous, we separate the client model architecture as a feature extractor that projects the input data into a latent space (common for all clients) and a classifier that acts on the latent space. We summarize our contributions as follows:

- A thorough theoretical generalization analysis is provided for model homogeneous and model heterogeneous FL highlighting the difficulty in the latter case. To the best of our knowledge, this is the first theoretical generalization error bound for the model heterogeneous FL setup, which justifies *the advantage of participating in FL* and *the importance of aligning the latent space distributions across the clients*.

- The proposed simple algorithm (Fed-CMA) can align the latent space conditional distributions and the classification weights across all clients in a federated way with convergence guarantees for finding the first-order stationary points to general non-convex problems.

- Multiple detailed numerical experiments are performed under different FL settings over both synthetic and real datasets. It can be observed that Fed-CMA outperforms the considered baselines achieves reduced communication complexity as compared to other model heterogeneous FL algorithms.

2.2 Preliminaries

2.2.1 Model Heterogeneous Federated Learning

Consider a K-class classification task, $\mathcal{T} := \langle \mathcal{D}, c \rangle$, where $\mathcal{D}$ is the distribution on $\mathcal{X} \subset \mathbb{R}^d$ and c is the ground truth labeling function that maps $\mathcal{X}$ to $\mathcal{Y} := \{1, \cdots, K\}$. We consider the model heterogeneous FL problem with M workers where each worker

has an individual task $\mathcal{T}_i := \langle \mathcal{D}_i, c \rangle \ \forall i \in [M]$. Each client owns $D_i := \{x_j, y_j\}_j^{N_i}$, such that $x_j \sim \mathcal{D}_i,\ y_j = c(x_j)$. Let $p_i \in [0,1]$ such that, $N_i = p_i N$, where $N = \sum_{i=1}^{M} N_i$. Also, each client has a model $\mathbf{w}_i$ in a hypothesis class $\mathcal{W}_i \subset \mathbb{R}^{d_i}$ (where d_i is the dimensionality of the model space $\mathcal{W}_i$) and a function $f_i : \mathcal{W}_i \times \mathcal{X} \to \mathcal{Y}$ which is used to make a prediction on a given data point. We denote $\mathbf{w}_{(i,t)}$ as the weights at client i at time t. We simply write $\mathbf{w}_i$ when time is clear from the context and $\mathbf{w}$ when worker and time are clear from the context. Given a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we define the loss function $l_i : \mathcal{W}_i \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for each worker as $l_i(\mathbf{w}_i, x, y) = l\big(f_i(\mathbf{w}_i, x), y\big)$. Given a finite dataset $D_i = \{(x_j, y_j)\}_{j=1}^{N_i}$, we define the population loss (true loss) L_i and the empirical loss $\widehat{L}_i$ (the subscript i refers to $\mathcal{D}_i$ at the worker i) as $L_i(\mathbf{w}_i) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\big[l_i(\mathbf{w}_i, x, y)\big]$ and $\widehat{L}_i(\mathbf{w}, D_i) = \frac{1}{N_i}\sum_{j=1}^{N_i} l_i\Big(\mathbf{w}, x_j, y_j\Big)$. We simply use $\widehat{L}_i(\mathbf{w})$ when D_i is clear from the context. We use the terms "loss", error, and risk interchangeably. The objective of model heterogeneous FL is to simultaneously learn $\{\mathbf{w}_i^*\}_{i=1}^{M}$ such that

$$(\mathbf{w}_1^*, \mathbf{w}_2^*, \cdots, \mathbf{w}_M^*) = \arg\min_{\{\mathbf{w}_i\}_{i=1}^{M}} \sum_{i=1}^{M} \alpha_i L_i(\mathbf{w}_i), \tag{2.1}$$

where $\alpha_i \geq 0,\ \sum_{i=1}^{M} \alpha_i = 1$. Popular choices for α_i are $\frac{1}{M}$ or p_i.

The challenge in solving (2.1) is because $\{\mathbf{w}_i\}_{i=1}^{M}$ share different network architectures, meaning that the existing algorithms on parameter aggregation (such as FedAvg, FedProx) cannot be used. Under model heterogeneity, we only assume a common latent space ($\mathcal{Z} \subseteq \mathbb{R}^{d_e}$) for all workers. In other words, each worker i has a model $\mathbf{w}_i = (\mathbf{u}_i, \mathbf{v}_i) \in \mathcal{W}_i$ where $\mathbf{u}_i \in \mathcal{U}_i$, $\mathbf{v}_i \in \mathcal{V}$. We define the prediction function as $f_i(\mathbf{w}_i, x) := h\big(\mathbf{v}_i, g_i(\mathbf{u}_i, x)\big)$ where $g_i(\mathbf{u}_i, \cdot) : \mathcal{X} \to \mathcal{Z}$ projects $\mathcal{X}$ to the latent space and $h(\mathbf{v}_i, \cdot) : \mathcal{Z} \to \mathcal{Y}$ makes the prediction. In general, $\dim(\mathcal{U}_i) \gg \dim(\mathcal{V})$. Typically, consider a neural network where $\mathbf{u}_i$ corresponds to the weights of the feature extractor

and $\mathbf{v}_i$ corresponds to the weights of the classification layer (see Fig 2.2). For example, [41] and [42] consider a similar setup for personalizing the input representation learning layers while sharing the classification layer with the server.

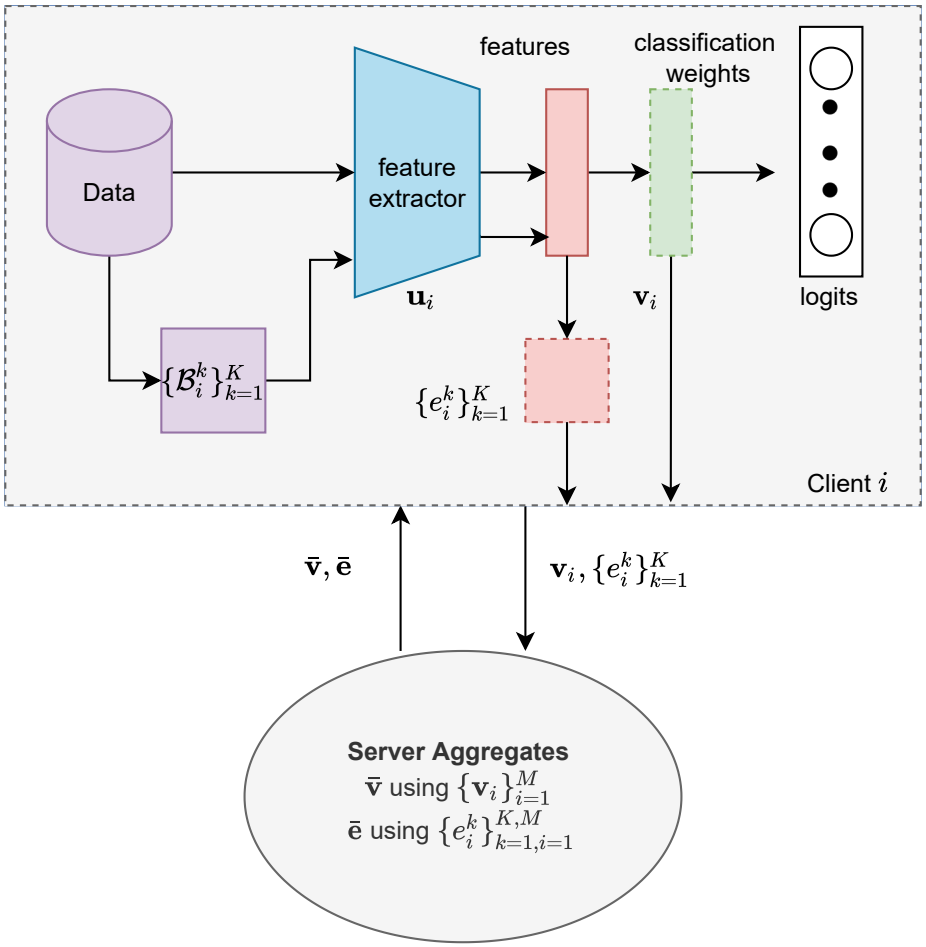


Figure 2.2: Illustration of Fed-CMA .

2.2.2 Domain Adaptation

In DA, a classifier is trained on a sufficiently large source dataset (task) such that it is expected to perform well on a target dataset (task) with few or no labeled

data points. For the purpose of understanding the DA perspective and studying the generalization, we shall consider a binary classification task. Let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} = [0, 1]$, which is construed as the probability of having the label 0. Let us denote the random variables in $\mathcal{X}, \mathcal{Y}$ as X and Y respectively. We use the terms "task", "domain", and "client" interchangeably from hereon. Consider that the clients have the same model class (or hypothesis) denoted by $\mathcal{W}$ and hence the same prediction function $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$. We define the disagreement between two models $\mathbf{w}, \mathbf{w}' \in \mathcal{W} \subseteq \{0, 1\}^{\mathcal{X}}$ as $L_i(\mathbf{w}, \mathbf{w}') = \mathbb{E}_{x \in \mathcal{D}_i}\big[|f(\mathbf{w}, x) - f(\mathbf{w}', x)|\big]$. Suppose that the target labeling functions $\{c_i : \mathcal{X} \to \mathcal{Y}\}$ are different for every client, then the error of a model $\mathbf{w} \in \mathcal{W}$ on a client i is $L_i(\mathbf{w}) := L_i(\mathbf{w}, c_i)$. The error of a model $\mathbf{w}$ on clients i, j can be related as [43]

$$L_j(\mathbf{w}) \leq L_i(\mathbf{w}) + d_{\mathcal{W}\Delta\mathcal{W}}(\mathcal{D}_i, \mathcal{D}_j) + \min\Big\{\mathbb{E}_{x \in \mathcal{D}_i}\big[|c_i(x) - c_j(x)|\big], \mathbb{E}_{x \in \mathcal{D}_j}\big[|c_i(x) - c_j(x)|\big]\Big\}, \tag{2.2}$$

where $d_{\mathcal{W}\Delta\mathcal{W}}$ is used to measure the divergence between the two distributions $\mathcal{D}_i, \mathcal{D}_j$ [44] (see the appendix for formal definitions). If the divergence is small, it is hard to discriminate between the two data distributions. This shows that, if the labeling functions c_i, c_j are close, and $d_{\mathcal{W}\Delta\mathcal{W}}$ is small, then a model trained on one client i (source) can perform well on another client j (target). Let $q_i(X, Y)$ denote the joint distribution of X, Y for client i. Suppose that the labeling functions $\{c_i\}$ are the same for all the clients, then $q_i(Y|X) = q_j(Y|X)$, but the marginals of X are not equal. This is called the *covariate shift* assumption [45] which we consider in this work. Under this assumption, the last term in (2.2) is zero, and only $d_{\mathcal{W}\Delta\mathcal{W}}$ needs to be small. However, given two tasks, the divergence between the distributions in $\mathcal{X}$ is fixed. This motivated the learning of *domain invariant representations* such

that the divergence of distributions in the representation space (or latent space) is minimized. Broadly, these methods take the following approaches. **(i) Minimizing an integral probability metric (IPM):** Along with the classification loss on source data, the network is explicitly trained to minimize an IPM such as maximum mean discrepancy (MMD [46]) [47, 48, 49] or a central moment discrepancy (CMD) [50] between the source and target datasets at the representation layer. **(ii) Adversarial methods:** [51, 52, 53] train the network by minimizing the label classification loss and maximizing a domain classification loss.

Let $\mathcal{Z} \subset \mathbb{R}^{d_e}$ be a latent space, $g(\mathbf{u}, \cdot) : \mathcal{X} \to \mathcal{Z}$ be a representation function where $\mathbf{u} \in \mathcal{U}$ and Z be a random variable in $\mathcal{Z}$. Given a task $\langle \mathcal{D}, c \rangle$, $\mathbf{u}$ induces the distribution $\tilde{\mathcal{D}}(\mathbf{u})$ on $\mathcal{Z}$ and the labeling function $\tilde{c}_{\mathbf{u}} : \mathcal{Z} \to \mathcal{Y}$ such that $\mathbb{E}_{z \in \tilde{\mathcal{D}}(\mathbf{u})}\big[\mathbb{I}_B(z)\big] = \mathbb{E}_{x \in \mathcal{D}}\big[\mathbb{I}_B(g(\mathbf{u}, x))\big]$, where $B \subset \mathcal{Z}$ is a measurable set, $\mathbb{I}_B$ is an indicator function over B, and $\tilde{c}_{\mathbf{u}}(z) := \mathbb{E}_{x \in \mathcal{D}}\big[c(x) | g(\mathbf{u}, x) = z\big]$, $\forall z \in \mathcal{Z}$. Let $q_i(Z, Y)$ be the joint distribution at client i using a representation $\mathbf{u}_i$. Under this setup, the induced labeling functions in the latent space may not be equal even when the labeling functions in the input space are the same [43] and several works followed to address this shortcoming by assuming a *generalized label shift* condition [54, 55] where $q_i(Z|Y) = q_j(Z|Y)$. This motivates our conditional alignment technique to solve (2.1). In contrast with DA (even the multi-source) setting, we have the following challenges: **(i) restriction on data sharing and model sharing, (ii) learning simultaneously for all clients**. In classic multi-source DA, the goal is to train a classifier on all sources so that it can generalize well on the target task. In our Federated setting, every client simultaneously solves a multi-source DA problem viewing itself as the target. On the other hand,

each client has a labeled dataset unlike in unsupervised DA, where the target has no labeled data and one might need to estimate label distribution ratios.

2.3 Generalization Result for Model Heterogeneous FL

Next, we will first discuss the existing results for generalization in FL and highlight the limitations of these results. Let us consider the case where all the client models are homogeneous, and the client weights can be shared. The weighted data distribution is denoted as $\mathcal{D}_{\boldsymbol{\alpha}} = \sum_{i=1}^{M} \alpha_i \mathcal{D}_i$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_M)$ be the domain weights for each user such that $\sum_{j=1}^{M} \alpha_i = 1$, and let j be the target domain (which can be one of the clients). There are two types of existing generalization results for FL in the literature where the final prediction (i) uses a weighted average of the client model weights i.e., $f\big(\sum_i \alpha_i \mathbf{w}_i, x\big)$ [56], (ii) uses an ensemble of the predictions on the client models, i.e., $\sum_i \alpha_i f_i(\mathbf{w}_i, x)$ [34, 35].

Averaged model weights Following Section 2.2.1, the empirical loss is defined as $\widehat{L}_i(\mathbf{w}_i) := \frac{1}{N_i} \sum_{(x,y)\in D_i} |f(\mathbf{w}_i, x) - y|$ and the $\boldsymbol{\alpha}$ empirical weighted loss as $\widehat{L}_{\boldsymbol{\alpha}}(\mathbf{w}) = \sum_{i=1}^{M} \alpha_i \widehat{L}_i(\mathbf{w})$. The true weighted loss $L_{\boldsymbol{\alpha}}$ can be defined similarly by following Section 2.2.1. Theorem 2 of [56] shows a generalization bound for FL (when j is the target): with probability at least $1 - \delta$,

$$L_j(\mathbf{w}_j) \leq \widehat{L}_{\boldsymbol{\alpha}}\Big(\sum_{i=1}^{M} \alpha_i \mathbf{w}_i\Big) + \sum_{i=1}^{M} \alpha_i \Big(\frac{1}{2} d_{\mathcal{W}\Delta\mathcal{W}}(\mathcal{D}_i, \mathcal{D}_j)\Big) + \mathcal{O}\Big(\frac{\log \frac{1}{\delta}}{\sqrt{N}}\Big). \tag{2.3}$$

Here, $\mathcal{O}(\cdot)$ hides logarithmic terms and model class complexities that are constant. For the same model class, if the target domain j only uses the data available locally, from standard results in generalization, we know that the user's generalization error can be bounded as $L_j(\mathbf{w}_j) \leq \widehat{L}_j(\mathbf{w}_j) + \mathcal{O}\Big(\frac{\log 1/\delta}{\sqrt{N_j}}\Big)$ with probability at least $1 - \delta$. The

benefit of joining FL can be seen from the $\frac{1}{\sqrt{N}}$ term in (2.3) since $N \gg N_j$ when M is large. The $\widehat{L}_{\boldsymbol{\alpha}}(\sum_i \alpha_i \mathbf{w}_i)$ (empirical loss at all users) and $d_{\mathcal{W}\Delta\mathcal{W}}$ (captures heterogeneity among users) terms must be smaller than $\widehat{L}_j(\mathbf{w}_j)$ for this benefit to be realizable. This encourages the FL objective to solve $\arg\min \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{w})$ by sharing the model weights.

However, this result has a major drawback. It only holds for model homogeneous FL, since the result is with respect to the weighted model $\sum_{i=1} \alpha_i \mathbf{w}_i$, which is clearly not possible in model heterogeneous (or model private) FL.

Ensembled model predictions We slightly abuse the notation for this discussion by defining the loss as follows: $\widehat{L}_i(f_i) := \frac{1}{N_i} \sum_{(x,y)\in D_i} |f_i(\mathbf{w}_i, x) - y|$. The generalization result where the target distribution is $\mathcal{D}_j$ can be summarized as follows (although there are subtle variations in Theorem 5.1 in [34] and Theorem 1 in [57]): with probability at least $1 - \delta$,

$$L_j\Big(\sum_{i=1}^{M} \alpha_i f_i\Big) \leq \sum_{i=1}^{M} \alpha_i \widehat{L}_i(f_i) + \sum_{i=1}^{M} \frac{\alpha_i}{2} d_{\mathcal{W}\Delta\mathcal{W}}(\mathcal{D}_i, \mathcal{D}_j) + \sum_{i=1}^{M} \mathcal{O}\Big(\frac{\alpha_i \log \frac{1}{\delta}}{\sqrt{N_i}}\Big). \tag{2.4}$$

Although this result holds for the model heterogeneous case, the result fails to capture the advantage of performing FL for the following reasons. Firstly, the bound contains the term $\sum_{i=1}^{M} \mathcal{O}\Big(\frac{1}{\sqrt{N_i}}\Big)$ which is worse than $\mathcal{O}\Big(\frac{1}{\sqrt{N_j}}\Big)$, while also containing the divergence terms $d_{\mathcal{W}\Delta\mathcal{W}}$ and the local loss terms $\sum_{i\neq j} \alpha_i \widehat{L}_i(f_i)$. Therefore, the result does not show any improvement over local training. The result also holds when the models are trained locally and the ensemble is used only during prediction. This does not motivate the distillation loss (with communication during training) in [34, 35]. Even more importantly, the impact of the public dataset distribution (used for distillation) is not captured in the result. Therefore, this bound is uninformative and fails to justify the benefit of FL.

Also note that the original results in [56, 34, 35] include an additional term λ which quantifies the best oracle loss. However, we observe that this λ term can be avoided and thus we omitted it from the generalization results in (2.3) and (2.4).

We shall now address the limitations mentioned above by (i) decoupling the representation layer from the classification layer which allows us to study model heterogeneity in FL and (ii) using results from multi-source domain adaptation which allows us to show $\frac{1}{\sqrt{N}}$ dependence without assuming a centralized dataset. While Theorem 1 of [40] follows (i), they still show a $\frac{1}{\sqrt{N_i}}$ dependence and only consider the model homogeneous case.

2.3.1 Model Homogeneous FL

Let us first show the result for the model homogeneous case, where averaging the model weights is optimal. Given a representation function $\mathbf{u}$, denote the induced distribution of the i^{th} client as $\tilde{\mathcal{D}}_i(\mathbf{u})$ and the induced labeling function as $\tilde{c}_{\mathbf{u}}$. The empirical risk of the i^{th} client for the model $(\mathbf{u}, \mathbf{v})$ is defined as $\widehat{L}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}}) := \frac{1}{N_i} \sum_{x \in D_i} |h(\mathbf{v}, g(\mathbf{u}, x)) - \tilde{c}_{\mathbf{u}}(g(\mathbf{u}, x))|$ and the true risk $L_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ is the expectation of $\widehat{L}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ taken with respect to the draw of D_i. The domain-weighted empirical risk is defined similarly as $\widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}) = \sum_{i=1}^{M} \alpha_i \widehat{L}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ and the true weighted risk is defined similarly as $L_{\boldsymbol{\alpha}}(\mathbf{v})$. We denote the mixture of the induced client distributions defined as $\tilde{\mathcal{D}}_{\boldsymbol{\alpha}} = \sum_{i=1}^{M} \alpha_i \tilde{\mathcal{D}}_i(\mathbf{u})$. Let the target distribution be given by $\mathcal{D}_j$ and the induced distribution is $\tilde{\mathcal{D}}_j(\mathbf{u})$. We now state the generalization bound.

Theorem 1 (Model Homogeneity)**.** *Given* $\mathbf{u}$*, let* $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v})$ *and* $\mathbf{v}_j^* = \arg\min_{\mathbf{v}\in\mathcal{V}} L_{j,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$*. Then for any* $\delta > 0$*, w.p.* $> 1-\delta$*, we have*

$$L_{j,\mathbf{u}}(\widehat{\mathbf{v}}) - L_{j,\mathbf{u}}(\mathbf{v}_j^*) \leq 4\sqrt{\sum_{i=1}^{M} \frac{\alpha_i^2}{p_i}}\mathcal{O}\Big(\sqrt{\frac{\log\frac{1}{\delta}}{2N}}\Big) + B,$$

where $B = \sum_{i=1}^{M} \alpha_i d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_i(\mathbf{u}), \tilde{\mathcal{D}}_j(\mathbf{u})\big)$.

Let $\alpha_i = p_i$, then the generalization error decays as $\mathcal{O}\big(\sqrt{1/N}\big)$. Therefore, the dependence on the number of data points shows that the client attains better generalization error by participating in the FL setup provided the divergence term is small. Note that, as compared to (2.3), we have divergence between the induced distributions $d_{\mathcal{V}\Delta\mathcal{V}}(\tilde{\mathcal{D}}_i(\mathbf{u}), \tilde{\mathcal{D}}_j(\mathbf{u}))$ as compared to $d_{\mathcal{W}\Delta\mathcal{W}}(\mathcal{D}_i, \mathcal{D}_j)$ which is fixed for the given client datasets. The divergence term in Theorem 1 can be minimized by learning invariant feature representations $\mathbf{u}$ which is the topic of interest in domain invariant representation learning [49, 43].

The following result provides a way to empirically estimate the $d_{\mathcal{V}\Delta\mathcal{V}}$ divergence based on the data samples.

Lemma 1 ([44],[58])**.** *Let* $\mathcal{V}$ *be a hypothesis space on* $\mathcal{Z}$ *with VC dimension* $d_{\mathcal{V}}$*. If* $\mathcal{U}$ *and* $\mathcal{U}'$ *are samples of size* n *from* $\mathcal{D}$ *and* $\mathcal{D}'$ *respectively. Then, for any* $\delta \in (0,1)$ *with probability at least* $1-\delta$*,*

$$d_{\mathcal{V}\Delta\mathcal{V}}(\mathcal{D}, \mathcal{D}') \leq 4\sqrt{\frac{d\log(2n) + \log(\frac{2}{\delta})}{n}} + 2\Big(1 - \min_{\mathbf{v}\in\mathcal{V}}\Big\{\frac{1}{n}\sum_{z:h(\mathbf{v},z)=0} \mathbb{I}[z\in\mathcal{U}] + \frac{1}{n}\sum_{z:h(\mathbf{v},z)=1} \mathbb{I}[z\in\mathcal{U}']\Big\}\Big).$$

We train a classifier to identify the distribution to which the data points belong. The divergence term is large when the learned function is able to classify properly between the distributions, and the divergence term is small when we are unable to learn a hypothesis that can distinguish between the distributions.

Intuition behind controlling $d_{\mathcal{V}\Delta\mathcal{V}}$: We now present a simple example to illustrate when $d_{\mathcal{V}\Delta\mathcal{V}}$ is small. Let us consider two clients whose datasets are drawn from two homoscedastic Gaussian distributions $\mathcal{N}_i(\mu_i, \Sigma), i \in \{1, 2\}$ in $\mathbb{R}^d$. Let us consider a 1-hidden layer network with identity activation functions. The hidden layer dimension is d_h and the output dimension is 1 and the weights of the two layers are given by $W_1 \in \mathbb{R}^{d_h \times d},\ W_2 \in \mathbb{R}^{1 \times d_h}$). For simplicity, consider a model homogeneous case, where the weights (W_1, W_2) are shared between the two clients. The projections of the two datasets in the hidden space are also Gaussian since we are just doing an affine transformation, i.e., the hidden layer projections are drawn from $\mathcal{N}_i(W_1\mu_i, W_1\Sigma W_1^T), i \in \{1, 2\}$. The output of the network is the projection of the hidden layer onto a single dimension. We know that the Bayes error for 2 homoscedastic classes with equal priors is given by $2\Phi\left(\frac{-|W_2W_1\mu_1 - W_2W_1\mu_2|}{2\sqrt{W_2W_1\Sigma W_1^T W_2}}\right)$, where Φ is the cdf of the standard normal distribution. Since no classifier can do better than the Bayes error, we have

$$\min_{\mathbf{v}\in\mathcal{V}} \left[\frac{1}{n}\sum_{z:h(\mathbf{v},z)=0} I[z \in \mathcal{U}] + \frac{1}{n}\sum_{z:h(\mathbf{v},z)=1} I[z \in \mathcal{U}']\right] \geq 2\Phi\left(\frac{-|W_2W_1\mu_1 - W_2W_1\mu_2|}{2\sqrt{W_2W_1\Sigma W_1^T W_2}}\right).$$

Thus, when the distance between the means in the latent space, $W_1\mu_1$ and $W_1\mu_2$, is high, then the Bayes error $2\Phi\left(\frac{-|W_2W_1\mu_1 - W_2W_1\mu_2|}{2\sqrt{W_2W_1\Sigma W_1^T W_2}}\right)$ is low. As a result, to keep the divergence $d_{\mathcal{V}\Delta\mathcal{V}}$ small, the latent space means must be closer, so that $2\Phi\left(\frac{-|W_2W_1\mu_1 - W_2W_1\mu_2|}{2\sqrt{W_2W_1\Sigma W_1^T W_2}}\right)$ is large.

2.3.2 Model Heterogeneous FL

It is important to understand that due to model heterogeneity, we can not show a bound similar to (2.3) since the model weights of different clients can not be averaged. Since in our setup, only the classification layer weights are shared, therefore we need to decouple the representation layers from the classification layers to show the

generalization bound. We use the same approach as Theorem 1 to consider the latent space distributions induced by the clients' private representation functions $\{\mathbf{u}_i\}_{i=1}^M$.

For the model heterogeneous case (since there is no global model), let $\mathbf{u}_i$ be the representation layer weights of the ith user. We then define the induced distributions $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$, labeling functions $\tilde{c}_{\mathbf{u}_i}$, empirical risk $\widehat{L}_{i,\mathbf{u}_i}(\mathbf{v}_i, \tilde{c}_{\mathbf{u}_i})$ with respect to $\mathbf{u}_i$ for every client. The weighted empirical risk is then $\widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}) = \sum_{i=1}^M \alpha_i \widehat{L}_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_i})$. A major difference is that the induced labeling functions $\tilde{c}_{\mathbf{u}_i}$ are different for every client whereas in Theorem 1, it is the same for all clients due to shared $\mathbf{u}$. Let the target distribution be given by $\mathcal{D}_j$ and the corresponding representation be $\mathbf{u}_j$ that induces $\tilde{\mathcal{D}}_j(\mathbf{u}_j)$ and $\tilde{c}_{\mathbf{u}_j}$. We now state the generalization bound as follows.

Theorem 2 (Model Heterogeneity)**.** *Let* $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(h)$ *and* $\mathbf{v}_j^* = \arg\min_{\mathbf{v}\in\mathcal{V}} L_{j,\mathbf{u}_j}(\mathbf{v}, \tilde{c}_{\mathbf{u}_j})$ *be the minimizer of the true target risk. Then for any* $\delta > 0$*, w.p.* $> 1-\delta$*, we have*

$$L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}}) - L_{j,\mathbf{u}_j}(\mathbf{v}_j^*) \leq 4\sqrt{\sum_{i=1}^M \frac{\alpha_i^2}{p_i}}\mathcal{O}\Big(\sqrt{\frac{\log\frac{1}{\delta}}{N}}\Big) + 2A + B,$$

where $A = \sum_{i=1}^M \alpha_i \min\Big\{\mathbb{E}_{z\in\tilde{\mathcal{D}}_i(\mathbf{u}_i)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big],\ \mathbb{E}_{z\in\tilde{\mathcal{D}}_j(\mathbf{u}_j)(z)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big]\Big\}$ *and* $B = \sum_{i=1}^M \alpha_i d_{\mathcal{V}\Delta\mathcal{V}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i), \tilde{\mathcal{D}}_j(\mathbf{u}_j))$.

The bound is similar to that of Theorem 1 with two key differences: (i) the addition of term A and (ii) the divergence in B is computed between the induced distributions on private representations $\mathbf{u}_i$ of the clients. These two differences are inherently due to the presence of model heterogeneity. The term A determines the closeness of $\tilde{c}_{\mathbf{u}_j}$ with every other $\tilde{c}_{\mathbf{u}_i}$. Observe that in the model homogeneous case where $\mathbf{u}$ is shared among the clients, $A = 0$ since all the induced labeling functions are equal to $\tilde{c}_{\mathbf{u}}$. Similarly, as compared to the model homogeneous case, B here is measured between the $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$ and

$\tilde{\mathcal{D}}_j(\mathbf{u}_j)$ for $\mathbf{u}_i \neq \mathbf{u}_j$ leading to a higher divergence. Due to these differences, achieving good generalization in model heterogeneous FL is more challenging than in model homogeneous FL. Observe that, in Theorem 1 and 2, the error measured is dependent on the representations since $\widehat{\mathbf{v}}, \mathbf{v}_j^*$ are dependent on the representations. Showing the result without fixing the representations is more challenging and we leave this as future work.

Role of the representations $\{\mathbf{u}_i\}$: The term B is non-zero when the data distributions are non-iid. If the given representations $\{\mathbf{u}_j\}$ are such that in the latent space $\mathcal{Z}$, the induced distribution of the target $\tilde{\mathcal{D}}_j(\mathbf{u}_j)$ is close to the induced $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$, then $d_{\mathcal{V}\Delta\mathcal{V}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i), \tilde{\mathcal{D}}_j(\mathbf{u}_j))$ is small. The A term, determines the closeness of $\tilde{c}_{\mathbf{u}_j}$ with every other $\tilde{c}_{\mathbf{u}_i}$. Observe that in the model homogeneous case where $\mathbf{u}$ is shared among the clients, $\lambda_{\boldsymbol{\alpha}} = 0$ since all the induced labeling functions are equal to $\tilde{c}_{\mathbf{u}}$. Even when the representations are shared and the $d_{\mathcal{V}\Delta\mathcal{V}}$ is small, this is not guaranteed to be small [43]. The goal of representation learning should therefore be to simultaneously reduce the divergence between the marginal distributions and the distance between the labeling functions. While the divergence can be minimized by minimizing an IPM (such as MMD), the distance between the induced labeling functions is difficult to measure. Therefore, in Fed-CMA , we propose to minimize the divergence between the class conditional latent space distributions to bring the induced labeling functions closer.

While FL research primarily focuses on improving the convergence rate by addressing issues such as variance reduction or client drift [59], the role of the feature extractors in reducing the degree of heterogeneity in the latent space receives rare attention. To the best of our knowledge, our work is the first to show generalization

bounds for FL, which provide these fundamental insights and highlight the benefits of participating in FL.

The generalization result can be further improved by adopting the *clustering structure implies sufficiency* result of [54] and the optimal transport bound of [55] which uses the Wasserstein metric to measure the divergence between the latent space distributions without requiring to measure the difference between the induced labeling functions.

2.4 Proposed Algorithm

2.4.1 Maximum Mean Discrepancy (MMD)

MMD [46] is a popular and powerful choice to compute the distance between two distributions. Define a feature map $\phi : \mathcal{X} \to \mathcal{H}_k$ where $\mathcal{H}_k$ is a reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}_k}$. The MMD between two distributions $\mathcal{D}, \mathcal{D}'$ is defined as $\text{MMD}(\mathcal{D}, \mathcal{D}') = \Big\| \mathbb{E}_{x \sim \mathcal{D}}[\phi(x)] - \mathbb{E}_{x' \sim \mathcal{D}'}[\phi(x')] \Big\|_{\mathcal{H}_k}$. The empirical estimate of the MMD given two datasets D, D' of size N each is given by $\text{MMD}(D, D') = \frac{1}{N} \sum_{x_i, x_j \in D} k(x_i, x_j) + \frac{1}{N} \sum_{x'_i, x'_j \in D'} k(x'_i, x'_j) - \frac{2}{N} \sum_{x_i \in D, x'_j \in D'} k(x_i, x'_j)$. A popular choice for the kernel k is a radial basis kernel or a gaussian kernel, $k(x, y) = \exp^{-\frac{\|x-y\|_2^2}{2\gamma}}$, where γ is the bandwidth parameter. For every client i with data $D_i = \{x_j, y_j\}_{j=1}^{N_i}$ and weights $(\mathbf{u}_i, \mathbf{v}_i)$, we define the dataset in the latent space induced by $\mathbf{u}_{(i,t)}$ at time t as $\tilde{D}_{(i,t)} = \{ \big(z_j, y_j\big) | z_j = g_i(\mathbf{u}_{(i,t)}, x_j) \; \forall (x_j, y_j) \in D_i \}$. Suppose, if clients can communicate the latent space datasets to the other clients, then the MMD distance can be used as a regularizer to align the distribution of a client's latent space representations to that of the other clients j. More precisely, at every local iteration t, client i minimizes $\widehat{L}_i(\mathbf{w}_{(i,t)}) + \sum_{j=1, j \neq i}^{M} \text{MMD}(\tilde{D}_{(i,t)}, \tilde{D}_{(j,\tau_j)})$

where τ_j is the last time step at which client j communicated the latent dataset to client i. This way, each client can train their representation layers $\mathbf{u}_i$ such that the latent space distributions are aligned with those of other clients. Furthermore, from the insights from the previous section, clients can compute MMD between the class conditional datasets of every client.

However, to practically use this algorithm, there are some challenges. The accuracy of empirical MMD improves with the number of data samples, but the computation complexity of MMD is quadratic in the number of data points (i.e., $\mathcal{O}(N^2)$ if N is the number of data points used in MMD). This is a huge computational overhead considering that the clients may be resource constrained, not to mention the difficulty in tuning the bandwidth parameter for each MMD computation (especially when class conditional alignment is used). It was also shown that MMD is more effective when using multiple kernels [49] which further increases the complexity. Moreover, there maybe privacy issues with sharing the latent space datasets. While techniques like differentially private dataset release [60] can be employed, it further increases the computation complexity at the clients, which is not desirable. In the next section, we propose an algorithm based on matching just the first order moments of the latent space datasets to assist in aligning the latent space conditional distributions.

2.4.2 FedCMA

We recall the example from [43] (Fig 1) which shows that marginal alignment of distributions is insufficient when the labeling functions are different. The goal of representation learning should therefore be to simultaneously reduce the terms A and B. Towards this, we propose conditional alignment of distributions in the latent

space which aims to learn representation weights $\{\mathbf{u}_i\}$ such that the induced labeling functions are closer and the latent space distributions are closer (see Fig 2.3 for an illustration).

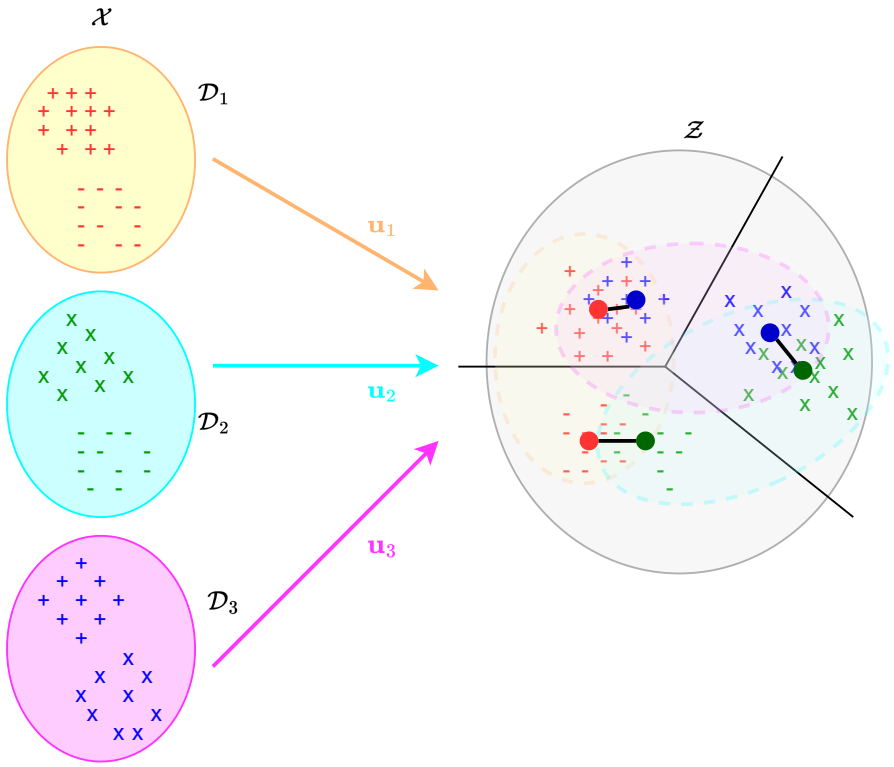


Figure 2.3: Illustration of conditional alignment

As compared to simultaneously minimizing the MMD between all pairs of clients, we propose a simple algorithm that maintains a global variable that tracks the weighted average of the empirical conditional mean embeddings of all the clients and we minimize the distance of each client's empirical mean embeddings with the global variable.

Let p_i^k denote the fraction of data points of label k available at client i out of all data points of label k at all clients. At the beginning of the training, each client i samples a fixed i.i.d minibatch $\mathcal{B}_i^k \sim \mathcal{D}_i$ for each kth label in the dataset D_i with b_i^k points. The size of b_i^k can be much smaller than the size of the original dataset N_i. This minibatch is used throughout the training. The class conditional mean embedding vector $e_i^k \in \mathbb{R}^{d^e}$ is computed as $e_i^k = \frac{1}{b_i^k} \sum_{x \in \mathcal{B}_i^k} g_i(\mathbf{u}_i, x)$. We abuse the notation slightly, by using $e_i^k(\mathbf{u})$ when the parameter $\mathbf{u}$ is of interest. These vectors are the class conditional empirical means of each class available at each client.

Further, principled approaches such as adding calculated noise to the mean embeddings can be performed to achieve rigorous differential privacy guarantees; however this is outside the scope of this work. We maintain a global variable $\mathbf{e} = \{e^k\}_{k=1}^K$ that stores a weighted average of the means of all the clients. This global variable is used to achieve consensus among the clients and align the client distributions in the latent space. Similarly, we maintain a global variable $\bar{\mathbf{v}}$ for the classification layer weights. The population local loss at worker i is

$$\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) = L_i(\mathbf{w}_i) + \frac{\lambda_1}{2} p_i \left\| \mathbf{v}_i - \bar{\mathbf{v}} \right\|_2^2 + \frac{\lambda_2}{2} \sum_{k=1}^{K} \frac{p_i^k}{b_i^k} \sum_{x \in \mathcal{B}_i^k} \left\| g_i(\mathbf{u}_i, x) - e^k \right\|_2^2. \tag{2.5}$$

Note that, the computation here is just linear in the number of data points as compared to quadratic in the case of MMD. Let $\mathbf{w}_{(i,t)} := (\mathbf{u}_{(i,t)}, \mathbf{v}_{(i,t)})$ be the parameters at ith worker and $\eta_{(t)}$ be the learning rate at time t. We define $\mathbf{e}_t = \{e_{(t)}^k\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$. Note that we use (t) to highlight that the subscript refers to the time. In Fed-CMA , we

perform the following updates on $\mathbf{w}_{(i,t)}, \bar{\mathbf{v}}_{(t)}, \mathbf{e}_{(t)}$ simultaneously,

$$\text{(Local update)} \begin{cases} \mathbf{w}_{(i,t+1)} & = \mathbf{w}_{(i,t)} - \eta_{(t)} \nabla_{\mathbf{w}_i} \widehat{\Phi}_i(\mathbf{w}_{(i,t)}, \bar{\mathbf{v}}_{(t)}, \mathbf{e}_{(t)}); \end{cases} \tag{2.6}$$

$$\text{(Global update)} \begin{cases} \bar{\mathbf{v}}_{(t+1)} & = \bar{\mathbf{v}}_{(t)}(1 - \lambda_1 \eta_{(t)}) + \lambda_1 \eta_{(t)} \Big(\sum_{i=1}^{M} p_i \mathbf{v}_{(i,t)} \Big); \\ e^k_{(t+1)} & = e^k_{(t)}(1 - \lambda_2 \eta_{(t)}) + \lambda_2 \eta_{(t)} \left(\sum_{i=1}^{M} p_i^k e_i^k \big(\mathbf{u}_{(i,t)} \big) \right), \end{cases} \tag{2.7}$$

where the empirical gradient $\widehat{\Phi}_i$ is computed using a minibatch ξ_i that is drawn i.i.d. from D_i and $0 < \lambda_1 \eta_{(t)} \leq 1$ and $0 < \lambda_2 \eta_{(t)} \leq 1$. The complete algorithm is given in Algorithm 1. We study the convergence properties of this algorithm in Section 2.5.

Algorithm 1 Fed-CMA

At Server:
for $t = 1, \cdots, T$ **do**
    Collect mean embeddings $\{e^k_{(i,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(i,t)}$ for all $i \in [M]$
    Update $\bar{\mathbf{e}}$ and $\bar{\mathbf{v}}$ using (2.7)
    Broadcast $\{e^k_{(t)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$ to the workers.
end for
At Worker i**:**
for $t = 1, \cdots, T$ **do**
    Receive $\{e^k_{(t)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$ from the server.
    Pick random minibatch $\xi_{(t)}$ from D_i and compute $\nabla \widehat{\Phi}_i$.
    Update $\mathbf{w}_{(i,t)}$ according to (2.6)
    Compute $\{e^k_{(i,t)}\}_{k=1}^K$ using $e_i^k = \frac{1}{b_i^k} \sum_{x \in \mathcal{B}_i^k} g_i(\mathbf{u}_{(i,t)}, x)$ for $k \in [K]$.
    Share with server: mean embeddings $\{e^k_{(i,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(i,t)}$.
end for

2.5 Convergence Results

Let us denote $\mathbf{a} = (\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$ and we can write the global objective as

$$\arg\min_{\mathbf{a}} \Phi(\mathbf{a}) := \arg \min_{\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \bar{\mathbf{e}}} \sum_{i=1}^{M} \Phi(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}). \tag{2.8}$$

Unless otherwise specified, $\|\cdot\|$ is $\|\cdot\|_2$ for vectors and $\|\cdot\|_{\mathrm{F}}$ for matrices. When we consider $\mathbf{a}$, we assume that all $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}$ are vectorized and concatenated such that $\mathbf{a} \in \mathbb{R}^{Md_i+2(d_e+1)K}$. We place the following assumptions on the loss functions and embedding maps.

Assumption 1. *The local losses $\{L_i(\mathbf{w}_i)\}_{i=1}^N$ are β_l Lipschitz continuous, and β_s smooth in $\mathbf{w}_i$. Note that $\|\mathbf{w}_i\|^2 = \|\mathbf{u}_i, \mathbf{v}_i\|^2 = \|\mathbf{u}_i\| + \|\mathbf{v}_i\|$. The losses L_i are lower bounded uniformly by a scalar $L_{\inf}$.*

Assumption 2. *The function $g_i(\mathbf{u}_i, x)$ is β_e Lipschitz continuous and β_g smooth $\forall i \in [M]$ with respect to $\mathbf{u}_i$. Moreover, we assume that $g_i(\mathbf{u}_i, x) \in [0,1]^{d_e}$.*

Assumption 3. *The empirical gradient is an unbiased gradient of the population loss $\mathbb{E}[\nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})] = \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$. The variance of the stochastic gradient is bounded, i.e.,*

$$\mathbb{E}\Big[\Big\|\nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) - \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})\Big\|^2\Big] \leq G_1 \Big\|\nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})\Big\|^2 + G_2^2.$$

Assumption 1 is commonly used in analyzing non-convex SGD. Assumption 2 implies that the output of $g_i(\mathbf{u}_i, x)$ is bounded and this is easily satisfied in deep neural networks where the activation function is a sigmoid function. While the boundedness of the activation function is not necessary, it makes some parts of the proof simpler. The more general ReLU activation can be used by making some changes to the loss function; more details are provided in the discussion after Lemma 13. Assumption 3 follows from assuming that the local dataset at each client i is drawn i.i.d from $\mathcal{D}_i$. For simplicity, we consider only one local update (2.6) before computing the global update (2.7). We now state the convergence theorem.

Theorem 3. *Let Assumptions 1, 2, and 3 hold and run the algorithm with T timesteps. If we chose a constant learning rate $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ that satisfies $0 < \eta \leq \min\left\{\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \frac{1}{C_0(G_1+1)}\right\}$, where $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, and C_0 is a problem specific constant, then we have $\min_{t:1,\cdots,T} \mathbb{E}\left[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\right] \leq \sqrt{\frac{2C_1C_0G_2^2}{T}}$.*

We defer the proof to Appendix. Note that here the algorithm converges to a first-order stationary point since we are dealing with smooth and non-convex functions. We now restrict the training only to the latent space to obtain convergence to the global optimum which we will then use to prove the generalization result.

Continuing FL in the latent space

Let $\left(\{(\mathbf{u}_i^*, \mathbf{v}_i^*)\}_{i=1}^M, \bar{\mathbf{v}}^*, \mathbf{e}^*\right)$ be the output from solving (2.8). We now freeze the representation functions $\{\mathbf{u}_i^*\}_{i=1}^M$ for all the clients. For every dataset $D_i = \{x_j, y_j\}_{j=1}^{N_i}$, we define a projected dataset $\tilde{D}_i = \{\left(z_j, y_j\right) | z_j = g_i(\mathbf{u}_i^*, x_j)\ \forall (x_j, y_j) \in D_i\}$. We now define the local objective as $\arg\min F_i(\mathbf{v}) := \frac{1}{N_i}\sum_{(x,y)\in D_i; z=g_i(\mathbf{u}_i^*,x)} l(\mathbf{v}, z, y)$. The global objective is then defined as

$$\arg\min_{\mathbf{v}} F(\mathbf{v}) := \sum_{i=1}^{M} p_i F_i(\mathbf{v}). \tag{2.9}$$

To solve (2.9), we initialize with $\mathbf{v}_{(i,0)} = \bar{\mathbf{v}}^*$, follow the FedAvg algorithm [61] and update the local weights $\mathbf{v}_{(i,t)}$ at $i-$th worker and the global weights $\mathbf{v}$ as

$$\mathbf{v}_{(i,t+\tau+1)} = \mathbf{v}_{(i,t+\tau)} - \eta_{(t+\tau)}\nabla\widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)}); \qquad \mathbf{v}_{(t+E)} = \sum_{i=1}^{M} p_i \mathbf{v}_{(i,t+E)}, \tag{2.10}$$

where τ is the local iteration number, $\nabla\widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)})$ is the stochastic gradient at time $t+\tau$ by sampling a local minibatch $\xi_{(i,t+\tau)}$ and $\eta_{(t+\tau)}$ is the learning rate. After every E local update, the server aggregates the local weights and for simplicity, we assume full device participation in updating the weights. After the global update,

the server assigns $\mathbf{v}_{(i,t+E)} = \mathbf{v}_{(t+E)}$ to every client. Before stating the convergence theorem, we make the following assumptions.

Assumption 4. *The local loss functions F_i are all $L-$smooth and μ strongly convex for all $i \in [M]$.*

Assumption 5. *The second moment of the stochastic gradients of the $i-th$ worker at time t is bounded as: $\mathbb{E}\Big[\Big\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)}) - \nabla F_i(\mathbf{v}_{(i,t)})\Big\|^2\Big] \leq \sigma_i^2$, for all $i \in [M]$.*

Assumption 6. *The expected squared norm of the stochastic gradients is uniformly bounded as $\mathbb{E}\Big[\Big\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)})\Big\|^2\Big] \leq G^2$ for all $i \in [M]$.*

The problem we solve in (2.9) is convex since the cross entropy loss is convex and we are only optimizing the final layer (classification) weights $\mathbf{v}$ (keeping the $\mathbf{u}_i^*$ fixed). Moreover, we can assume that each local objective uses an L2 regularization term, thus satisfying strong convexity. The Assumptions 4, 5, 6 are standard for analyzing FedAvg for non-iid scenarios [61].

Definition 1 ([61])**.** *Let F^*, F_i^* be the minimum values of F and F_i respectively. The degree of non-iid (heterogeneity) is defined as $\Gamma = F^* - \sum_{i=1}^{M} p_i F_i^*$.*

When the data $\tilde{D}_i$ are all i.i.d, then the degree of heterogeneity Γ goes to zero as the sample size increases. Here, the degree of heterogeneity between $\{\tilde{D}_i\}_{i=1}^M$ is determined by the degree of heterogeneity between $\{D_i\}_{i=1}^M$ and the learned $\{\mathbf{u}_i^*\}_{i=1}^M$. We now state the convergence theorem.

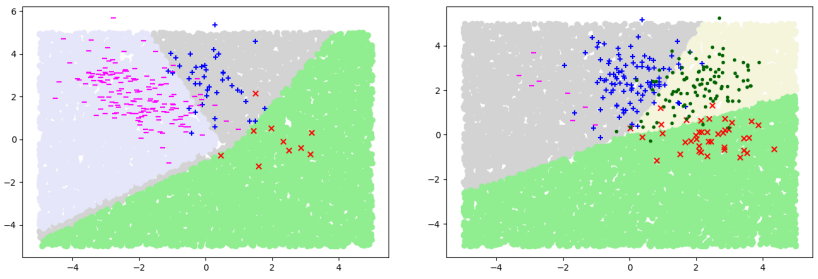
Theorem 4 (Theorem 1 [61])**.** *Let Assumptions 4 to 6 hold and L, μ, σ_i, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_{(t)} = \frac{2}{\mu(\gamma+t)}$. Then solving* (2.10) *with full client participation satisfies $\mathbb{E}[F(\mathbf{v}_{(T)})] - F^* \leq \frac{\kappa}{\gamma+T-1}\Big(\frac{2B}{\mu} +$*

$\frac{\mu\gamma}{2}\mathbb{E}[\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2]\Big)$, *where* $B = \sum_{i=1}^{M} p_i^2\sigma_i^2 + 6L\Gamma + 8(E-1)^2G^2$, $\mathbf{v}^*$ *is the unique minimizer of* (2.9).

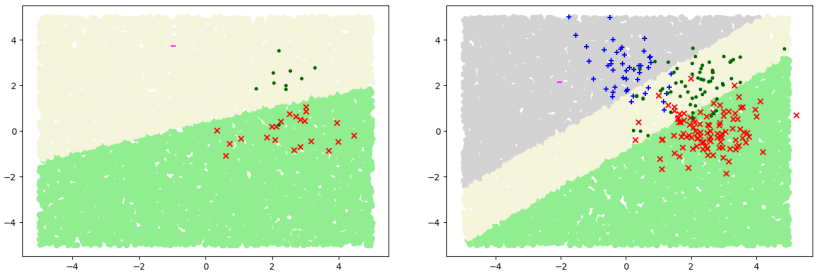
Theorem 4 states that applying FedAvg to the problem (2.9), the resultant iterate $\mathbf{v}_{(T)}$ convergences with a rate of $\mathcal{O}(\frac{1}{T})$. The faster rate compared to Theorem 3 is due to the simplification of the problem to the strongly convex case by freezing the $\{\mathbf{u}_i\}_{i=1}^{M}$. Moreover, unlike Theorem 3, where the algorithm converges to a local optimum, here the convergence is to the global optimum of (2.9). However, (2.9) itself depends on the frozen representations at which the latent space training took place. The rate of convergence slows with higher values of Γ, thus implying that the frozen $\{\mathbf{u}_i^*\}$ from solving (2.8) also play a role in the convergence of $\mathbf{v}_{(T)}$. Also, observe that the convergence rate depends on $\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2$. Suppose, the freezing is done when $\mathbf{u}_i$ are far away from minimizing (2.8), then $\bar{\mathbf{v}}^*$ is also far away from the optimal point and therefore it slows down convergence. Note that, while this dependence on $\mathbf{u}_i^*$ is not desirable from the convergence perspective, we want to highlight that we are dealing with model heterogeneity and this decoupling of $\mathbf{u}_i, \mathbf{v}_i$ is required to show a reasonable generalization result.

2.6 Numerical Simulations

We first evaluate Fed-CMA on a synthetic dataset to illustrate the conditional alignment in the latent space. We then compare Fed-CMA against other FL algorithms in the model heterogeneity setting on various image datasets. The details of the experimental setup is provided in Section A.4.1.

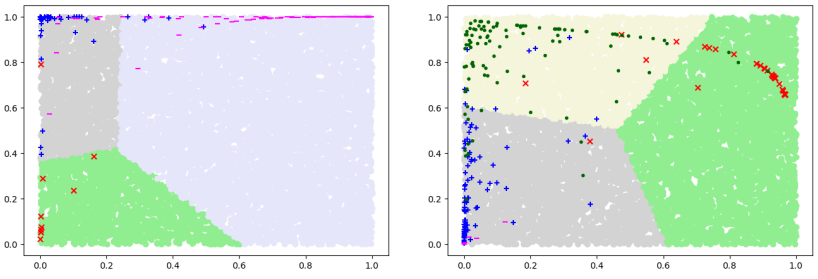


(a) Worker 0 **(b)** Worker 1

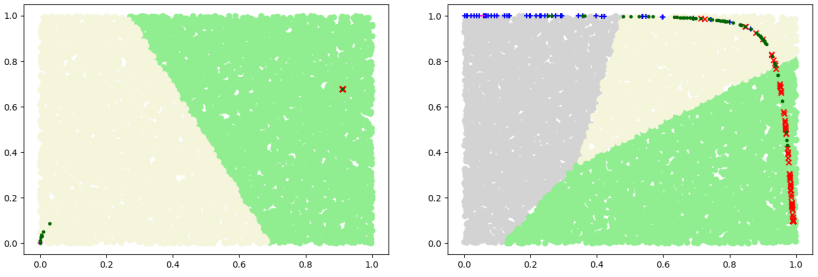


(c) Worker 2 **(d)** Worker 3

Figure 2.4: Local input space decision boundaries

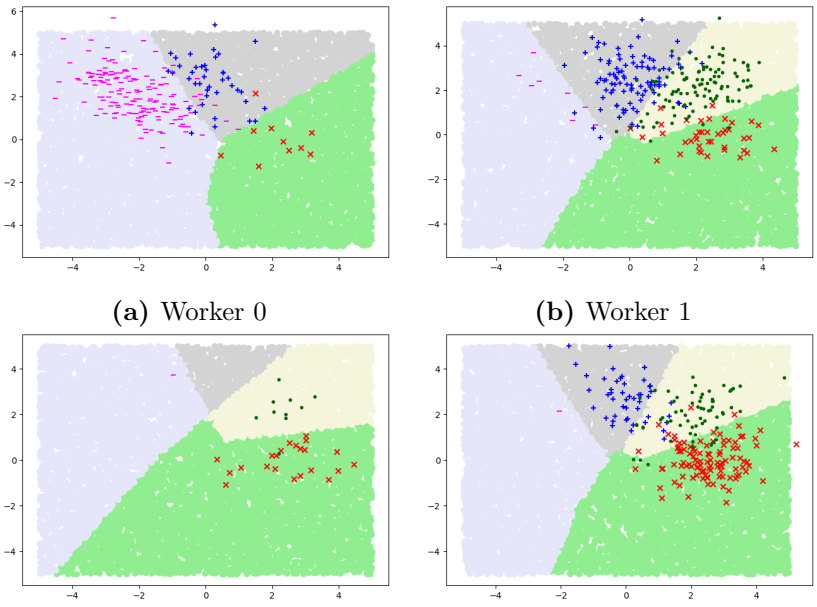


(a) Worker 0 **(b)** Worker 1



(c) Worker 2 **(d)** Worker 3

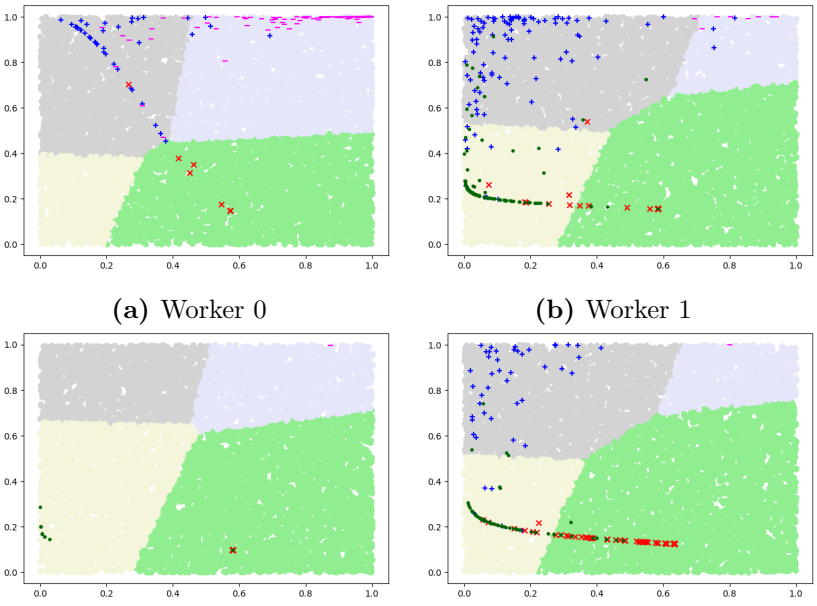
Figure 2.5: Local latent space decision boundaries



(a) Worker 0 **(b)** Worker 1

(c) Worker 2 **(d)** Worker 3

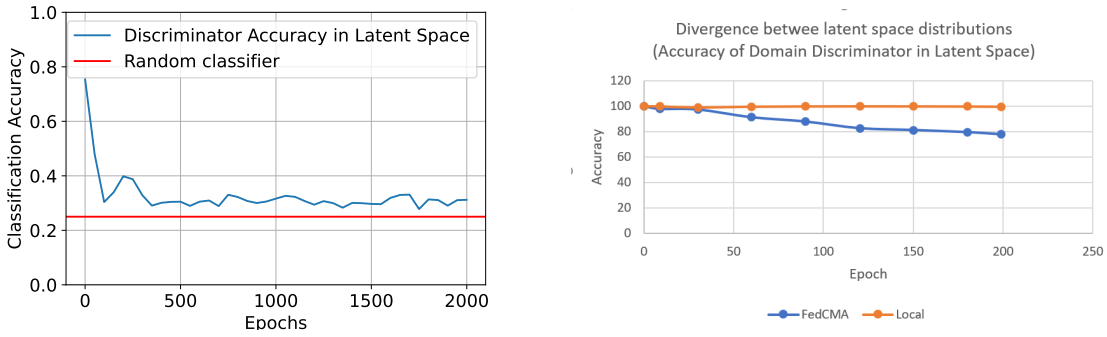
Figure 2.6: FedCMA input space decision boundaries



(a) Worker 0 **(b)** Worker 1

(c) Worker 2 **(d)** Worker 3

Figure 2.7: FedCMA latent space decision boundaries



(a) Synthetic dataset **(b)** CIFAR10 dataset

Figure 2.8: Discriminator classification accuracy (smaller the better)

Synthetic dataset We solve a 4-class classification problem, where we generate a synthetic dataset in $\mathbb{R}^2$ from a mixture of four Gaussian distributions and distribute them in a non-i.i.d. way among 4 clients (see Fig 2.4, 2.6). We use `Linear(2,2)` $\rightarrow$ `Linear(2,2)` $\rightarrow$ `Linear(2,4)` at every client for simplicity. After training, we plot the decision boundaries in the original and latent spaces (both $\mathbb{R}^2$). We see that the distributions and decision boundaries in the latent space are aligned for Fed-CMA (see Fig 2.7) resulting in much better decision boundary in the original space, even for workers with scarce data (see Fig 2.6). We empirically approximate $d_{\mathcal{V}\Delta\mathcal{V}}$ by training a classifier to discriminate between the latent space data of the four clients (motivated from [58], see supplementary material for more details). As we observe from Fig 2.8a, as the training progresses the accuracy of the discriminator reaches that of a random classifier which means the latent space data among the workers are more aligned and it gets difficult to tell from which client the data came from.

MNIST datasets We perform an evaluation on MNIST, EMNIST and FEMNIST [21] datasets. For MNIST and EMNIST, we consider $M = 20$ clients, and to simulate **data heterogeneity**, we follow [62] where a Dirichlet distribution $\text{Dir}_M(\alpha)$ is used to partition among the M clients. For smaller values of α, the data is more heterogeneous and for larger values of α the data is more homogeneous across the clients. For the MNIST and EMNIST datasets, we consider full client availability. In the FEMNIST experiment, we consider around 25% of the FEMNIST dataset which contains 923 clients with 50% client participation at each round.

Note that it is hard to compare the performance of Fed-CMA with earlier works such as Fed-ET [35], FedDF [34], and KT-pFL [33] since they utilize additional information in the form of a public dataset that is used for knowledge distillation. Moreover, these works may involve sharing the parameters with the server. Therefore, we do not provide a comparison with these works. While the main results of FedGen [40] require sharing all the model weights with the server, FedGen also accommodates limited parameter sharing (such as sharing only the classification layer weights $\mathbf{v}_i$). Therefore, in this section, we compare Fed-CMA with FedGen under the limited parameter setting. Additionally, we also provide a comparison against local training and against FedAvg which simply shares the final layer weights (called FedAvgSim in [41]). We use the same network and setup used in [40] using the official implementation[1] and allowed multiple local client updates between communication rounds.

From the results in Table 2.2, we observe that Fed-CMA outperforms all the baselines in the limited parameter setting. The improvement offered by Fed-CMA over the other baselines is much higher in the high data heterogeneity (small α) setting and

[1] `https://github.com/zhuangdizhu/FedGen`

the improvements diminish as α increases. This shows that in more practical settings, Fed-CMA can provide much-needed gains to justify participating in the FL setup. We provide detailed experimental setup and ablations (marginal alignment vs conditional alignment, advantage of sharing mean embeddings) in the supplementary material.

dataset	α	Local	FedAvg	FedGen	FedCMA
MNIST	0.02	76.28 (0.39)	76.14 (0.48)	75.95 (0.06)	**79.22** (0.74)
	0.03	59.89 (0.26)	59.82 (0.23)	60.08 (0.61)	**64.03** (0.26)
	0.05	61.36 (0.71)	61.26 (0.70)	60.82 (0.50)	**62.57** (0.39)
	0.10	62.27 (0.48)	62.26 (0.35)	62.84 (0.07)	**63.94** (0.42)
EMNIST	0.02	50.2 (0.10)	49.77 (0.11)	49.93 (0.06)	**51.0** (0.29)
	0.03	47.95 (0.33)	47.46 (0.24)	47.5 (0.32)	**49.26** (0.15)
	0.05	52.73 (0.35)	52.2 (0.36)	52.48 (0.33)	**53.96** (0.48)
	0.10	**47.31** (0.35)	46.84 (0.29)	46.35 (0.34)	46.5 (0.32)
FEMNIST	-	62.49 (0.64)	63.50 (0.79)	-	**64.50** (1.2)

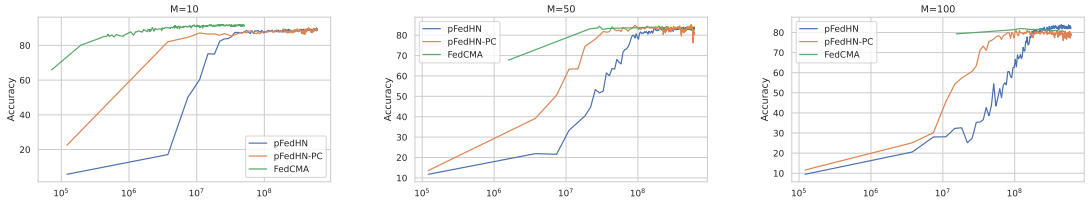
Table 2.2: Comparison on MNIST, EMNIST, and FEMNIST datasets with data heterogeneity and limited parameter sharing

CIFAR-10 dataset We also compare our work against the personalized FL algorithms pFedHN, pFedHN-PC in [63] for the CIFAR-10 dataset. We use the same non-i.i.d. data split, networks, and setup used in the official implementation of [63][2]. The comparison is provided in Table 2.3 and we observe that Fed-CMA outperforms the compared algorithms in two out of the three cases. Additionally, we also provide a comparison of the communication cost (in terms of the number of parameters) for the three algorithms in Fig 2.9. We observe that Fed-CMA achieves similar accuracy as the other two algorithms despite requiring an order of communication lesser. Note

[2] `https://github.com/AvivSham/pFedHN`

Workers	pFedHN	pFedHN-PC	FedCMA
10	88.34	89.05	**91.36**
50	83.62	83.62	**84.24**
100	**82.73**	80.81	80.92

Table 2.3: Comparison on CIFAR10 with data heterogeneity and limited parameter sharing



(a) M=10 **(b)** M=50 **(c)** M=100

Figure 2.9: Comparison of communication performed during training for CIFAR10 (x-axis) vs accuracy (y-axis)

that, the communication complexity of Fed-CMA at each round is $\mathcal{O}(K \times M \times d_e)$ irrespective of the dimension of the feature extractor weights. We also empirically approximate $d_{\mathcal{V}\Delta\mathcal{V}}$ by training a domain discriminator in the latent space and conclude from Fig 2.8b that the reduced accuracy of the domain discriminator is due to a better alignment in the latent space for Fed-CMA (which improves the generalization performance). Observe that for local training, the domain discriminator has 100% accuracy, implying that there is zero alignment of latent space distributions.

Chapter 3: Byzantine resilience to an arbitrary number of attackers

3.1 Introduction

With increasing data size and model complexity, the preferred method for training machine learning models at scale is to use a distributed training setting. This involves a parameter server that coordinates the training with multiple worker machines by communicating gradients and parameters. However, this setting is prone to fail in the presence of dishonest workers or non-malicious failed workers [20]. Under the more general setting, it is studied as the Byzantine general problem which dates back to reliable computing in distributed systems [64, 65, 66]. Recently, there has been a significant interest in devising distributed machine learning schemes to defend Byzantine adversaries [67, 68, 69, 70]. In this setting, a certain fraction of the workers are assumed to be adversarial; instead of sending the actual gradients computed using a randomly sampled mini-batch to the server, the adversarial workers send arbitrary or potentially adversarial gradients that could derail the optimization at the server. This problem has been studied under different settings such as gradient encoding [71], asynchronous updates [72, 73], heterogeneous datasets [74, 75], decentralized learning [76], [77, 78], Federated Learning [79, 80]. There has also been some work in

developing attack techniques that break existing defenses [79, 81, 82]. Please refer to [20] for a detailed list of references.

Type	Attack	Fraction of Adversaries f $f < 0.5$	$f \in [0.5, 1)$
Omniscient / Collusion	IPM [81]	√	√
Omniscient / Collusion	LIE [82]	√	-
Omniscient / Collusion	OFOM [79]	√	√
Omniscient / Collusion	PAF [79]	√	√
Local / Failure	Sign Flip/Reverse [69]	√	√
Local / Failure	Random Sign Flip	√	√
Local / Failure	Gaussian Attack [69]	√	√
Local / Failure	Constant Attack [74]	√	√
Data Poisoning	Label Flipping	√	√
Mixed Attacks	Multiple types of attacks	√	-

Table 3.1: Summary of various attacks that ByGARS or ByGARS++ is robust to.

Fraction of adversaries < 0.5 **:** One of the main assumptions in past studies about Byzantine attacks in machine learning is that the number of adversarial workers is less than half of the total number of workers. The fundamental reason for this assumption is that the majority-based robust statistics approaches require at least more than half of the samples to be correct to give a good estimate. For example, the underlying concept of geometric median [68] yields a robust estimator as long as less than half of the data (used for aggregation) is corrupted, and when more than half the data is corrupted, it provably fails.

Distance (between workers gradients) based methods such as Krum [69], and Bulyan [83] were proposed with convergence guarantees. Several robust statistics based approaches such as median of means [68], coordinate-wise median and trimmed

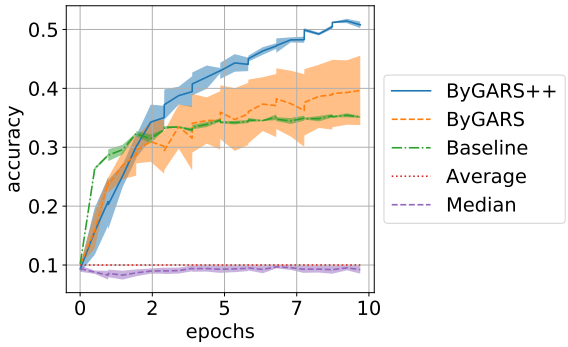


Figure 3.1: Comparison of the top-1 accuracy of ByGARS and ByGARS++ using CIFAR-10 dataset with one benign worker and seven attackers using different attack strategies. The seven attackers include one Gaussian adversary, two Sign flip adversaries, one random sign flip adversary, two label flip adversaries, and one constant value adversary.

mean [84], Lipschitz inspired coordinate median [77], majority-based SignSGD [85] and history based approaches [67], [86] were also shown to have strong convergence guarantees. Despite such guarantees, the above median and majority-inspired approaches fail to apply to an arbitrary number of attackers.

Arbitrary number of adversaries: A more practical and challenging problem is to ensure convergence even in the presence of an arbitrary number of adversaries. One way to address this is to assume the availability of auxiliary data at the server and filter out the adversarial gradients [87, 88, 73] by computing some score using the auxiliary data that differentiates adversarial gradients from benign ones. Jin et al. [89] consider the asynchronous update setting without an auxiliary dataset, but assume that every worker knows the gradients of every other worker and uses this information to locally filter out the adversarial gradients.

However, these methods require the number of adversaries (or an upper bound) which is crucial to filtering out the adversarial workers but this information may not

be available in practical scenarios. Moreover, if the estimate is inaccurate, it either leads to filtering out benign workers (reduction in effective data size thus affecting generalization) or allows more adversarial gradients to be considered as benign (thus affecting convergence).

In contrast, we propose a *reputation score* based aggregation, obviating the need to know the number of adversaries which makes the proposed approach more applicable to practical scenarios. The *reputation score* of a worker signifies how relevant the corresponding gradient direction is to the optimization problem, with the intuition being that workers with positive reputation score as being helpful towards the optimization and reputation scores with zero or negative values as being irrelevant or adversarial towards the optimization respectively. Concurrent to our work, Cao et al. [90] also develop a similar concept of reputation score to defend an arbitrary number of attackers in the Federated Learning setting. Ji et al. [91] also use an auxiliary dataset to aggregate gradients using an LSTM. In comparison, we use a simple reputation score-based aggregation, with theoretical guarantees.

We compute the *reputation score* of each worker using the auxiliary dataset. We expect that two stochastic gradients (computed on small mini-batches of the two datasets with similar distributions) would make a small angle with each other with high probability. On the other hand, a random vector will be almost orthogonal to the correct stochastic gradients due to the intrinsic properties of random high-dimensional vectors. This intuition allows us to compute the reputation score for each worker reliably early on in the training process. We summarize our main contributions as follows

1. We propose a novel reputation score-based gradient aggregation method for distributed machine learning with learnable reputation scores.

2. We show that our algorithm is Byzantine tolerant (in the sense of [81]) to an arbitrary number of attackers. Unlike previous works (that filter out adversaries) that require an upper bound on the number of adversaries, we do not assume such knowledge.

3. We use two time-scale stochastic approximation theories to establish the convergence of the proposed algorithm under reasonable assumptions (with strongly convex loss function)

4. Empirical evidence on strongly convex and non-convex objectives suggests that our proposed algorithms are robust to almost all state-of-the-art Byzantine attacks. We also show that our algorithms can defend some *mixed attacks* where multiple different attacks are performed at the same time (see Fig 3.1).

3.2 Problem setup

We consider distributed machine learning with a *parameter server - worker* setup. The parameter server maintains the model parameters and updates the parameters with gradients received from the workers. We denote the model parameters by $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^d$ and the number of workers by m. We assume that each worker j has access to dataset, $D_j := \{x_i^j, y_i^j\}_{i=1}^{n_j} \sim \mathcal{D}$, where $N = \sum_j n_j$ is the total number of data points. In the traditional distributed machine learning scenario, the server assigns the data partitions to the workers uniformly at random. In the Federated Learning scenario, this translates to each worker having its own dataset, which is not

shared with anyone (not even the server). Given a loss function $f(\cdot, x, y) : \mathbb{R}^d \to \mathbb{R}$, $x, y \sim \mathcal{D}$, the objective is to minimize the population loss $F(\mathbf{w}) := \mathbb{E}_{x,y\sim\mathcal{D}}[f(\mathbf{w}, x, y)]$, i.e., $\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w})$. We denote the true gradient of the population loss at $\mathbf{w}_t$ by $\nabla F(\mathbf{w}_t)$. A good worker samples a subset of the data $\mathcal{D}_{j,t} \subset \mathcal{D}_j$, and computes a stochastic gradient $\tilde{h}_{t,j} := \frac{1}{|\mathcal{D}_{j,t}|} \sum_{x,y\in\mathcal{D}_{j,t}} \nabla f(\mathbf{w}_t, x, y)$. The good workers communicate the stochastic gradient $h_{t,j} := \tilde{h}_{t,j}$ to the server, whereas adversarial workers send an arbitrary vector drawn from some distribution (either by computing the stochastic gradient on its subset of data and modifying the gradient adversarially or by sending an arbitrary random vector). We assume that this attack distribution remains fixed for the adversary throughout the training. We denote the set of gradients received by the server as $H_t^T = [h_{t,1}, \cdots, h_{t,m}] \in \mathbb{R}^{d\times m}$. Note that we assume a synchronous setting here, i.e. all the workers communicate the gradients at the same time to the server. We assume that the server has access to an auxiliary dataset $D_{aux} := \{x_i, y_i\}_{i=1}^n \sim \mathcal{D}$. The server can sample a subset $\xi_{aux,t}$ of the auxiliary dataset and compute auxiliary loss $L_t(\mathbf{w}) = \frac{1}{|\xi_{aux,t}|} \sum_{(x,y)\in\xi_{aux,t}} f(\mathbf{w}, x, y)$, such that $\mathbb{E}[\nabla L_t(\mathbf{w}_t)] = \nabla F(\mathbf{w}_t)$ when the expectation is taken with respect to the draw of the dataset.

3.3 Algorithm

In this section, we first motivate the importance of using a reputation score for gradient aggregation and introduce the ByGARS class of algorithms to compute the reputation scores and aggregate gradients.

The idea of the reputation score is to ensure that the aggregated gradient is a descent direction for the population loss F. Suppose, at time t, the reputation score vector is $\mathbf{q}_t = [q_{t,1}, \ldots, q_{t,m}]^T$ and the received gradients are H_t, then the weighted

aggregation of the gradients with the reputation score is $H_t^T \mathbf{q}_t = \sum_{i=1}^m q_{t,i} h_{t,i}$. Consider the non-adversarial case where all workers correctly send the computed gradients. Then $q_{t,i} = 1/m$, and $-H_t^T \mathbf{q}_t$ is a descent direction (and an unbiased estimate of the gradient). Consider the case where we know which workers are adversarial, simply making the reputation scores $q_{t,i}$ for those workers equal to 0 is enough to defend against the attacks. The problem now is to compute a good reputation score for the workers using only the gradients sent to the server, such that the aggregated gradient is a descent direction for the objective. We propose the ByGARS (Byzantine Gradient Aggregation using Reputation Scores) class of algorithms.

3.3.1 ByGARS

Algorithm 2 ByGARS: Byzantine Gradient Aggregation using Reputation Scores

$\mathbf{w}_0$ initialized randomly and sent to workers
$\mathbf{q}_0 = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
    receive $H_t^T = [h_{t,1}, \cdots, h_{t,m}]$ from workers
    $\mathbf{q}_{t+1}^0 = \mathbf{q}_t$
    for $i = 1, \cdots, k$ **do**
        $\widehat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}^{i-1}$ ▷ *pseudo update*
        $\mathbf{q}_{t+1}^i \leftarrow \mathbf{q}_{t+1}^{i-1} + \alpha_t \gamma_t H_t \nabla L_t(\widehat{\mathbf{w}}_{t+1})$ ▷ *meta update*
    end for
    $\mathbf{q}_{t+1} = \mathbf{q}_{t+1}^k$
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_{t+1}$ ▷ *actual update*
    Send $\mathbf{w}_{t+1}$ to workers
end for
Return $\mathbf{w}_{T+1}$

We start with an initial reputation score of $\mathbf{q}_0 = \mathbf{0} \in \mathbb{R}^m$, and iteratively improve the estimate of the reputation score. At step t, we perform a *pseudo update* to $\mathbf{w}_t$ (γ_t

is a step size parameter) as

$$\hat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t. \tag{3.1}$$

If $\mathbf{q}_t$ is a good reputation score and γ_t is sufficiently small, then $-H_t^T \mathbf{q}_t$ is a descent direction and thus $F(\hat{\mathbf{w}}_{t+1})$ must be lower in value than $F(\mathbf{w}_t)$ or other points in its neighborhood. However, we neither have access to the true function F nor the data from the workers. Instead, we have a small auxiliary dataset that is drawn from the same distribution as the data at the workers. This auxiliary dataset allows us to construct the loss function $L_t(\cdot)$ (see Section 3.2), and we can solve the following optimization problem to compute a better reputation score $\mathbf{q}_{t+1}^*$:

$$\mathbf{q}_{t+1} = \arg\min_{\mathbf{q} \in \mathbb{R}^m} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}). \tag{3.2}$$

Using the current estimate $\mathbf{q}_t$, we use an iterative update rule. We compute the loss on a random mini-batch of the auxiliary dataset D_{aux} using $\hat{\mathbf{w}}_t$, which is denoted as $L_t(\hat{\mathbf{w}}_t) = L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t)$, and henceforth referred to as *auxiliary loss*. The objective is to find a descent direction on the auxiliary loss at the current iterates. We do this by performing a first-order update to $\mathbf{q}_t$ by computing the gradient of the auxiliary loss evaluated at $\hat{\mathbf{w}}_t$ wrt $\mathbf{q}_t$. We refer to this gradient as the auxiliary gradient denoted by $\nabla L_t(\hat{\mathbf{w}}_t)$. The gradient computation and *meta update* to $\mathbf{q}_t$ (α_t is a step size parameter) is given by

$$\begin{aligned} \mathbf{q}_t \leftarrow \ & \mathbf{q}_t - \alpha_t \frac{d}{d\mathbf{q}_t} L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t) \\ & = \mathbf{q}_t - \alpha_t(-\gamma_t H_t) \nabla L_t(\mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t) \\ & = \mathbf{q}_t + \alpha_t \gamma_t H_t \nabla L_t(\hat{\mathbf{w}}_t). \end{aligned} \tag{3.3}$$

The updated reputation score is used to find the updated gradient aggregation $H_t^T \mathbf{q}_t$ which is used for the actual update. This is summarized in Algorithm 2 (note the change in notation, eg. superscript i, due to the meta updates).

3.3.2 ByGARS++: Computationally Efficient

Algorithm 3 ByGARS++: Faster algorithm to compute Reputation Scores

$\mathbf{w}_0$ initialized randomly and sent to workers
$\mathbf{q}_0 = \mathbf{0}$
for $t = 1, \cdots, T$ **do**
    Receive $H_t^T = [h_{t,1}, \cdots, h_{t,m}]$ from workers
    Compute $\nabla L_t(\mathbf{w}_t)$ using a subset of the auxiliary data
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t$ ▷ *actual update*
    Send $\mathbf{w}_{t+1}$ to workers
    $\mathbf{q}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t)$ ▷ *meta update*
end for
Return $\mathbf{w}_{T+1}$

Algorithm 2 has an additional computational overhead due to multiple parameter updates and multiple gradient computations to update the reputation score in the meta updates. This increased computation at the server keeps the workers idle and waiting, thus negating the computational speed-up achieved from distributed learning. In order to overcome this limitation, and driven by the motivation of ByGARS, we propose a computationally efficient algorithm and use it to prove theoretical convergence guarantees (Section 3.4).

We propose Algorithm 3 (ByGARS++), in which we avoid computing multiple *pseudo updates* $\hat{\mathbf{w}}_t$ used for performing *meta updates*, by simulataneously updating $\mathbf{w}_t, \mathbf{q}_t$ as given by eq (3.4). Note that we perform an update to $\mathbf{q}_t$ using the auxiliary

gradients evaluated at $\mathbf{w}_t$ (and not at $\hat{\mathbf{w}}_t$), and we use a stochastic approximation for the update.

$$\begin{aligned} \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t, \\ \mathbf{q}_{t+1} &\leftarrow (1-\alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t) \end{aligned} \tag{3.4}$$

In this case, the reputation score of each worker is updated using only the inner product between the gradient sent by the worker, and the auxiliary gradient, both evaluated at $\mathbf{w}_t$. The only additional computation as compared to traditional distributed SGD is the update of $\mathbf{q}_t$ which takes $\mathcal{O}(md)$ time. However, the server can update $\mathbf{q}_t$ when the workers are computing the gradients for the next time step (line 7, 8 of Algorithm 3), therefore ByGARS++ has the same computational complexity as traditional distributed SGD.

3.3.3 Reputation Scores

If a worker consistently sends gradients that are not in the descent direction (of the optimization at the respective iterates), then the reputation score is either zero or accumulates negative values (since the inner product between the worker gradient and auxiliary gradient is either negative or close to zero in expectation). Therefore, by multiplying the received worker gradient by the reputation score, either its impact on the aggregated gradient is reduced (when the reputation score is close to zero) or recover the actual direction of descent (when the reputation score is negative and the worker sends gradients that make an obtuse angle with the auxiliary/true gradient in expectation).

When the parameters are far away from the optima, the inner product between the benign gradients and the auxiliary gradients are positive and higher in magnitude, and

therefore contribute heavily towards the reputation scores. Whereas when we are closer to the optima, the inner product value is random [92] (due to the directions of the stochastic gradients being random) and hence does not contribute to the reputation score. This phenomenon can destroy the reputation of good workers/boosts that of adversaries, therefore we employ a decaying learning rate schedule for both γ_t and α_t. Thus, by the time the parameters are close enough to the optima or a flat region (in non-convex settings), the learning rates would have decayed significantly. This enables the reputation score to accumulate over time and converge; therefore, the score is robust to the noisy inner products near the optima. As we will see in Section 3.4, the decaying learning rate is required for the analysis of the algorithm under the two-timescale stochastic approximation theory.

It is important to note that the algorithms rely on the availability of the auxiliary dataset. It is a reasonable assumption, for example, this data can be taken from publicly available datasets (that match the distribution of the data available at the workers), from prior data leaks (that is now publicly available), or data given voluntarily by workers. We provide more analysis on the effect of auxiliary dataset size on the performance of our algorithms in Section 3.5.5 and observe that a very small auxiliary dataset size is sufficient.

3.4 Convergence of ByGARS++

We first state the assumptions that are important to show convergence of Algorithm 3. We first define a filtration. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a standard probability space

and let $\mathcal{F}_t$ be the σ-algebra generated by all the randomness realized up to time t:

$$\begin{aligned}\mathcal{F}_0 =&\sigma\{\mathbf{w}_0, \mathbf{q}_0\},\\ \mathcal{F}_t =&\sigma\{\mathbf{w}_0, h_{1,0}, \ldots, h_{m,0}, \mathbf{q}_0, \ldots, \mathbf{w}_{t-1}, h_{1,t-1}, \ldots, h_{m,t-1}, \mathbf{q}_{t-1}, \mathbf{w}_t, \mathbf{q}_t\}.\end{aligned}$$

It is easy to see that $\mathcal{F}_t \subset \mathcal{F}_{t+1}$, and thus, $\{\mathcal{F}_t\}_{t\in\mathbb{N}}$ is a filtration.

Assumption 1. *The Byzantine adversaries corrupt the gradients using multiplicative noise. If the worker i computes a stochastic gradient $\tilde{h}_{t,i}$ which is an unbiased estimate of $\nabla F(\mathbf{w}_t)$, the worker sends $h_{t,i} := \tilde{\kappa}_{t,i}\tilde{h}_{t,i}$ to the parameter server, where $\tilde{\kappa}_{t,i}$ is an i.i.d multiplicative noise with mean κ_i and finite second moment. The random noise satisfies $|\tilde{\kappa}_{t,i}| \leq \kappa_{\max}$ almost surely for all the workers. The workers have the following types:*

1. *Benign worker: $\mathbb{E}[h_{t,i}|\mathcal{F}_t] = \nabla F(\mathbf{w}_t)$ with $\kappa_i = 1$;*
2. *Scaled adversary: $\mathbb{E}[h_{t,i}|\mathcal{F}_t] = \kappa_i \nabla F(\mathbf{w}_t)$, where κ_i is a real number (negative or positive);*
3. *Random adversary: $\mathbb{E}[h_t|\mathcal{F}_t] = 0$,where adversary sends random gradients with mean 0 (i.e. $\kappa_i = 0$).*

There is at least one benign or scaled adversary with $\kappa_i \neq 0$ among the workers. Further, we assume the adversaries' noise distributions do not change with time.

Assumption 2. *The iterates are bounded, i.e.,* $\sup_t \|\mathbf{w}_t\|, \sup_t \|\mathbf{q}_t\| < \infty$.

Assumption 3. *The stochastic gradients computed at the server (using the auxiliary dataset) are unbiased estimates of the true gradient $\nabla F(\mathbf{w}_t)$, i.e., $\mathbb{E}[\nabla L_t(\mathbf{w}_t)|\mathcal{F}_t] = \nabla F(\mathbf{w}_t)$. Moreover, all the stochastic gradients have bounded noise variance, i.e., $\mathbb{E}[\|\nabla L_t(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\|^2|\mathcal{F}_t] \leq \bar{\sigma}^2$ and $\mathbb{E}[\|\tilde{h}_{t,i} - \nabla F(\mathbf{w}_t)\|^2|\mathcal{F}_t] \leq \bar{\sigma}^2$ where $\bar{\sigma} \in (0, \infty)$.*

Assumption 4. *The population loss F is c-strongly convex, with $\mathbf{w}_*$ as the unique global minimum with $\nabla F(\mathbf{w}_*) = 0$. Further, ∇F is a locally Lipschitz function with bounded gradients.*

Assumption 1 is motivated by Xie et al. [81] where the adversary sends a negatively scaled gradient of the sum of the gradients of benign workers. This is a reasonable assumption as the goal of the attacker is to derail the training progress by corrupting the aggregate gradient so that it is not a descent direction. This also includes system failures, where the sign bit of the gradient is flipped erroneously during communication [88]. This adversary model is used in several works including [85, 74, 73]. Although this assumption is restrictive, it is important to emphasize that it is required only for the theoretical analysis, but our empirical results show that the proposed algorithms are robust to different types of attacks (summarized in Table 3.1). Assumption 2 of bounded iterates is typical in stochastic approximation literature; see, for instance, [93, 94, 95]. To avoid making this assumption, a typical workaround is to project the iterates back to a very large set in case the iterates go outside the set [96, 94]. For single timescale stochastic approximation, [97] derives some sufficient conditions under which the iterates would be automatically bounded almost surely. However, we are unable to leverage this result since this method has not been extended to two-timescale stochastic approximation. Assumptions 3, 4 are standard assumptions to show the convergence results.

3.4.1 Main Result

We can now establish the following result.

Lemma 2. *The following holds under Assumptions 1, 3:*

$$\mathbb{E}[H_t|\mathcal{F}_t] = \boldsymbol{\kappa}\nabla F(\mathbf{w}_t)^T \tag{3.5a}$$

$$\mathbb{E}[H_t\nabla L_t(\mathbf{w}_t)|\mathcal{F}_t] = \boldsymbol{\kappa}\|\nabla F(\mathbf{w}_t)\|^2 \tag{3.5b}$$

Proof. We know that $h_{t,i} = \tilde{\kappa}_{t,i}\tilde{h}_{t,i}$, where $\tilde{\kappa}_{t,i}$ and $\tilde{h}_{t,i}$ are independent of each other. Therefore, $\mathbb{E}[H_t|\mathcal{F}_t] = \boldsymbol{\kappa}\nabla F(\mathbf{w}_t)^T$ due to the unbiasedness of the stochastic gradients $\tilde{h}_{t,i}$. Moreover, the randomness in $\nabla L_t(\mathbf{w}_t)$ is independent of the randomness in H_t given $\mathcal{F}_t$, therefore, $\mathbb{E}[H_t\nabla L_t(\mathbf{w}_t)|\mathcal{F}_t] = \mathbb{E}[H_t|\mathcal{F}_t]\mathbb{E}[\nabla L_t(\mathbf{w}_t)|\mathcal{F}_t] = \boldsymbol{\kappa}\|\nabla F(\mathbf{w}_t)\|^2$. □

We now state the main convergence result.

Theorem 5. *Suppose that* $\{\alpha_t\}, \{\gamma_t\}$ *are diminishing stepsizes, that is,* $\sum \alpha_t = \infty, \sum \gamma_t = \infty, \sum \alpha_t^2 < \infty, \sum \gamma_t^2 < \infty$, *with* $\gamma_t/\alpha_t \to 0$ *as* $t \to \infty$. *If Assumptions 1, 2, 3, 4 are satisfied, then* $\{\mathbf{w}_t\}$ *generated by ByGARS++ converges almost surely to* $\mathbf{w}_*$.

The proof leverages two-timescale stochastic approximation to establish the above result. Observe that the results requires $\frac{\gamma_t}{\alpha_t} \to 0$ as $t \to \infty$, i.e., the learning rate γ_t (to update $\mathbf{q}_t$) decays faster than the learning rate α_t (to update $\mathbf{w}_t$). In other words, we want $\mathbf{q}_t$ to converge faster to a stable value than $\mathbf{w}_t$ and the importance of this can be observed from the proof of the theorem. We now prove Theorem 5.

3.4.2 Proof of Theorem 5

We use $\boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_m]^T$ to denote the mean vector of the random multiplicative noises of the workers. We show that all the hypotheses of Theorem 2 of [95] are satisfied by ByGARS++, which leads to the desired convergence result. Consider the

two functions and the corresponding differential equations (here $\mathbf{w}, \mathbf{q}$ are functions of continuous time and $\dot{\mathbf{w}}(t)$ denotes differentiation with respect to time t):

$$G_1(\mathbf{w}, \mathbf{q}) = -(\boldsymbol{\kappa}^T \mathbf{q}) \nabla F(\mathbf{w}), \quad \dot{\mathbf{w}}(t) = G_1(\mathbf{w}(t), \bar{\mathbf{q}}(t)) \tag{3.6}$$

$$G_2(\mathbf{w}, \mathbf{q}) = -\mathbf{q} + \boldsymbol{\kappa} \|\nabla F(\mathbf{w})\|^2, \quad \dot{\mathbf{q}}(t) = G_2(\mathbf{w}, \mathbf{q}(t)). \tag{3.7}$$

Lemma 3. *The differential equation in* (3.7) *has a unique globally asymptotically stable equilibrium, which is denoted by* $\phi(\mathbf{w})$ *and is given by* $\phi(\mathbf{w}) = \boldsymbol{\kappa} \|\nabla F(\mathbf{w})\|^2$. *The differential equation in* (3.6) *with* $\bar{\mathbf{q}}(t) = \phi(\mathbf{w}(t))$ *has a unique globally asymptotically stable equilibrium* $\mathbf{w}_*$.

Proof. To see the first result, note that for any $\mathbf{w}$, the solution to the differential equation is $\mathbf{q}(t) = (\mathbf{q}(0) - \boldsymbol{\kappa} \|\nabla F(\mathbf{w})\|^2) \exp(-t) + \boldsymbol{\kappa} \|\nabla F(\mathbf{w})\|^2$, which is globally asymptotically stable with $\phi(\mathbf{w}) = \mathbf{q}(\infty) = \boldsymbol{\kappa} \|\nabla F(\mathbf{w})\|^2$. We now use the Lyapunov stability theory to establish the second statement. Let us substitute into (3.6) $\bar{\mathbf{q}}(t) = \phi(\mathbf{w}(t))$. Now define the Lyapunov function $V(\mathbf{w}) := F(\mathbf{w}) - F(\mathbf{w}_*)$, which is a valid Lyapunov function since F is strongly convex by Assumption 1. We get

$$\nabla V(\mathbf{w}(t))^T G_1(\mathbf{w}(t), \phi(\mathbf{w}(t))) =$$

$$- \|\boldsymbol{\kappa}\|^2 \|\nabla F(\mathbf{w}(t))\|^4 < 0 \text{ for all } \mathbf{w}(t) \neq \mathbf{w}_*.$$

Thus, the differential equation has a unique globally asymptotically stable equilibrium where the Lyapunov function is 0, that is, at $\mathbf{w}_*$. □

Define $u_t = -G_1(\mathbf{w}_t, \mathbf{q}_t) - H_t^T \mathbf{q}_t$ and $v_t = H_t \nabla L_t(\mathbf{w}_t) - \boldsymbol{\kappa} \|\nabla F(\mathbf{w}_t)\|^2$. Using the definition of u_t and v_t, the algorithm ByGARS++ is re-written as

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \gamma_t H_t^T \mathbf{q}_t \\ &= \mathbf{w}_t + \gamma_t G_1(\mathbf{w}_t, \mathbf{q}_t) + \gamma_t u_t \end{aligned} \tag{3.8}$$

$$\begin{aligned} \mathbf{q}_{t+1} &= (1-\alpha_t)\mathbf{q}_t + \alpha_t H_t \nabla L_t(\mathbf{w}_t) \\ &= \mathbf{q}_t + \alpha_t G_2(\mathbf{w}_t, \mathbf{q}_t) + \alpha_t v_t. \end{aligned} \tag{3.9}$$

We now need to show that u_t and v_t are martingale difference stochastic processes. From (3.5a), we observe that $\mathbb{E}[H_t|\mathcal{F}_t] = \boldsymbol{\kappa} \nabla F(\mathbf{w}_t)^T$, which implies that $\mathbb{E}[u_t|\mathcal{F}_t] = 0$. Further, it is easy to deduce that $\mathbb{E}[\|u_t\|^2|\mathcal{F}_t] \leq M(1 + \|\mathbf{q}_t\|^2)$ which follows from Assumption 3 and for a large $M > 0$ that depends on the bounds on $\|\nabla F(\mathbf{w}_t)\|$ and $\|\kappa\|$. Thus, u_t is a martingale difference noise. Next, from (3.5b), $\mathbb{E}[H_t \nabla L_t(\mathbf{w}_t)|\mathcal{F}_t] = \boldsymbol{\kappa} \|\nabla F(\mathbf{w}_t)\|^2$, therefore, $\mathbb{E}[u_t|\mathcal{F}_t] = 0$. Again, it is easy to show that $\mathbb{E}[\|v_t\|^2|\mathcal{F}_t] \leq M(1 + \|\mathbf{q}_t\|^2)$. This implies $\{v_t\}$ is also a martingale difference noise.

It is clear from the expressions that since ∇F is locally Lipschitz, G_1 and G_2 are locally Lipschitz maps. In addition, $\phi(\mathbf{w})$ is also a locally Lipschitz map. Theorem 5 now follows from the result in Lemma 3 and Theorem 2 of [95].

3.5 Simulations

For the theoretical results, we assume strongly convex loss functions and multiplicative noise adversaries. However, for the empirical results, we show that these assumptions are not required. In particular, we consider non-convex loss functions and several different attacks and show the efficacy of the proposed algorithms. We present the results of our algorithms on CIFAR-10 [98] for multi-class classification using supervised learning. We used a two convolutional layer CNN for CIFAR-10. For each dataset, we set aside a small auxiliary dataset of size 250 (sampled randomly

from the training data) at the server, and the remaining training data is distributed uniformly to the workers. We assume a setup with 1 server and 8 workers. In order to show the robustness of our proposed algorithms to an arbitrary number of attackers, we consider the number of attackers $\in \{3, 6, 8\}$.

3.5.1 Attack mechanisms (Table 3.1)

The attacks were grouped broadly into (i) Omniscient / Collusion attacks, (ii) Local attacks / System failures, and (iii) Data Poisoning attacks. We further propose a *mixed attack*, where we combine multiple Local attacks and Data Poisoning attacks. In the **Omniscient / Collusion** attacks, the adversaries have complete information about all other workers including the benign ones, or only about the other adversaries. In LIE attack [82], the adversary adds well-crafted perturbations to the empirical mean of the benign gradients that is sufficient to avoid convergence. In OFOM, PAF [79] a large arbitrary vector is added to the empirical mean of the benign gradients and sent to the server. In Inner Product Manipulation, Xie et al. [81] multiply the empirical mean of benign gradients and in IPM [81] the empirical mean of the benign gradients is multiplied with a negative value.

In **Local** attacks, the attacker doesn't have any information about the other workers. Instead, the worker either sends an arbitrary gradient to the server or uses the gradient it computed. Examples of the first case include a Gaussian Attack (a vector drawn from a Gaussian distribution of mean 0, covariance 200) [69], or Constant attack (a vector of all 1s multiplied by an arbitrary scalar, say 100) [74]. In the second case, the attacker can compute the local stochastic gradient $\tilde{h}_{t,i}$, and send $\tilde{\kappa}_{t,i}\tilde{h}_{t,i}$ where $\tilde{\kappa}_{t,i} = -1$ is known as sign flip attack [69, 74, 85, 89]. Hardware/communication

failures that corrupt the gradients (unintentionally) by flipping the sign bit can also be included under these attacks. In addition to the sign flipping attack, we propose a *random* sign flip attack where $\tilde{\kappa}_{t,i} \sim \mathcal{N}(\mu, \sigma^2)$. In our simulations, we used $\mu = -2$ and $\sigma = 1$. Note that the mean of the distribution can take negative as well as positive values.

In **Data Poisoning** attacks, the underlying data used to compute the gradients is poisoned so that the model outputs the attacker-chosen targets during inference [82]. There are several varieties of data poisoning (also called backdooring) attacks, but we only consider label-flipping attacks in this work. Under the label flipping attack, for example in the MNIST dataset, the attacker maps the labels as $l \to (9 - l)$ for $l \in \{0, \cdots, 9\}$, and uses these labels for computing the gradients. Note that, label flipping attack is also a Local attack.

In addition to these attacks, we propose a *Mixed Attack* where seven attackers use different attacks (one Gaussian, two Sign flips, one random sign flip, two label flips, and one constant attacker). This is motivated by the need to develop algorithms that are robust to several different types of attacks at the same time. We summarize all these attacks in Table 3.1.

3.5.2 Baselines

The generalization performance of SGD improves with the size of the dataset [99]. Since existing techniques for an arbitrary number of Byzantines filter out the gradients that are sent by the adversarial workers, (in a truly distributed Setting) the generalization performance of those techniques is limited by the data at benign workers (and the number of byzantine attackers perceived by the algorithm) and the

amount of data available with them. In the case where all workers are adversaries, the only available truthful data is the auxiliary dataset. Hence, we consider plain averaging of the gradients available at these benign workers (and auxiliary gradient)

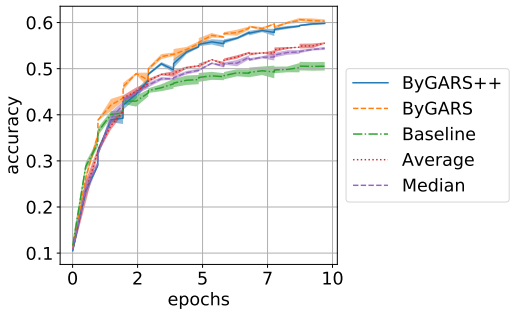
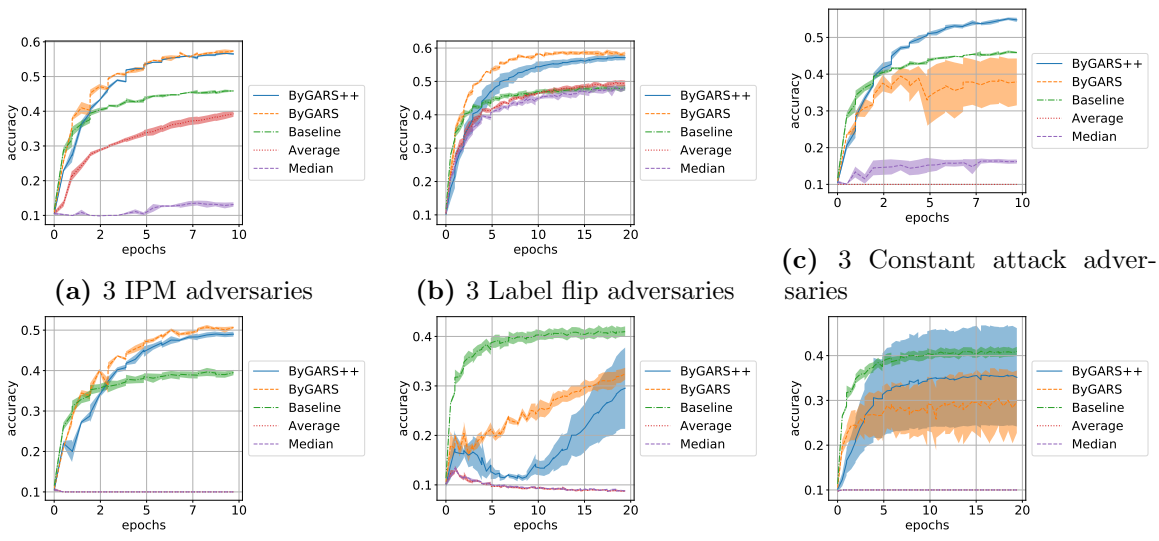


Figure 3.2: This figure shows the top-1 accuracy of models trained on CIFAR-10 dataset under *No Attack*.



(a) 3 IPM adversaries

(b) 3 Label flip adversaries

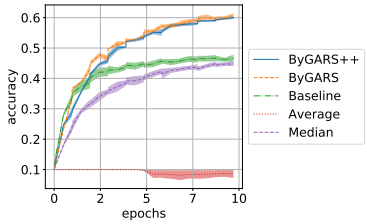
(c) 3 Constant attack adversaries

(d) 6 IPM adversaries

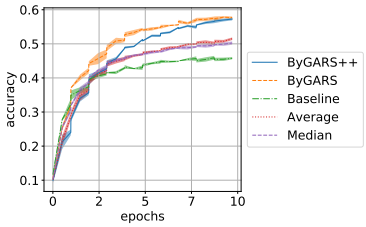
(e) 6 Label flip adversaries

(f) 6 Constant attack adversaries

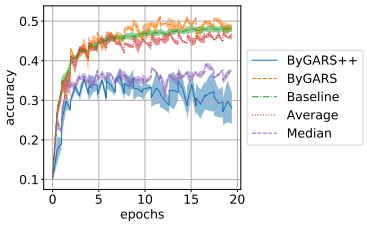
Figure 3.3: Top-1 accuracy of CIFAR-10 under IPM, label flip, and constant attack adversaries. Note the reduction in performance from the top row to the bottom row, due to the increase in the number of adversaries.



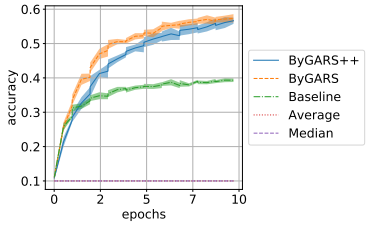
(a) 3 Random sign flip adversaries



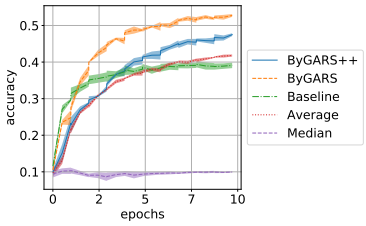
(b) 3 OFOM adversaries



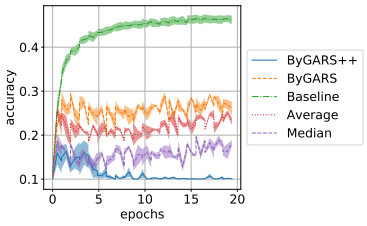
(c) 3 LIE adversaries



(d) 6 Random sign flip adversaries

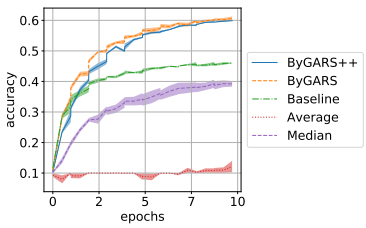


(e) 6 OFOM adversaries

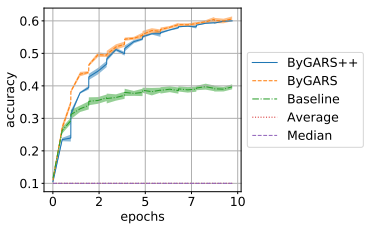


(f) 4 LIE adversaries

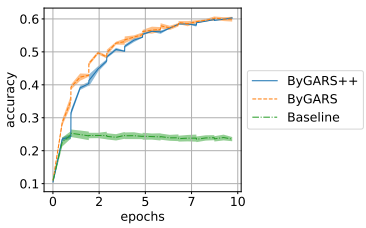
Figure 3.4: Top-1 accuracy of CIFAR-10 under random sign flip, OFOM, and LIE adversaries. Note the reduction in performance from the top row to the bottom row, due to the increase in the number of adversaries.



(a) 3 Sign flip adversaries



(b) 6 Sign flip adversaries



(c) 6 Sign flip adversaries

Figure 3.5: This figure shows the top-1 accuracy of models trained on the CIFAR-10 dataset under different Sign flip adversaries. Note that, despite the presence of 100% adversaries, we are able to recover the performance of *No Attack*. Also note that the only truthful gradients available to the baseline in (c) are from the auxiliary data, and hence the worse performance.

as the *Baseline*. Note that this is the **best** any Byzantine resilient algorithm that relies on filtering out adversarial gradients can do. Primarily for this reason, along

with other implementation-specific details required for other works that consider an arbitrary number of adversaries (Zeno requires trim parameter b that needs knowledge of the number of adversaries), we chose to compare our algorithm with *Baseline* (described above) as the gold standard. In addition to this, for illustration purposes, we also consider plain averaging of all gradients (no defense) denoted by *Average*, and coordinate-wise median [84] denoted by *Median* in our empirical analysis.

3.5.3 Distributed Setup

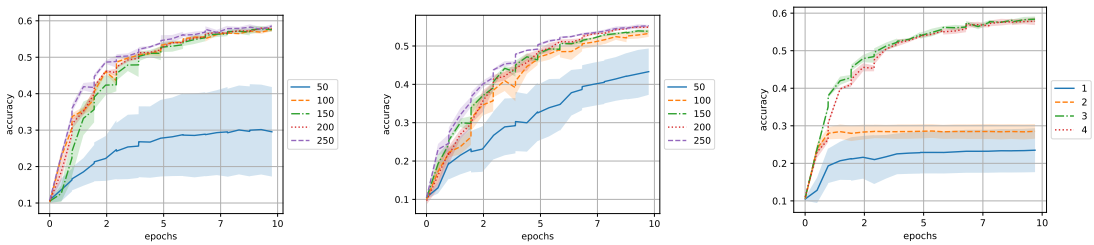
Throughout this section, we assume a setup with 1 server and 8 workers. We first compare the performance of the algorithms in the absence of any attacks (termed as *No Attack*). In order to show the robustness of our proposed algorithms to an arbitrary number of attackers, we consider a different number of attackers (3, 6, 8). We consider the case of all 8 adversaries for Sign Flipping attack. As we will see, by allowing negative reputation scores for these workers, our algorithms will achieve similar performance as that of *No Attack*.

3.5.4 Implementation Details and Hyperparameters

We scheduled the learning rate (γ_t used to update $\mathbf{w}_t$) to decay as: $\gamma_t = \gamma_0 \times \frac{1}{1+\beta t}$, and meta learning rate (α_t used to update $\mathbf{q}_t$) as : $\alpha_t = \alpha_0 \times \frac{1}{1+\beta_m t^{0.9}}$. Note that we only change the meta-update parameters for the two algorithms. We used the following hyperparameters: For CIFAR10 we used γ_0= 0.2, β= 0.9, (α_0= 0.2, $\beta_m = 0.5$ for ByGARS) and (α_0= 0.001, $\beta_m = 0.1$ for ByGARS++).

In order to avoid `NaN, Inf` values (due to adversaries), we normalize all the gradients (irrespective of adversarial or benign or auxiliary) during training. Specifically, we normalize the auxiliary gradients always to have a unit norm, for ByGARS

all worker gradients are normalized to the unit norm, for ByGARS++ all worker gradients are normalized to value 2, and for the rest (*Baseline, Average, Median*) all worker gradients are normalized to value 5. Since our objective is to illustrate the effectiveness of the proposed algorithms against the adversaries, we did not perform hyperparameter tuning to obtain the best test accuracy. Instead, we show that for appropriately chosen hyperparameters, the proposed algorithms converge and are Byzantine resilient. Unless otherwise mentioned, all the results mentioned use an auxiliary dataset of size 250, and $k = 3$ meta iterations (for ByGARS).



(a) ByGARS ablation on auxiliary dataset **(b)** ByGARS++ ablation on auxiliary dataset **(c)** ByGARS ablation on meta iterations

Figure 3.6: Ablations on the auxiliary dataset and meta iterations for 3 Label flip adversaries.

3.5.5 Results and Ablations

We present the mean and standard deviation (shared) of the results for 4 different trials for the CIFAR-10 dataset. An advantage of our proposed methods is that, for a given dataset and model, we used the same learning rate, and learning rate decay for all types of attacks, with the only difference being the meta-learning rate and meta-learning rate decay schedules for ByGARS, and ByGARS++ (Section 3.5.4). From

this, it is clear that the proposed algorithms are robust to all types of attacks discussed and do not require attack-specific information to fine-tune the hyper-parameters. Also, from Fig. 3.2 it is evident that there is no trade-off in employing our algorithm in the case of *No Attack*. This shows that our proposed algorithms can serve a much more general purpose in distributed learning applications.

We can observe from Fig. 3.3, 3.4, 3.5 of CIFAR-10 results that both ByGARS and ByGARS++ achieve Byzantine robustness against all of the threat models used under a varying number of adversaries (with the exception of LIE attack in a few cases). As expected, the median fails to defend when the fraction of adversaries is > 0.5 under all attacks. ByGARS and ByGARS++ on the other hand, do not see a lot of degradation wrt *Baseline* when the fraction of adversaries is > 0.5.

Note that, LIE attack [82] was only defined for fraction of adversaries ≤ 0.5, so we evaluated our algorithms against LIE attack with 3 attackers (< 0.5), and 4 attackers ($= 0.5$). We observe that ByGARS is robust to 3 attackers for the CIFAR-10 dataset, and suffers some degradation in performance when there are 4 attackers. ByGARS++, on the other hand, suffers some degradation on CIFAR-10 with 3 attackers and fails to defend against 4 attackers. Also, it is interesting to observe that *Median, Average* (no defense) perform reasonably well against LIE with 3 attackers. This was also observed in [82] and the authors point out that LIE is crafted to break defenses like Krum, Bulyan, etc. The authors explain that using plain averaging, the small noise added to the gradients gets averaged out and the impact on the aggregated gradient is minimal. Perhaps, ByGARS performs well against LIE due to the reputation score-based aggregation (weighted averaging) as it was observed that the reputation scores were almost equal for all the workers in this case.

We also present results when different types of attacks are carried out at the same time, called *mixed attack*. We considered one benign worker and the seven attackers include one Gaussian adversary, two Sign flip adversaries, one random sign flip adversary, two label flip adversaries, and one constant value adversary. From Fig. 3.1, we observe that the proposed algorithms can defend this *mixed attack*. To the best of our knowledge, we are the first to show robustness against different types of attacks acting at the same time without requiring separate tuning of the parameters to defend against the attacks. The fact that we used the same hyperparameters against all attacks (including the *mixed attack*) makes our algorithm more applicable to practical scenarios. Note that, the mixed attacks that include the LIE attack were not defended properly since the performance degrades even when only the LIE attack is present.

It is important to note that, one would expect the *Baseline* to be the best in all scenarios. However, we point out that by using the reputation scores, we are directly affecting the step size of each update performed, and hence it is not surprising to observe that our algorithms perform better than the *Baseline* under *No Attack* or weaker adversary models such as Sign Flip.

Additionally, in order to understand the impact of the auxiliary dataset and the meta updates on the proposed algorithms, we perform an ablation analysis. We chose the label flip attack (with 3 attackers) and fixed it for all the ablation analysis.

Size of Auxiliary Dataset

The key to the effectiveness of ByGARS and ByGARS++ is the availability of an auxiliary dataset that is drawn from the same distribution as the testing and training data. We study the dependence of the algorithms on the auxiliary dataset by repeating

experiments with different sizes of the auxiliary dataset. From Fig. 3.6a, 3.6b we can observe that a very small amount of auxiliary dataset is sufficient to face any number of adversarial workers, and the performance increases with an increase in the size of the auxiliary dataset. However, we observe that the algorithm is quite robust to the size of the auxiliary dataset once a sufficient size is reached. Also, we conclude that only a small auxiliary dataset is sufficient to efficiently compute the reputation scores (CIFAR-10 has 50,000 data points, whereas we require 250 data points for the auxiliary dataset). Note that the default size used in all other experiments is 250.

Number of meta iterations (ByGARS)

In order to study the dependence of ByGARS on the number of meta iterations, we repeated the experiments with a number of meta iterations ranging from 1 to 4 (we don't do this for ByGARS++ since there is only one meta update). From Fig. 3.6c, we can observe that the higher the number of meta iterations, the better the test accuracy. We observe that 3 meta iterations are sufficient and use them in the rest of the experiments. However, when we further increase the meta-iterations, we overfit the auxiliary data set.

3.5.6 Discussion

In this paper, we proposed two algorithms that compute the reputation scores of workers in a distributed machine-learning setup, in order to defend against any number of byzantine adversaries. However, the proposed idea of using reputation scores is much more general as it provides a way to quantify the importance of using a particular local dataset for optimizing a global model even in the non-adversarial case. We also provide more experimental results on the MNIST dataset [100] and a

synthetic dataset (strongly convex loss function) in the full version and show that similar observations hold for these two datasets.

Note that the robustness of our algorithm comes from the fact that we did not design the defense based on any particular criteria such as norm difference, or majority-based ideas. We devised the algorithm in order for the optimization to find a descent direction, and hence the superior performance across a range of attacks. However, it is important to note that in this paper, we assumed that the behavior of the attackers is stationary and this limits the ability of the attackers to adapt to the defense algorithm used on the server. With the knowledge of the defense algorithm of the server, a more intelligent adversary can employ a non-stationary attack distribution (simply turn benign when the reputation score is sufficiently negative, and flip back when the reputation score is positive), which is a more challenging problem for the server. We believe that the proposed reputation score-based aggregation provides a good platform to address this challenging setting and we consider this as a potential future direction.

Chapter 4: Finite Sample Analysis of Minmax Variant of Offline Reinforcement Learning for General MDPs

4.1 Introduction

Reinforcement learning (RL) has attracted significant interest from the research community in the last decade, inspired by the early successes of deep reinforcement learning [101, 102]. However, online RL algorithms require access to the real environment throughout training and require large datasets, which are generated through interactions with the environment. This either requires a high-fidelity simulator that mimics the environment or requires the cost of interactions with the environment to be low. These are not practical for many real-world problems. Building a high fidelity simulator for physical systems is very expensive [103, 104]. Many simulators also suffer from generalization issues due to the gap between the simulation and the real environment [105, 106]. Similarly, the cost of interactions with the environment in tasks related to healthcare and autonomous driving is very expensive and sometimes impractical [107]. Moreover, in such safety-critical applications, it is not safe to deploy semi-trained policies in real environments making online RL difficult to deploy for learning optimal policies. This has been a major hindrance to the adoption of reinforcement learning algorithms for deployment in sequential decision-making problems

[107]. One solution to this is to learn the policy from logged data which is often called Batch Reinforcement learning or Offline Reinforcement learning (ORL) [108, 107].

Fitted value iteration is a class of approximate dynamic programming algorithms that approximate the value function [109, 110, 111, 28, 112] using a finite set of data and a suitable function approximating class. The finite dataset typically comprises state, action, next state, and reward obtained over long periods of time. Fitted Q Iteration is a special case where the function approximated is the state-action value function or the Q function. Several studies have been conducted on analyzing the finite sample properties of the fitted value iteration under various settings [113, 111, 110, 114, 115]. For Fitted Value Iteration in [111], the authors assume availability for multiple data points at a given state (or state-action pair if using for ORL) enabling sufficient data coverage. However, this is a strong assumption in practical offline RL setups where the data coverage may not be sufficient. More recently [116] provided a simpler method to compute the finite sample analysis for ORL and a min-max variant algorithm, and provided sharp convergence guarantees by using specialized concentration inequalities. Moreover, [116] does not assume the availability of multiple samples as in [111]. However, [116] analysis only holds for the case when the function space is finite-dimensional. Reference [113] also studies the min-max variant algorithm for general function space and single sample path case, but assumes that the data collection policy is known (which is not always possible in real-world examples). Reference [117] and [118] studied sharp convergence guarantees of Fitted Q Iteration for general function space.

In the aforementioned analyses, the action space is assumed to be unconstrained by the state the agent is in. This does not hold true in several real-world problems where

there are physical or safety constraints that govern the permissible actions in any given state. These constraints are commonplace in robotics, economics, e-commerce, inventory management, [119, 120, 121, 122, 123], etc. Recent work by [124] addresses this by studying the asymptotic analysis of fitted Q iteration with multiple samples per state-action pair and under the state-dependent action set constraints by assuming some smoothness properties on the function approximation space and the MDP. In this work, we further generalize the work and derive the sample complexity guarantees for a min-max variant of Fitted Q Iteration for general function space. We assume the availability of an i.i.d. dataset about the state, action, reward, and next state, no knowledge of the policy used to collect the dataset, and some other mild assumptions on the MDP with denumerable state and action spaces.

4.1.1 Examples of State-Constrained MDPs

We provide here four practical applications where state and action spaces are denumerable and the permissible actions are dependent on the state of the system.

Eco-driving in Connected and Automated Vehicles (CAVs): Consider an automated vehicle that can receive future signal phase and timing information and traffic information via vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication respectively. Using this information, the CAV can optimize the vehicle speed and battery state-of-charge, which are the states of the optimal control problem aimed at minimizing the energy consumption [125, 126]. The control actions are the engine and electric motor torques which are generally computed from the non-linear engine and motor torque-speed curves respectively. Both engine and motor speed eventually depend on the vehicle speed as formulated in [127] and [128]. In this

application, the set of actions is constrained by the current state of the system, the policy used for collecting data is generally not available due to complex interactions among the subsystems, and a huge amount of offline data is available.

Vehicle rebalancing in ride-hailing systems: Consider a vehicle rebalancing problem, where vehicles are relocated to meet customers' demands. In [129], each vehicle is modeled as an agent and the state of each vehicle consists of (i) vehicle state (empty, hasPassengers, full) (ii) presence of current active requests. The action set is {pickUp, rebalance, doNothing} and it is state-dependent. E.g., pickup action can only be executed when the last passenger is dropped off or the relocation destination is reached. Offline reinforcement learning with state-dependent action constraint can be used to derive the optimal rebalancing policies for the vehicles [130].

Robotics: Robotic vehicles and manipulators in industrial settings have to navigate tight spaces and meet safety regulations. Recent works have focused on robotic safety constraints in reinforcement learning, where the agent uses some constraint barrier functions or is constrained to explore within a safe set of policies [119, 120, 121, 122], among several others. Typically, these safety constraints appear as constraints on actions (and future states) and are dependent on the current state.

Online advertisements: Consider the problem of search-based online advertising [131]. Here, a search platform displays ads relevant to queries entered by a user by allowing advertisers to bid on each query. An auto-bidding agent is an automated algorithm that determines a bid (dollar value) depending on the relevance of the user query to the advertiser's choice of bidded keywords. Whenever an ad is clicked by a

Work	Function Space	Knowledge of data collection policy	Sample Complexity
[113]	General	$\surd$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$
[116]	Finite	$\times$	$\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$
This paper	General	$\times$	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$

Table 4.1: A summary of prior works on the analysis of the min-max variant.

user, the advertiser pays some amount to the search platform that is determined by the auction mechanism. The goal of the agent is to maximize the number of ad clicks on a given day where the spending is constrained by a fixed daily budget. The agent must therefore balance between aggressively bidding for the current search query and saving budget for future search queries. We studied this problem recently in the offline RL setting where the auto-bidding agent is trained from past data of the bids [132]. The auto-bidding agent has information about the past spend (on that particular day) along with several other features about the query and the likelihood of a click. The past spend is used to determine the budget remaining for the day which is a key factor in determining the bid amount for future queries on that day. The auto-bidding agent can not bid more than the daily budget and the participation of the auto-bidding agent stops when the daily budget is depleted. In this setting, the admissible actions are constrained by the current state.

4.1.2 Notation

Let $\mathcal{X}$ be a measurable space. We use $\Delta(\mathcal{X})$ to denote the set of all probability measures on the space $\mathcal{X}$. Let $f : \mathcal{X} \to \mathbb{R}$ be a measurable function and $\mu \in \Delta(\mathcal{X})$.

We denote the (p, μ) norm of the function f as

$$\|f\|_{p,\mu} = \sqrt[p]{\int |f(x)|^p d\mu}.$$

Let X be a random variable taking values in $\mathcal{X}$ with distribution $\mu \in \Delta(\mathcal{X})$. Let $f : \mathcal{X} \to \mathbb{R}$. Define $\mathbb{V}$ as the variance

$$\mathbb{V}_{x\sim\mu}[f(X)] = \int \Big(f(x) - \mathbb{E}[f(X)]\Big)^2 d\mu \tag{4.1}$$

The set of all continuous and bounded functions $f : \mathcal{X} \to \mathbb{R}$ is denoted by $\mathcal{C}_b(\mathcal{X})$ and measurable functions $f : \mathcal{X} \to \mathbb{R}$ is denoted by $\mathcal{M}(\mathcal{X})$.

4.2 Problem Formulation

Let the MDP be defined by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the state space (which can be finite or continuous), $\mathcal{A}$ be the action space (finite or continuous). Let η_{init} be the distribution of the starting state. At a state $s \in \mathcal{S}$, the set of feasible actions is given by $\Gamma(s) \subseteq \mathcal{A}$. We use $\mathcal{B}$ to denote the feasible state-action pairs: $\mathcal{B} = \{(s, a) \in \mathcal{S} \times \mathcal{A} \mid a \in \Gamma(s)\}$. The reward function is denoted by $R : \mathcal{B} \to [0, R_{\max}]$. This is common since in most practical applications, the reward is bounded. The transition kernel of the MDP which determines the state dynamics is denoted by $P : \mathcal{B} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the set of all probability distribution s over $\mathcal{S}$. We use γ to denote the discount factor.

Let $V^\pi : \mathcal{S} \to \mathbb{R}$ denote the value function defined by

$$V^\pi(s) = \mathbb{E}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R(s_h, a_h) \Big| s_1 = s, a_h \sim \pi(\cdot|s_h)\right]$$

The goal is to learn a stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes $v^\pi := \mathbb{E}_{s\sim\eta_{\text{init}}}[V^\pi(s)]$. Let $Q^\pi : \mathcal{B} \to \mathbb{R}$ denote the state-action value function (also called Q

function) as

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R(s_h, a_h) | s_1 = s, a_1 = a, a_h \sim \pi(\cdot|s_h), h \geq 2\right].$$

It is clear that since the reward is bounded by $R_{\max}$ and due to the discount factor, we have $\|V^{\pi}\|_{\infty} \leq V_{\max} = \frac{R_{\max}}{1-\gamma}$ and $\|Q^{\pi}\|_{\infty} \leq V_{\max}$. We make the following assumptions for our analysis.

Assumption 7. *The following holds:*

1. *The set $\mathcal{B}$ is a compact subset of Euclidean space.*
2. *The reward function R is continuous.*
3. *The correspondence $\Gamma : \mathcal{S} \to \mathcal{A}$ is upper hemicontinuous.*
4. *The state transition kernel P is weakly continuous when $\Delta(\mathcal{S})$ is endowed with the usual weak topology.*

It is a common assumption to make when the state space and action spaces are denumerable; see, for example, Assumption 3.3.3 in [133, p. 28]. Under these assumptions, there exists an optimal policy π^* (see Chapter 4 [133] for more details). Let V^*, Q^* denote the corresponding value and state-action value functions. Denote by $v^* = \mathbb{E}_{s\sim\eta_{\text{init}}}[V^*(s)]$.

For a function $f \in \mathcal{C}_b(\mathcal{B})$, let $V_f(s') = \max_{a'\in\Gamma(s')} f(s', a')$. We define the Bellman operator $\mathcal{T} : \mathcal{M}(\mathcal{B}) \to \mathcal{M}(\mathcal{B})$ as

$$(\mathcal{T}f)(s,a) := R(s,a) + \gamma \mathbb{E}_{s'\sim P(s,a)}[V_f(s')].$$

The optimal Q function satisfies $Q^* = \mathcal{T}Q^*$. The goal of the agent is to design an algorithm to compute Q^* that reduces the Bellman error to 0, that is, satisfies $\|Q^* - \mathcal{T}Q^*\|_{2,\mu} = 0$.

4.2.1 Data Collection Policy and ORL Problem

The offline dataset is constructed by using a stationary policy $\pi_b : \mathcal{S} \to \Delta(\mathcal{A})$ with the environment. We refer to this as the data collection policy or the behavior policy. The past interactions with the environment using the behavior policy are logged as the dataset $D^{1:n} := (s_i, a_i, r_i, s_i')_{i=1}^n$, which we assume is independent and identically distributed. We use $\mu \in \Delta(\mathcal{B})$ to denote the stationary distribution (occupation measure of state-action pair) of the MDP under the stationary policy π_b. Therefore, it follows that (s_i, a_i) is drawn i.i.d from μ for $i \in [n]$. Note that since in practical settings, the behavior policy is not known, here we do not assume the knowledge of π_b.

Consider two function approximating classes $\mathcal{F}, \mathcal{G} \subset \{f : \mathcal{B} \to [0, V_{\max}] : f \in \mathcal{C}_b(\mathcal{B})\}$. These function classes could be neural networks, RKHS, non-parametric function approximators, etc. The ORL problem is to learn a state-action value function $f \in \mathcal{F}$ using dataset D such that the Bellman residual $\|f - \mathcal{T}f\|_{2,\mu}$ is minimized. Fitted Q iteration attempts to solve the ORL problem by using a rich function approximating class and iterative use of Bellman residual minimization on the empirical risk

$$\mathcal{L}_D(f, f') = \frac{1}{n} \sum_{i=1}^{n} \Big(f(s_i, a_i) - r_i - \gamma V_{f'}(s_i') \Big)^2.$$

We now define the operator $\widehat{\mathcal{T}}_{\mathcal{G}} : \mathcal{F} \to \mathcal{G}$ such that

$$\widehat{\mathcal{T}}_{\mathcal{G}} f = \arg\min_{g \in \mathcal{G}} \mathcal{L}_D(g, f), \text{ where } f \in \mathcal{F}. \tag{4.2}$$

Fitted Q Iteration (FQI) using the function approximating class $\mathcal{F}$ involves iteratively applying the operator $\widehat{\mathcal{T}}_{\mathcal{F}}$, i.e.,

$$f_{t+1} = \widehat{\mathcal{T}}_{\mathcal{F}} f_t.$$

Remark 1. *In some cases, the dataset $D^{1:n}$ may be very large. In this case, at iteration t, techniques can be used to create a smaller dataset $D'_t \subset D^{1:n}$, which is used for evaluating the operator in* (4.2)*. We do not analyze this setting in this paper.*

4.2.2 Key Difficulties and Solution Approach

Unlike supervised learning problems, Bellman residual minimization can not be solved using empirical risk minimization. In other words, the expectation of the empirical risk does not equal to $\|f - \mathcal{T}f\|^2_{2,\mu}$ – it over-estimates the Bellman error by a variance term as has been demonstrated in [116, 113]. To see this, let us define $\mathcal{L}_\mu(f; f') = \mathbb{E}[\mathcal{L}_D(f, f')]$ where the expectation is taken with respect to the draw of the dataset $D^{1:n}$, i.e., $(s, a) \sim \mu, s' \sim P(s, a)$. For completeness, we show in Appendix B.3 that

$$\mathcal{L}_\mu(f; f) = \|f - \mathcal{T}f\|^2_{2,\mu} + \mathbb{E}_{(s,a)\sim\mu}\Big[\mathbb{V}_{s'\sim P(s,a)}[V_f(s')]\Big].$$

One approach to addressing this is to draw two uncorrelated samples in the computation of $L_D(f, f)$ [113], i.e., for every state-action pair, two next states should be sampled according to $P(s, a)$. However, this assumption is not practical in the continuous-state continuous-action ORL setting since we can not guarantee that multiple next states can be sampled for a given state-action pair or that the same state-action pair is visited twice while collecting the dataset.

Another approach followed is to estimate the variance and subtract it from the empirical objective, and this approach results in the following min-max formulation of the offline RL algorithm [116]

$$\widehat{f} := \arg\inf_{f\in\mathcal{F}} \sup_{g\in\mathcal{G}} \mathcal{L}_D(f; f) - \mathcal{L}_D(g; f) \tag{4.3}$$

where $\mathcal{G} \subset \{g : \mathcal{B} \to [0, V_{\max}] | g \in \mathcal{C}_b(\mathcal{B})\}$ is another rich function class that is continuous in the actions. In this work, we study the finite sample complexity of this algorithm in the general state space and general function space setting.

Reference [116] study this algorithm under the finite state space, finite action space, and when the function space is finite. The finite function space assumption allows them to use a simple union bound argument along with the Bernstein inequality to get the sharp sample complexity. In this paper, we assume a general function space and use a covering number argument to achieve the sample complexity bound. Reference [113] study a similar algorithm, however, the algorithm there requires the knowledge of the behavior policy (another difference being that they study the case where the data is generated from a single sample path and for finite action space). In particular, their objective is

$$\arg \inf_{f \in \mathcal{F}} \sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\pi_b(a_i|s_i)} \Big[\Big(f(s_i, a_i) - r_i - \gamma V_{f'}(s_i') \Big)^2 - \Big(g(s_i, a_i) - r_i - \gamma V_{f'}(s_i') \Big)^2 \Big]$$

where π_b is the behavior policy used to collect the single sample path data. In contrast, here we analyze the algorithm when such knowledge of the data collection policy is unknown. For a general state space and general action space, estimating the data collection policy from the finite data leads to high variance estimates, subsequently affecting the fitted learning objective.

4.2.3 Preliminaries

We now introduce the following two quantities that capture the strength of the function approximation spaces $\mathcal{F}$ and $\mathcal{G}$:

$$\epsilon_{\mathcal{F}} = \inf_{f \in \mathcal{F}} \|f - \mathcal{T} f\|_{2,\mu}^2, \quad \epsilon_{\mathcal{F},\mathcal{G}} = \sup_{f \in \mathcal{F}} \inf_{g \in \mathcal{G}} \|g - \mathcal{T} f\|_{2,\mu}^2.$$

If Q^* is realizable in $\mathcal{F}$, i.e., $Q^* \in \mathcal{F}$, then $\epsilon_{\mathcal{F}} = 0$. When $\mathcal{F} = \mathcal{G}$, $\epsilon_{\mathcal{F},\mathcal{G}}$ is called the inherent Bellman error. We provide the finite sample guarantees based on $\epsilon_{\mathcal{F}}$ and $\epsilon_{\mathcal{F},\mathcal{G}}$, therefore it is inherently assumed that these quantities are small in order to control the bounds.

A distribution $\nu \in \Delta(\mathcal{B})$ is admissible in MDP, if there exists $h \geq 1$, and a potentially non-stationary and stochastic policy $\pi := (\pi_1, \pi_2, \ldots)$ such that

$$\nu(ds, da) = \mathbb{P}[s_h \in ds, a_t \in da | s_1 \sim \eta_{\text{init}}, a_t \sim \pi_t(\cdot|s_t)]$$

We denote $s' \sim P(\nu)$ as a shorthand for $(s, a) \sim \nu, s' \sim P(s, a)$. Also, we define $\pi_{f,f'}(s)$ as

$$\pi_{f,f'}(s) = \arg \max_{a \in \Gamma(s)} \max\{f(s, a), f'(s, a)\}. \tag{4.4}$$

For every given $f \in \mathcal{F}$, denote $g_f^* = \arg\min_{g \in \mathcal{G}} \|g - \mathcal{T}f\|_{2,\mu}$ and observe that $\|g_f^* - \mathcal{T}f\|_{2,\mu}^2 \leq \epsilon_{\mathcal{F},\mathcal{G}}$.

Definition 2. *Define the class of functions* $Z_{\mathcal{F}} = \{Z_f : \mathcal{B} \times \mathbb{R} \times \mathcal{S} \to \mathbb{R} \,|f \in \mathcal{F}\}$ *such that*

$$\begin{aligned} &Z_f(s, a, r, s') := \\ &\quad \big(f(s, a) - r - \gamma V_f(s')\big)^2 - \big((\mathcal{T}f)(s, a) - r - \gamma V_f(s')\big)^2 \end{aligned} \tag{4.5}$$

Definition 3. *Define a function class* $X_{\mathcal{F}} : \{X_{g,f,g_f^*} : \mathcal{B} \times \mathbb{R} \times \mathcal{S} \to \mathbb{R} \mid f, g \in \mathcal{F}\}$ *where*

$$\begin{aligned} &X_{g,f,g_f^*}(s, a, r, s') \\ &:= \big(g(s, a) - r - \gamma V_f(s')\big)^2 - \big(g_f^*(s, a) - r - \gamma V_f(s')\big)^2. \end{aligned} \tag{4.6}$$

Definition 4. *Define the function class* $Y_{\mathcal{F},\mathcal{G}} : \{Y_{g,f} : \mathcal{B} \times \mathbb{R} \times \mathcal{S} \to \mathbb{R} \mid g \in \mathcal{G}, f \in \mathcal{F}\}$ *such that*

$$\begin{aligned} &Y_{g,f}(s, a, r, s') \\ &= \big(g(s, a) - r - \gamma V_f(s')\big)^2 - \big((\mathcal{T}f)(s, a) - r - \gamma V_f(s')\big)^2. \end{aligned} \tag{4.7}$$

We use $\mathtt{d}_{X_{\mathcal{F}}}$, $\mathtt{d}_{Z_{\mathcal{F}}}$ and $\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}$ to be the pseudo-dimension of the function classes described above. These notations are introduced in Definition 11 (Appendix B.2).

4.3 Assumptions and Main Results

Our main assumption on the MDP is that we assume the existence of a finite concentrability coefficient from [116] (for the case of finite state and finite action space). We now state the assumption adapted to the current setup of state-dependent action sets.

Assumption 8 (Concentrability coefficient)**.** *For all admissible $\nu \in \Delta(\mathcal{B})$, we assume that $C < \infty$ such that $\|\frac{d\nu}{d\mu}\|_\infty \leq C$.*

The above assumption implies that the transitions are sufficiently stochastic and $\nu(\cdot,\cdot) \leq C\mu(\cdot,\cdot)$, $\forall\ (s,a) \in \mathcal{B}$. Note that this assumption is much stronger than the usual discounted average concentrability of future states [111].

We next assume the finiteness of the capacity of the function approximation class since our sample complexity bounds depend on the function class capacity.

Assumption 9 (Finite capacity of function classes)**.** *The pseudo-dimensions $\mathtt{d}_{X_{\mathcal{F}}}$, $\mathtt{d}_{Z_{\mathcal{F}}}$ and $\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}$ are all assumed to be finite.*

Remark 2. *A sufficient condition that ensures that the function classes have finite capacity is discussed in [134, 124]. In [134], the author shows that the optimal value/ Q function (under state-dependent action constraints) is Lipschitz continuous under the following assumptions: the transition function is Lipschitz continuous in (s,a), the reward function is Lipschitz in (s,a), the correspondence Γ is Lipschitz continuous under the Hausdorff metric, and the Bellman operator is a contraction. In addition,*

if we assume $\mathcal{F}$ and $\mathcal{G}$ are Lipschitz continuous function classes and Γ is Lipschitz continuous correspondence, then it can be shown that $Z_{\mathcal{F}}, X_{\mathcal{F}}, Y_{\mathcal{F},\mathcal{G}}$ are also Lipschitz continuous using Lemma 3.2 in [134]. The finite capacity of the Lipschitz and uniformly bounded function class follows from Theorem 2.7.1 and 2.7.11 of [135].

We now state the finite sample analysis result of the offline RL algorithm (4.3).

Theorem 6 (Error bound for min-max)**.** *Suppose Assumptions 7, 8, and 9 hold. Given a dataset $D = \{(s_i, a_i, r_i, s_i')\}_{i=1}^n$, two classes of bounded functions $\mathcal{F}$, $\mathcal{G}$ and $\epsilon, \delta > 0$, then with probability at least $1-\delta$, the output policy of (4.3), $\pi_{\widehat{f}}$ satisfies*

$$v^* - v^{\pi_{\widehat{f}}} \leq \frac{2\sqrt{C}}{(1-\gamma)^2}\Big(\sqrt{\epsilon + \epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}}}\Big) \tag{4.8}$$

when

$$\begin{aligned} n \geq \frac{K_1 V_{\max}^4}{\epsilon^2}\Bigg[\log\frac{16e}{\delta} + \log\Bigg(&2(\mathtt{d}_{X_{\mathcal{F}}}+1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{X_{\mathcal{F}}}} \\ &+ (\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}+1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}} \\ &+ (\mathtt{d}_{Z_{\mathcal{F}}}+1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{Z_{\mathcal{F}}}}\Bigg)\Bigg], \end{aligned}$$

where $\mathtt{d}_{X_{\mathcal{F}}}, \mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}, \mathtt{d}_{Z_{\mathcal{F}}}$ are the pseudo-dimensions of the spaces $X_{\mathcal{F}}, Y_{\mathcal{F},\mathcal{G}}, Z_{\mathcal{F}}$ respectively, and $K_1 = 64 \times 128 \times 36$ and $K_2 = 6 \times 64$.

Remark 3. *The sample complexity does not get affected by arbitrary re-scaling of the reward function as long as ϵ is scaled by the square of the scaling factor for the reward, and each function in the function classes $\mathcal{F}$ and $\mathcal{G}$ also get rescaled by the same factor. Observe that when we rescale the reward function R by $p > 0$ and ϵ by p^2, (i) $V_{\max}$, v^*, and $v^{\pi_{\widehat{f}}}$ also get rescaled by p, (ii) $\frac{V_{\max}^2}{\epsilon}$ does not get rescaled, and (iii) the $\epsilon_{\mathcal{F}}$ and $\epsilon_{\mathcal{F},\mathcal{G}}$ terms get rescaled as by p^2. Thus, if we scale ϵ to $p^2\epsilon$, then the*

scaling term appears on the sides of the inequality in (4.8) *and the lower bound on* n *remains the same. Consequently, the sample complexity result remains unchanged.*

Dependence on function class We can observe that, the error depends on $\epsilon_{\mathcal{F}}, \epsilon_{\mathcal{F},\mathcal{G}}$. When the function class considered is sufficiently rich (such as a neural network class or RKHS), we can assume that $Q^* \in \mathcal{F}$ and $\mathcal{T}f \in \mathcal{F}$, which results in $\epsilon_{\mathcal{F}} = 0$ and $\epsilon_{\mathcal{F},\mathcal{G}} = 0$.

When $\mathcal{F} = \mathcal{G}$, observe that the function classes $Z_{\mathcal{F}}$ and $Y_{\mathcal{F},\mathcal{G}}$ are the same, i.e. $(\mathtt{d}_{Z_{\mathcal{F}}} = \mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}})$. In addition, when the function class is sufficiently rich where $\epsilon_{\mathcal{F},\mathcal{G}} = 0$, then $g_f^* = \mathcal{T}f$ which implies that the function class $X_{\mathcal{F}}$ is equal to $Z_{\mathcal{F}}$ and $Y_{\mathcal{F},\mathcal{G}}$. The result then is simplified as follows: The following holds with probability at least $1 - \delta$,

$$v^* - v^{\pi_{\hat{f}}} \leq \frac{2\sqrt{C}}{(1-\gamma)^2}\Big(\sqrt{\epsilon}\Big)$$

when

$$n \geq \frac{K_1 V_{\max}^4}{\epsilon^2}\left[\log\frac{64e(\mathtt{d}_{X_{\mathcal{F}}}+1)}{\delta} + \mathtt{d}_{X_{\mathcal{F}}}\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)\right].$$

Dependence on ϵ Ignoring log terms in the sample complexity bound, we observe that

$$v^* - v^{\pi_{\hat{f}}} \leq \mathcal{O}(\sqrt{\epsilon})$$

when $n \geq \mathcal{O}(\frac{1}{\epsilon^2})$. This shows that, we can achieve an error of ϵ by using approximately $\mathcal{O}(\frac{1}{\epsilon^4})$ data samples. This result is consistent with the earlier results [113].

4.4 Proof of Theorem 6

In this section, we prove the main result of the paper. We follow the analysis of [116], however, unlike [116] the focus is not to obtain fast rates of convergence

($\frac{1}{\epsilon^2}$ dependence instead of $\frac{1}{\epsilon^4}$), but to obtain convergence rates for the general state space and state-dependent action space setting. Although challenging, with some additional effort, it may be possible to obtain the fast rates such as those in [116], [117] using specialized concentration inequalities (see Section 4.5).

4.4.1 Proof Outline

It is evident that $v^* - v^{\pi_{\widehat{f}}}$ is related to $\mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f})$. Thus, we need to bound $\mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f})$ in terms of its empirical counterpart $\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f};\widehat{f})$ using concentration inequalities. Accordingly, we first derive a decomposition of the empirical term $\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f};\widehat{f})$ into three terms I, II, III and bound each of these terms separately. We state the decomposition lemma below.

Lemma 4 (Decomposition Lemma)**.** *For $f^* \in \mathcal{F}$ s.t, $\|f^* - \mathcal{T}f^*\|_{2,\mu}^2 \leq \epsilon_{\mathcal{F}}$, we have*

$$\begin{aligned}\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f}) \leq & \underbrace{\mathcal{L}_D(f^*;f^*) - \mathcal{L}_D(\mathcal{T}f^*,f^*)}_{I} \\ & + \underbrace{|\mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f}) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}\widehat{f},\widehat{f})|}_{II} \\ & + \underbrace{|\mathcal{L}_D(\mathcal{T}f^*,f^*) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f^*;f^*)|}_{III}\end{aligned} \tag{4.9}$$

Proof. This result is established in Appendix B.1. □

In what follows, we divide the proof of Theorem 6 into three steps. We derive an upper bound on $v^* - v^{\pi_{\widehat{f}}}$ as a function of $\mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f})$ in Subsection 4.4.2. We then bound the three terms noted in Subsection 4.4.4 using the concentration of measures results derived in Subsection 4.4.3. Finally, we prove Theorem 6 in Subsection 4.4.5.

4.4.2 Relation between $v^* - v^{\pi_{\hat{f}}}$ and $\mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f})$

We shall first show the relation between, $v^* - v^{\pi_{\hat{f}}}$ and $\mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f})$, that will be used to prove the main theorem.

Lemma 5. *The following holds true,*

1. *Let ν be any admissible distribution. Then $\forall\ f : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$,*

$$\|f\|_{2,\nu} \le \sqrt{C}\|f\|_{2,\mu}.$$

2. *We denote $\eta_h^\pi := \mathbb{P}[s_h = s|s_1 \sim \eta_1, \pi]$, and π_f as the policy greedy with respect to the state-action value function $f : \mathcal{B}\to\mathbb{R}$, i.e., $\pi_f(s) := \arg\max_{a\in\Gamma(s)} f(s,a)$.*

   Then we have

$$v^* - v^{\pi_f} \le \sum_{h=1}^{\infty}\gamma^{h-1}\Big(\|Q^* - f\|_{2,\eta_h^{\pi_f}\times\pi^*} + \|Q^* - f\|_{2,\eta_h^{\pi_f}\times\pi_f}\Big).$$

3. *Let $f, f' : \mathcal{B}\to\mathbb{R}$. Then we have $\forall\ \nu\in\Delta(\mathcal{B})$,*

$$\|V_f - V_{f'}\|_{2,P(\nu)} \le \|f - f'\|_{2,P(\nu)\times\pi_{f,f'}}.$$

4. *For an exploratory distribution $\mu\in\Delta(\mathcal{B})$, any distribution $\nu\in\Delta(\mathcal{B})$, policy π, and $f, f' : \mathcal{B}\to\mathbb{R}$, we have*

$$\|f - Q^*\|_{2,\nu} \le \frac{\sqrt{C}}{1-\gamma}\|f - \mathcal{T}f\|_{2,\mu}.$$

Proof. The results can be adapted directly from [116] to the general state space setting in this paper using Assumptions 7 and 8. □

Lemma 6. *For $f, g \in \mathcal{F}$, we have $\|g - \mathcal{T}f\|_{2,\mu}^2 = \mathcal{L}_\mu(g; f) - \mathcal{L}_\mu(\mathcal{T}f; f)$.*

Proof. From the definitions in $\mathcal{L}_\mu$ and $\mathcal{L}_D$, we have

$$\begin{aligned}
\mathcal{L}_\mu(g; f) - \mathcal{L}_\mu(\mathcal{T}f; f) &= \mathbb{E}\Big[\mathcal{L}_D(g, f) - \mathcal{L}_D(\mathcal{T}f; f)\Big] \\
&= \mathop{\mathbb{E}}_{\substack{(s,a)\sim\mu,\\ s'\sim P(s,a)}} \Big[(g(s,a) - r - \gamma V_f(s'))^2 - (\mathcal{T}f(s,a) - r - \gamma V_f(s'))^2\Big] \\
&= \mathop{\mathbb{E}}_{\substack{(s,a)\sim\mu,\\ s'\sim P(s,a)}} \Big[(g(s,a)^2 - \mathcal{T}f(s,a)^2) \\
&\qquad + 2(r + \gamma V_f(s'))(\mathcal{T}f(s,a) - g(s,a))\Big] \\
&\overset{i}{=} \mathop{\mathbb{E}}_{(s,a)\sim\mu} \Big[(g(s,a)^2 - \mathcal{T}f(s,a)^2) + 2\mathcal{T}f(s,a)(\mathcal{T}f(s,a) - g(s,a))\Big] \\
&= \mathop{\mathbb{E}}_{(s,a)\sim\mu,} \Big[(g(s,a) - \mathcal{T}f(s,a))^2\Big] \\
&= \|g - \mathcal{T}f\|_{2,\mu}^2
\end{aligned}$$

where (i) follows from the definition of the operator $\mathcal{T}$. The proof is complete. □

Lemma 7.

$$v^* - v^{\pi_{\widehat{f}}} \leq \frac{2\sqrt{C}}{(1-\gamma)^2}\left(\sqrt{\mathcal{L}_\mu(\widehat{f}; \widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f}, \widehat{f})}\right). \tag{4.10}$$

Proof. Substituting $f = \widehat{f}$ in the result of Lemma 4, we get

$$\|\widehat{f} - Q^*\|_{2,\nu} \leq \frac{\sqrt{C}}{1-\gamma}\|\widehat{f} - \mathcal{T}\widehat{f}\|_{2,\mu}.$$

The proof then follows by applying Lemmas 5 and 6 to the above equation. □

From the above Lemma, we observe that it is sufficient to bound $\mathcal{L}_\mu(\widehat{f}, \widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f}, \widehat{f})$ to prove the main theorem.

4.4.3 Using Concentration Inequality

Recall the definition of $Z_{\mathcal{F}}$ from Definition 2. It is straight forward that the class of functions $Z_{\mathcal{F}}$ is bounded, and using the bounds on $R(\cdot,\cdot)$ and $\mathcal{F}$, we can observe that $\forall f \in \mathcal{F}$, $|Z_f(\cdot,\cdot,\cdot,\cdot)| \leq 3V_{\max}^2$. For $f \in \mathcal{F}$ and $(s_i, a_i, r_i, s_i') \in D$, we can denote an i.i.d random variable $Z_f^i := Z_f(s_i, a_i, r_i, s_i')$. Now, we can observe from Lemma 6 that

$$\frac{1}{n}\sum_{i=1}^{n} Z_{\widehat{f}}^i = \mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f});$$
$$\mathbb{E}[Z_{\widehat{f}}^1] = \mathcal{L}_\mu(\widehat{f};\widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f};\widehat{f}) = \|\widehat{f} - \mathcal{T}\widehat{f}\|_{2,\mu}^2.$$

Reference [116] assume a finite function space $\mathcal{F}$ and proceed to bound $\mathbb{E}[Z_{\widehat{f}}^1] - \frac{1}{n}\sum_{i=1}^{n} Z_{\widehat{f}}^i$ using Bernstein's inequality and apply a union bound over the function space $\mathcal{F}$ (consequently the bound depends on $|\mathcal{F}|$). However, we can not do this for a general function space. Instead, we use the Pollard's concentration inequality (Lemma 19) to bound $\mathbb{E}[Z_{\widehat{f}}^1] - \frac{1}{n}\sum_{i=1}^{n} Z_{\widehat{f}}^i$.

Lemma 8. *With probability at least $1-\delta_1$, we have*

$$\mathbb{E}[Z_{\widehat{f}}^1] \leq \frac{1}{n}\sum_{i=1}^{n} Z_{\widehat{f}}^i + \epsilon/8,$$

where $\delta_1 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, Z_{\mathcal{F}}, D^{1:n})] \exp\left(\frac{-n\epsilon^2}{64\times 128\times 36\ V_{\max}^4}\right)$.

Proof. We can directly apply Lemma 19 (along with Remark 4) on the class of functions $Z_{\mathcal{F}}$ with $B = 6V_{\max}^2$ and $\epsilon/8$ instead of ϵ. We get

$$\mathbb{P}\left\{\sup_{f\in\mathcal{F}}\left(\mathbb{E}[Z_f^1] - \frac{1}{n}\sum_{i=1}^{n} Z_f^i\right) > \epsilon/8\right\}$$
$$\leq 8\mathbb{E}[N_1(\epsilon/64, Z_{\mathcal{F}}, D^{1:n})] \exp\left(\frac{-n\epsilon^2}{64\times 128\times 36\ V_{\max}^4}\right)$$

Since the above inequality holds for all $f \in \mathcal{F}$, it certainly holds for a given $\hat{f} \in \mathcal{F}$. Note that, every $Z_f \in Z_{\mathcal{F}}$ is defined by a $f \in \mathcal{F}$. The result follows by taking the value of δ_1 equal to the right-hand side of the above equation. □

We now bounded $\mathcal{L}_\mu(\widehat{f}, \widehat{f}) - \mathcal{L}_\mu(\mathcal{T}\widehat{f}, \widehat{f})$ in terms of $\mathcal{L}_D(\widehat{f}; \widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f}, \widehat{f})$ by using Pollard's concentration inequality. In the next subsections, we will continue to bound $\mathcal{L}_D(\widehat{f}; \widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f}, \widehat{f})$ using the Lemma 4 and repeated use of the Pollard's concentration inequality.

4.4.4 Bounding terms I, II, III

In this section, we bound the terms in Lemma 4.

Lemma 9 (Term I in (4.9))**.** *With probability at least* $1 - \delta_1$*, we have*

$$\mathcal{L}_D(f^*; f^*) - \mathcal{L}_D(\mathcal{T}f^*; f^*) = \frac{1}{n}\sum_{i=1}^{n} Z^i_{f^*} \leq \frac{\epsilon}{8} + \epsilon_{\mathcal{F}} \tag{4.11}$$

where $\delta_1 = 8N_1(\frac{\epsilon}{64}, Z_{\mathcal{F}}, D^{1:n}) \exp\left(\frac{-n\epsilon^2}{64\times 128\times 36\ V^4_{\max}}\right)$.

Proof. From Lemma 6, $\mathbb{E}[Z^1_{f^*}] = \mathcal{L}_\mu(f^*; f^*) - \mathcal{L}_\mu(\mathcal{T}f^*; f^*) = \|f^* - \mathcal{T}f^*\|^2_{2,\mu} \leq \epsilon_{\mathcal{F}}$. Therefore, applying Lemma 19 (along with Remark 4) and since $\mathbb{E}[Z^1_{f^*}] \leq \epsilon_{\mathcal{F}}$,

$$\begin{aligned}
&\mathbb{P}\left\{\frac{1}{n}\sum_{i=1}^{n} Z^i_{f^*} > \frac{\epsilon}{8} + \epsilon_{\mathcal{F}}\right\} \\
&\leq 8\mathbb{E}[N_1(\frac{\epsilon}{64}, Z_{\mathcal{F}}, D^{1:n})] \exp\left(\frac{-n\epsilon^2}{64 \times 128 \times 36\ V^4_{\max}}\right).
\end{aligned}$$

The result follows similarly by taking the value of δ_1 as the right-hand side of the above equation. This bounds term (I) in (4.9). □

We are left to compute bounds on the terms II, III. Observe that both the terms II, III in Lemma 4 are of the form $|\mathcal{L}_D(\mathcal{T}f, f) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f; f)|$ where term II

considers $\widehat{f}$ and term III considers f^*. Therefore, in the following Lemma, we want to bound for any $f \in \mathcal{F}$,

$$|\mathcal{L}_D(\mathcal{T}f, f) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f; f)|.$$

This can then be used to bound each of the terms II and III. Before stating the Lemma, we first discuss the function classes $X_{\mathcal{F}}$ and $Y_{\mathcal{F},\mathcal{G}}$.

Recall the definition of $X_{\mathcal{F}}$ from Definition 3. Similar to $Z_{\mathcal{F}}$, we can show that $X_{\mathcal{F}}$ is a bounded class of functions, and $\forall f, g \in \mathcal{F}$, $|X_{g,f,g_f^*}(\cdot,\cdot,\cdot,\cdot)| \leq 3V_{\max}^2$. For each $(s_i, a_i, r_i, s_i') \in D$, we denote an i.i.d random variable $X_{g,f,g_f^*}^i := X_{g,f,g_f^*}(s_i, a_i, r_i, s_i')$. Now, from the definition of $\mathcal{L}_D$ and $\mathcal{L}_\mu$,

$$\frac{1}{n}\sum_i^n X_{g,f,g_f^*}^i = \mathcal{L}_D(g; f) - \mathcal{L}_D(g_f^*, f);$$
$$\mathbb{E}[X_{g,f,g_f^*}^1] = \mathcal{L}_\mu(g; f) - \mathcal{L}_\mu(g_f^*, f).$$

Recall the definition of $Y_{\mathcal{F},\mathcal{G}}$ from Definition 4. Similarly, we can again show that $Y_{\mathcal{F},\mathcal{G}}$ is a bounded class of functions, and $\forall f \in \mathcal{F}, g \in \mathcal{G}$, $|Y_{g,f}(\cdot,\cdot,\cdot,\cdot)| \leq 3V_{\max}^2$. For each $(s_i, a_i, r_i, s_i') \in D$, we denote an i.i.d random variable $Y_{g,f}^i := Y_{g,f}(s_i, a_i, r_i, s_i')$. Now, from the definition of $\mathcal{L}_D$ and $\mathcal{L}_\mu$,

$$\frac{1}{n}\sum_i^n Y_{g,f}^i = \mathcal{L}_D(g; f) - \mathcal{L}_D(\mathcal{T}f, f);$$
$$\mathbb{E}[Y_{g,f}^1] = \mathcal{L}_\mu(g; f) - \mathcal{L}_\mu(\mathcal{T}f, f).$$

We are now ready for the next Lemma.

Lemma 10 (Terms II, III in (4.9))**.** *With probability at least* $1 - 2\delta_2 - \delta_3$*, we have*

$$|\mathcal{L}_D(\mathcal{T}f; f) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f; f)| \leq \epsilon_{\mathcal{F},\mathcal{G}} + \frac{3\epsilon}{8}$$

where $\delta_2 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, X_{\mathcal{F}}, D^{1:n}))] \exp\left(\frac{-n\epsilon^2}{64\times128\times36\ V_{\max}^4}\right)$ *and*
$\delta_3 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, Y_{\mathcal{F},\mathcal{G}}, D^{1:n}))] \exp\left(\frac{-n\epsilon^2}{64\times128\times36\ V_{\max}^4}\right)$.

Proof. Observe that

$$\begin{aligned}|\mathcal{L}_D(\mathcal{T}f;f) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f;f)| &= \Big|\frac{1}{n}\sum_{i=1}^{n}\left(X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f} + Y^i_{g^*_f,f}\right)\Big| \\ &\leq \Big|\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}\Big| + \Big|\sum_{i=1}^{n} Y^i_{g^*_f,f}\Big|\end{aligned} \tag{4.12}$$

Using Lemma 19 for the function space $X_{\mathcal{F}}$ with $B = 6V_{\max}^2$ and $\epsilon/8$ instead of ϵ, we get

$$\begin{aligned}&\mathbb{P}\left\{\sup_{f,g\in\mathcal{F}}\Big|\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f} - \mathbb{E}[X^1_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}]\Big| > \frac{\epsilon}{8}\right\} \\ &\leq 8\mathbb{E}[N_1(\epsilon/64, X_{\mathcal{F}}, D^{1:n}))]\exp\Big(\frac{-n\epsilon^2}{64\times 128\times 36\ V_{\max}^4}\Big).\end{aligned}$$

Observe that for $a, b \in \mathbb{R}$, $|a| - |b| \leq |a - b|$. From this (and using the same argument as in Remark 4), we get with probability greater than $1 - \delta_2$,

$$\Big|\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}\Big| \leq \Big|\mathbb{E}[X^1_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}]\Big| + \frac{\epsilon}{8} \tag{4.13}$$

where $\delta_2 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, X_{\mathcal{F}}, D^{1:n}))]\exp\Big(\frac{-n\epsilon^2}{64\times 128\times 36\ V_{\max}^4}\Big)$.

Now, observe that $\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f} \leq 0$, since $\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f} = \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f, f) - \mathcal{L}_D(g^*_f, f) \leq 0$ where the inequality follows due to the optimality of $\widehat{\mathcal{T}}_{\mathcal{G}}f$ given dataset D. Therefore, applying Lemma 19 to the function space $X_{\mathcal{F}}$ again we get,

$$\Big|\mathbb{E}[X^1_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}]\Big| \leq \frac{\epsilon}{8} \tag{4.14}$$

with probability at least $1 - \delta_2$.

Now by combining (4.13) and (4.14), we have with probability at least $1 - 2\delta_2$,

$$\Big|\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}\Big| \leq \epsilon/4$$

where $\delta_2 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, X_{\mathcal{F}}, D^{1:n}))]\exp\Big(\frac{-n\epsilon^2}{64\times 128\times 36\ V_{\max}^4}\Big)$.

Now, we bound the second term $\left|\frac{1}{n}\sum_{i=1}^{n} Y^i_{g^*_f,f}\right|$ in (4.12). We have, $\mathbb{E}[Y^i_{g^*_f,f}] = \|g^*_f - \mathcal{T}f\|^2_{2,\mu} \leq \epsilon_{\mathcal{F},\mathcal{G}}$.

Applying Lemma 19 to the function space $Y_{\mathcal{F},\mathcal{G}}$ similarly as above, we have with probability at least $1-\delta_3$,

$$\left|\frac{1}{n}\sum_{i=1}^{n} Y^i_{g^*_f,f}\right| \leq \mathbb{E}[Y^1_{g^*_f,f}] + \frac{\epsilon}{8} \leq \epsilon_{\mathcal{F},\mathcal{G}} + \frac{\epsilon}{8}$$

for $\delta_3 = 8\mathbb{E}[N_1(\frac{\epsilon}{64}, Y_{\mathcal{F},\mathcal{G}}, D^{1:n}))]\exp\left(\frac{-n\epsilon^2}{64\times128\times36\ V^4_{\max}}\right)$. The result follows by combining the bounds on $\left|\frac{1}{n}\sum_{i=1}^{n} X^i_{\widehat{\mathcal{T}}_{\mathcal{G}}f,f,g^*_f}\right|$ and $\left|\sum_{i=1}^{n} Y^i_{g^*_f,f}\right|$. □

Lemma 11. *With probability at least* $1-\delta_1-4\delta_2-2\delta_3$,

$$\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f}) \leq \epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}} + \frac{7\epsilon}{8} \tag{4.15}$$

where δ_1 *is as defined in Lemma 9;* δ_2 *and* δ_3 *are defined as in Lemma 10.*

Proof. Let us recall that, both the terms II and III can be bounded using Lemma 10. The result follows from Lemmas 4, 9 and 10. Note that we apply Lemma 10 twice, therefore the coefficients of δ_2 and δ_3 are multiplied by 2. □

4.4.5 Proof of Theorem 6

We now have all the intermediate results to prove the main theorem. From Lemmas 4, 8, and 11, with probability at least $1-2\delta_1-4\delta_2-2\delta_3$,

$$\mathbb{E}[Z^1_{\widehat{f}}] \leq \epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}} + \epsilon.$$

Substituting this result in (4.10), with $K_1 = 64 \times 128 \times 36$, we get

$$\begin{aligned}
&\mathbb{P}\Big\{v^* - v^{\pi_{\hat{f}}} > \frac{2\sqrt{C}}{(1-\gamma)^2}\sqrt{\epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}} + \epsilon}\Big\} \\
&\leq \exp\Big(\frac{-n\epsilon^2}{K_1 V_{\max}^4}\Big)\Big(16\mathbb{E}[N_1(\epsilon/64, Z_{\mathcal{F}}, D^{1:n})] \\
&\qquad\qquad + 32\mathbb{E}[N_1(\epsilon/64, X_{\mathcal{F}}, D^{1:n})] \\
&\qquad\qquad + 16\mathbb{E}[N_1(\epsilon/64, Y_{\mathcal{F},\mathcal{G}}, D^{1:n})]\Big).
\end{aligned}$$

We then apply Lemma 18 in the Appendix and let $K_2 = 6 \times 64$, we get

$$\begin{aligned}
&\mathbb{P}\Big\{v^* - v^{\pi_{\hat{f}}} > \frac{2\sqrt{C}}{(1-\gamma)^2}\sqrt{\epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}} + \epsilon}\Big\} \\
&\leq \exp\Big(\frac{-n\epsilon^2}{K_1 V_{\max}^4}\Big)\Bigg(16e(d_{Z_{\mathcal{F}}} + 1)\Big(\frac{K_2\ eV_{\max}^2}{\epsilon}\Big)^{d_{Z_{\mathcal{F}}}} \\
&\qquad\qquad + 32e(d_{X_{\mathcal{F}}} + 1)\Big(\frac{K_2\ eV_{\max}^2}{\epsilon}\Big)^{d_{X_{\mathcal{F}}}} \\
&\qquad\qquad + 16e(d_{Y_{\mathcal{F},\mathcal{G}}} + 1)\Big(\frac{K_2\ eV_{\max}^2}{\epsilon}\Big)^{d_{Y_{\mathcal{F},\mathcal{G}}}}\Bigg).
\end{aligned}$$

Now consider the right-hand side term to be δ, applying log and rearranging the terms, we get with probability at least $1 - \delta$,

$$v^* - v^{\pi_{\hat{f}}} \leq \frac{2\sqrt{C}}{(1-\gamma)^2}\Big(\sqrt{\epsilon + \epsilon_{\mathcal{F}} + \epsilon_{\mathcal{F},\mathcal{G}}}\Big)$$

when

$$\begin{aligned}
n \geq \frac{K_1 V_{\max}^4}{\epsilon^2}\Bigg[\log\frac{16e}{\delta} + \log\Bigg(&2(\mathtt{d}_{X_{\mathcal{F}}} + 1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{X_{\mathcal{F}}}} \\
&+ (\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}} + 1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{Y_{\mathcal{F},\mathcal{G}}}} \\
&+ (\mathtt{d}_{Z_{\mathcal{F}}} + 1)\Big(\frac{K_2 e V_{\max}^2}{\epsilon}\Big)^{\mathtt{d}_{Z_{\mathcal{F}}}}\Bigg)\Bigg].
\end{aligned}$$

The proof is complete.

4.5 Discussion

4.5.1 Experimental Results

In this section, we perform numerical simulations to study how the sample complexity materializes in practice. We adopt the *optimal charging schedule for a battery pack* example from [124]. Here, the battery pack is used to serve some random user demands and is charged using a random renewable source. The maximum capacity of a battery pack is given by a real value $B \in \mathbb{R}^+$. The state of charge of the battery pack is denoted by $G_t \in [0, 1]$ which is the fraction of charge as compared to the maximum capacity. We represent the net generation at time t (renewable generation minus demand) as G_t. We assume that G_t is a bounded random variable, that is uniformly distributed between $[G_{\max}, G_{\min}]$. The state of the system is given by $s_t = [\texttt{SoC}_t, G_{t-1}]^T$, and the action is given by a_t, which determines what fraction of the net demand (generation) is served. At every time step, the state $[\texttt{SoC}_t, G_{t-1}$ is observed and an action a_t is taken. When $G_{t-1} > 0$, since there is a net generation, the battery charge is increased by $a_t G_{t-1}$. When $G_{t-1} \leq 0$, since there is net demand, the battery charge decreases by $a_t G_{t-1}$. Note that, even when $G_{t-1} > 0$, a_t need not be very high, since the battery can get damaged due to overheating. At every time step, the battery also self discharges determined by some parameter $\beta \in [0, 1]$. The state update equation is given by,

$$\texttt{SoC}_{t+1} := \min\left\{\beta \texttt{SoC}_t + a_t \frac{G_{t-1}}{B}, 1\right\}.$$

For every action taken, the reward is specified by

$$r(s_t, a_t) = a_t \left(\tanh\left(\xi G_{t-1}\right) \frac{(r2 - r1)}{2} + \frac{(r_1 + r_2)}{2} \right),$$

where $\xi > 0$ is a scale parameter, r_1 is the reward of using renewable energy, r_2 is the utility of serving the user demand and $0 \leq r_1 < r_2$. Also, observe that the action space is constrained by the current state. Here, it depends on the current state of charge, i.e., a_t needs to be chosen such that $\texttt{SoC}_t \geq 0$ since more charge can not be extracted from the battery than what is present. Formally,

$$\Gamma(s_t) = \Big\{ a_t \in [0,1] : \beta \texttt{SoC}_t + a_t \frac{G_{t-1}}{B} \geq 0 \Big\}.$$

The goal of the RL problem is to maximize the reward until the battery doesn't run out of charge.

Offline Dataset

Here, we outline the method used to collect the offline dataset for this environment. We train a policy in the online setting using the TD3 algorithm [136]. We used the following parameters of the environment: $G_t \sim \text{Unif}(-10, 10), B = 10, \xi = 0.01, r_1 = 5, r_2 = 15, \beta = 0.97$. For online training, we used $\gamma = 0.9$ and used the same parameters as the original TD3 paper. We used the same function approximation class for $\mathcal{F}$ and $\mathcal{G}$: we used a neural network with two hidden layers each of width 4 and `ReLU` activation functions.

We use the output of the above algorithm and deploy it in the environment and log the interactions (s, a, r, s'). We now, use the logged dataset to solve (4.3).

Algorithm

To solve the min-max optimization in (4.3), we perform a bilevel optimization routine by alternating the updates on f, g.

This is written as

$$g_t = \arg\min_{g \in \mathcal{G}} \mathcal{L}_D(g, f_{t-1})$$

n	$v^{\pi_{\hat{f}}}$	v^{π_b}
100	4.3	2.9
1000	5.0	2.9
10000	5.7	2.9

Table 4.2: Comparison of sub-optimality with the number of data points n. Note that π_b is fixed for all three cases.

$$f_t = \arg\min_{f \in \mathcal{F}} \mathcal{L}_D(f, f) - \mathcal{L}_D(g_t, f)$$

At each iteration t, we sample a mini-batch of size 256 from the offline dataset D. We then use the samples to update the neural network weights (of g_t and f_t) using the Stochastic Gradient Descent (SGD) algorithm with a learning rate of `1e-5`. We iterate until $t = 10^5$.

Results

To show the impact of the dataset size, we take different dataset sizes, i.e. $n \in [500, 1000, 10000]$, and train three different algorithms using these datasets. Since we do not have the true V^*, it is difficult to compute v^* for this environment. Therefore, instead of measuring $v^* - v^{\pi_{\hat{f}}}$, we simply measure $v^{\pi_{\hat{f}}}$ where $\pi_{\hat{f}}$ is the policy greedy with respect to the output of (4.3). To compute $v^{\pi_{\hat{f}}}$, we used the initial state distribution, where $G_t \sim \text{Unif}(-10, 10)$ and $\texttt{SoC}_0 = 1.0$. We present the results of the algorithm in Table 4.2. We can observe that $v^* - v^{\pi_{\hat{f}}}$ reduces as the number of data points in the offline dataset increases which is in line with the theoretical results.

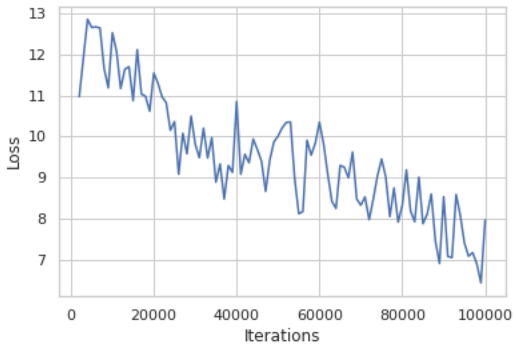


Figure 4.1: We plot the loss $\mathcal{L}_D(f_t; f_t) - \mathcal{L}_D(g_t; f_t)$ at every iteration t of the bi-level optimization algorithm.

4.5.2 Sharp Concentration Results

While earlier works [111, 113] obtain a dependence on the number of samples as $\mathcal{O}(\frac{1}{\epsilon^4})$, recent works were able to improve on the sample complexity to $\mathcal{O}(\frac{1}{\epsilon^2})$ [116, 117, 118]. The key result that leads to an improvement in the sample complexity is the sharper concentration inequality (stated below) as compared to using Pollard's concentration inequality (Lemma 19).

Lemma 12 (Lemma 11.6 [137]). *Let $B \geq 1$ and let $\mathcal{G}$ be a set of functions $g : \mathbb{R}^d \to [0, B]$. Let $Z, Z_1, \cdots, Z_n$ be i.i.d $\mathbb{R}^d$ valued random variables. Assume $\epsilon > 0$, $0 < \alpha < 1$ and $n \geq 1$. Then*

$$\mathbb{P}\left\{ \sup_{g \in \mathcal{G}} \frac{\frac{1}{n}\sum_{i=1}^n g(Z_i) - \mathbb{E}[g(Z)]}{\epsilon + \frac{1}{n}\sum_{i=1}^n g(Z_i) + \mathbb{E}[g(Z)]} > \alpha \right\}$$
$$\leq 4\mathbb{E}\left[N_1\Big(\frac{\alpha\epsilon}{5}, \mathcal{G}, Z_1^n\Big)\right] \exp\Big(- \frac{3\epsilon\alpha^2 n}{40B}\Big).$$

Reference [117, 118] decompose the error $v^* - v^{\pi_{\hat{f}}}$ in a way that exploits this sharp concentration result (notice the exponent of ϵ in exp term as compared to Pollard's

theorem reviewed in Theorem 19), thereby deriving a sharper sample complexity. Decomposing the error for the min-max variant studied in this paper to apply this result is challenging and we leave it as future work.

4.5.3 Single sample path

In this work, we assumed that the offline data available with the agent is sampled independently from a distribution $\mu \in \Delta(\mathcal{S} \times \mathcal{A})$. This assumption can be practical in several applications where the agent is highly scalable and deployed in a large number of environments to collect data. For example, in e-commerce/web applications, millions of users may query in parallel independent of one another and the deployed agent also responds independently of the other queries. However, there are several applications where a single controller/agent collects a single (long) trajectory of data in applications such as building energy management. Here the deployed agent is highly customized to the particular environment (e.g., building, factory) because each environment has different dynamics than the others. In these situations, one needs to deal with a long trajectory of data where the (s, a) pairs are no longer independent. [113] studies such a setting by assuming some mixing properties in the process. The sample complexity itself remains the same, however, the bound includes an additional term dependent on the mixing coefficient of the process.

4.5.4 Removing Concentrability Assumption

In this work, we consider the stricter condition (Assumption 8) of concentrability where every admissible $\nu \in \Delta(\mathcal{B})$ is absolutely continuous with respect to μ. A more general condition is the discounted concentrability of future state-action distributions where the distribution of future state-action pairs is assumed to be absolutely

continuous with respect to μ. Let P^π be an operator acting on $f : \mathcal{B} \to \mathbb{R}$ s.t. $(P^\pi f)(s,a) = \int_{\mathcal{B}} f(s', \pi(s'))P(ds'|s,a)$. For $m \geq 0$, and any arbitrary sequence of stationary policies $\{\pi_1, \cdots, \pi_m\}$

$$C_\mu(m) = \sup_{\pi_1,\cdots,\pi_m} \| \frac{d(\eta_{\text{init}} P^{\pi_1} P^{\pi_2} \cdots P^{\pi_m})}{d\mu} \|_\infty.$$

The discounted concentrability assumption requires $C_\mu = (1-\gamma^2) \sum_{m \geq 1} m\gamma^{m-1} C_\mu(m) < \infty$. The sample complexity analysis under discounted concentrability assumption requires substantially different arguments and thus is left as a future work.

Chapter 5: Resource Constrained Offline Reinforcement Learning

5.1 Resource Constraints

There have been many recent successes in the field of Reinforcement Learning [101, 138, 139, 140, 141]. In the online RL setting, an agent takes action, observes the outcome from the environment, and updates its policy based on the outcome. This repeated access to the environment is not feasible in practical applications; it may be unsafe to interact with the actual environment, and a high-fidelity simulator may be costly to build. Instead, offline RL, consumes fixed training data which consists of recorded interactions between one (or more) agent(s) and the environment to train a policy [107]. An agent with the trained policy is then deployed in the environment without further evaluation or modification. Notice that in offline RL, the deployed agent must consume data in the same format (for example, having the same features) as in the training data. This is a crippling restriction in many large-scale applications, where, due to some combination of resource/system constraints, all of the features used for training cannot be observed (or misspecified) by the agent during online operation. In this work, we lay the foundations for studying this Resource-Constrained setting for offline RL. We then provide an algorithm that improves performance by

transferring information from the full-featured offline training set to the deployed agent's policy acting on limited features. We first illustrate a few practical cases where resource-constrained settings emerge.

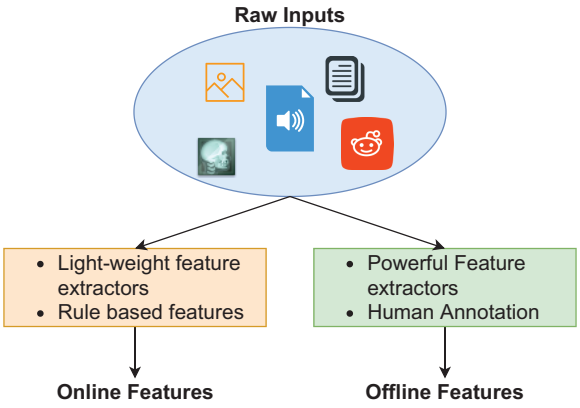


Figure 5.1: Example of system constraints

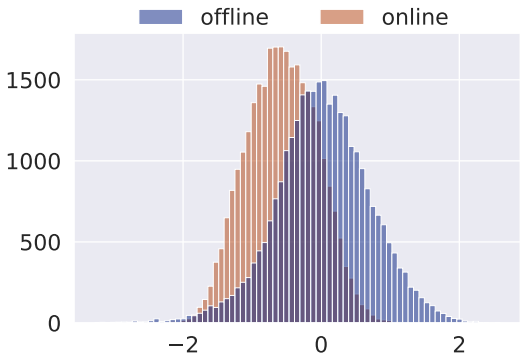


Figure 5.2: Histogram of rewards in online data collected by agents trained with online versus offline features

System Latency A deployed agent is often constrained by how much time it has to process the state of the environment and make a decision. For example, in a customer-facing web application, the customer will start to lose interest within a

fraction of a second. Given this constraint, the agent may not be able to fully process more than a few measurements from the customer before making a decision. This is in contrast to the process of recording the training data for offline RL, where one may take sufficient time to generate an abundance of features by post-processing high-dimensional measurements.

Power Constraints Consider a situation where an RL agent is being used in deep space probes or nano-satellites ([142]). In this case, an RL agent is trained on Earth with rich features and a large amount of sensory information. But when the agent is deployed and being used on these probes, the number of sensors is limited by power and space constraints. Similarly, consider a robot deployed in a real-world environment. The limited computing power of the robot prevents it from using powerful feature extractors while making a decision. However, such powerful feature extractors can be used during the offline training of the robot (Fig 5.1).

In the resource-constrained setting, one can simply ignore the offline features and only train the offline agent with the online features that are available during deployment. This strategy has the drawback of not utilizing all of the information available during training and can lead to a sub-optimal policy. To confirm this, we performed the following simple experiment. We consider an offline RL dataset for the OpenAI gym MuJoCo HalfCheetah-v2 environment and simulate the resource-constrained setting by removing a fixed set of randomly selected features during deployment (see Section 5.5.1 for more details). We train an offline RL algorithm, TD3+BC [143] using only the online features and collect online data in the environment using the trained policy. We repeat this assuming all features available during deployment, train a TD3+BC agent using the same offline dataset with all features, and collect online data in the

environment. We plot the histogram of rewards in the two datasets in Fig 5.2. We observe that the agent trained only with online features obtains a much smaller reward than the agent trained with offline features.

Traditionally, scenarios, where the observability of the state of the system is limited, are studied under the Partially Observable Markov Decision Process (POMDP) setting by assuming a belief over the observations [144]. In contrast, we have an offline dataset (which records rich but not necessarily full state transitions) along with partially obscured (with respect to the offline dataset) observations online. Our goal is to leverage the offline dataset to reduce the performance gap caused by the introduction of resource constraints. Towards this, we advocate using a teacher-student *transfer* algorithm. Our main contributions are summarized below:

- We identify a key challenge in offline RL: in the resource-constrained setting, datasets with rich features cannot be effectively utilized when only a limited number of features are observable during online operation.
- We propose the transfer approach that trains an agent to efficiently leverage the offline dataset while only observing the limited features during deployment.
- We evaluate our approach on a diverse set of tasks showing the applicability of the transfer algorithm. We also highlight that when the behavior policy used by the data-collecting agent is trained using a limited number of features, the quality of the dataset suffers. We propose a data collection procedure (RC-D4RL) to simulate this effect.

5.2 Resource-Constrained online systems

In the standard RL framework, we consider a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $\Delta(\mathcal{S})$ denotes the set of all probability distributions over $\mathcal{S}$, and $\gamma \in (0, 1)$ is the discount factor. We consider the discounted infinite horizon MDP in this paper. We consider the continuous control setting and assume that both $\mathcal{S}$ and $\mathcal{A}$ are compact subsets of a real-valued vector space. The transition at time t, is given by the tuple $(s_t, a_t, R(s_t, a_t), s_{t+1})$. Each policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, has a value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that estimates the expected discounted reward for taking action a in state s and uses the policy π after that. The goal of the agent is to learn the policy π that maximizes the expected discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$. In online RL, this problem is solved by interacting with the environment.

In *offline (or batch) RL* [108], instead of having access to the environment, the agent is provided with a finite dataset of trajectories or transitions denoted by $D = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$. The data is collected by one or many behavior policies that induce a distribution μ on the space of $\mathcal{S} \times \mathcal{A}$. The goal of the agent is to learn a policy using the finite dataset to maximize the expected discounted reward when deployed in the environment.

In the resource-constrained setting, the agent does not have access to the full state space or features during deployment. Instead, the agent can only observe from $\hat{\mathcal{S}}$ (another bounded subset of the real-valued vector space) that is different from $\mathcal{S}$. It is assumed that the space $\mathcal{S}$ is rich in information as compared to $\hat{\mathcal{S}}$. For example, $\hat{\mathcal{S}}$ might have fewer dimensions, or some entries may include extra noise (see Figure 5.1).

We will use online/limited features, to refer to observations from the online space $\hat{\mathcal{S}}$, offline/rich features to refer to observations from the offline space $\mathcal{S}$.

We assume that both online features and offline features are available in offline data.

The goal of the agent is to use the offline data and train a policy $\pi : \hat{\mathcal{S}} \to \Delta(\mathcal{A})$. The agent can use the offline features from $\mathcal{S}$ during training but is constrained to only use the online features from $\hat{\mathcal{S}}$ while making a decision. A similar paradigm of Learning Under Privileged Information (LUPI) [145] has been studied in the supervised learning setting, where the privileged information is provided by a knowledgeable teacher.

5.3 Related Work

Offline RL There has been an increasing interest in studying offline RL algorithms due to its practical advantages over online RL algorithms [31, 146, 147, 148]. Offline RL algorithms typically suffer from an overestimation of the value function as well as a distribution shift between the offline data and on-policy data. [149] and [29] advocate a pessimistic approach to value function estimation to avoid over-estimation of rarely observed state-action pairs. To constrain the on-policy data to be closer to offline data, several techniques have been explored, such as restricting the actions inside the expectation in the evaluation step to be close to the actions observed in the dataset [150], adding a regularization term during policy evaluation or iteration [151], [152],[153], adding a constraint of the form $\text{MMD}(\mu(\cdot|s), \pi(s))$ [154, 155, 156] on the policy [157], using behavior cloning [143], adding an entropy term in the value function estimation [152], and model-based approaches that learn a pessimistic MDP [158]. A thorough review of these techniques is presented in an excellent tutorial by

[107]. To the best of our knowledge, there is no existing work that addresses the resource-constrained offline RL setting where there is a mismatch between the offline features and online features.

Knowledge Transfer Knowledge transfer/distillation is widely studied in various settings including vision, language, and RL domains [159, 160]. In RL, under the domain transfer setting [161, 162], the teacher is trained on one domain/task and the student needs to perform on a different domain/task [163, 164, 165, 166]. [167] train a model so that features from different domains have similar embeddings, and [168] perturb the feature using a random noise centered at the privileged information. An offline RL algorithm for domain transfer has been proposed by [169]. Policy distillation is studied in the setting where the knowledge from a trained policy (teacher) is imparted to an untrained network (student) [170, 171]. This leads to several advantages such as model compression and the ability to learn from an ensemble of trained policies to improve performance [172].

One distinguishing feature of the resource-constrained setting that differentiates it from other transfer settings is that the teacher has access to the privileged information and the student needs to adapt using the data available without interactive learning. In most of the existing approaches, the difference between teacher and student was either the network size (which is also present in our setting due to the difference in input features) or the dynamics (as in the domain transfer case). To the best of our knowledge, we are the first to study policy distillation in the offline RL framework under the resource-constrained setting. Another interesting line of work is called Sim2Real [173, 174]. In these papers, they train a model using a simulator and then transfer the knowledge to real data. However, this work requires an accurate simulator

which results in a fairly expert teacher model. However, in the offline RL setting, depending on the data quality, the teacher itself might be weak.

Partially Observable MDP POMDP generalizes the MDP framework where the agent does not have access to the full features and only partially observes the state space [144, 175, 176]. More recently, [177] studied a model-based offline RL algorithm for image data under the POMDP setup. Our setting resembles this setup, but our agent also has access to the full privileged features in the offline dataset while training. This availability of the offline dataset with privileged information differentiates our setting and enables the student to inherit the knowledge from the rich space while only using the limited features during deployment.

5.4 Proposed Algorithm

5.4.1 TD3+BC (0, 1)

Let us first discuss TD3+BC [143] which is a recent state of the art algorithm for continuous control offline RL. Then we discuss a simple way to extend it to the resource-constrained setting. TD3+BC maintains two critic networks $Q_{\theta_1}, Q_{\theta_2} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with parameters θ_1, θ_2 and a deterministic actor network $\pi_\phi : \mathcal{S} \to \mathcal{A}$ with parameters ϕ. TD3+BC shows that by adding an additional behavior cloning term to an existing online RL algorithm TD3 [136] (that learns a deterministic policy), it is possible to attain state-of-the-art performance on the D4RL benchmark datasets [178]. The policy evaluation step in TD3+BC is the same as TD3 where the minimum of two Q functions is used to reduce the over-estimation bias of the value function (lines 7-9 in Algorithm 4). In the policy iteration step (5.1), the behavior cloning term $(\pi_\phi(s) - a)^2$ regularizes the policy to take actions similar to the actions observed in the dataset,

Algorithm 4 TD3+BC

Given: offline dataset D with full feature observations
Given: policy update frequency d; weighted average parameter τ, noise parameter $\bar{\sigma}$
Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ and an actor network π_ϕ
Initialize target networks $\theta_1^{'} \leftarrow \theta_1, \theta_2^{'} \leftarrow \theta_2, \phi^{'} \leftarrow \phi$.
for $t = 1, \cdots, T$ **do**
    Sample mini-batch of N transitions $(s, a, r, s') \in D$
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \min(\max(\epsilon, -c), c)$ where $\epsilon \sim \mathcal{N}(0, \tilde{\sigma})$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{N} \sum_{i=1}^{N} \left(y - Q_{\theta_i}(s, a)\right)^2$
    if t `mod` d $== 0$ **then**
        Update ϕ by deterministic policy gradient optimizing the objective

$$\arg\max_{\phi} \frac{1}{N} \sum_{i=1}^{N} \left[\lambda Q_{\theta_1}(s_i, \pi_\phi(s_i)) - \left(\pi_\phi(\hat{s}_i) - a\right)^2 \right] \tag{5.1}$$

        Update target networks:
            $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$
            $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
    end if
end for

where λ is given by

$$\lambda = \frac{\alpha}{\frac{1}{N} \sum_{(s_i, a_i)} |Q_{\theta_1}(s_i, a_i)|},$$

where α is a hyper-parameter.

Proposed Algorithm We now propose our transfer algorithm that adopts TD3+BC to the resource-constrained setting. We add an additional regularization term during the policy iteration step, that tries to keep actions predicted by the trained policy close to the actions taken by the teacher policy. This knowledge can be imparted to the student policy through the regularization term below (see (5.2) for

the full equation)

$$\overbrace{\beta_1\Big(\pi_\phi(\hat{s})-a\Big)^2}^{\text{Behavior Cloning}}-\overbrace{\beta_2\Big(\pi_{\phi^{\texttt{teacher}}}(s)-\pi_\phi(\hat{s})\Big)^2}^{\text{Transfer}}.$$

We weigh the two terms with weights β_1, β_2. The transfer term is beneficial in learning because the teacher (trained using offline RL algorithm on full features) often predicts a better action than those available in the dataset (depending on the quality of the dataset) as observed in earlier offline RL works [29]. To keep all terms in (5.2) at comparable magnitudes, we add a constraint that $\beta_1 + \beta_2 = 1$. If $\beta_1 = 1$ and $\beta_2 = 0$, the proposed algorithm is equivalent to the TD3+BC algorithm. For the purpose of this work, we consider the TD3BC(0, 1) algorithm.

5.4.2 Simultaneous Transfer Policy Iteration

The proposed algorithm incorporates a knowledge transfer technique into the TD3+BC algorithm to train the network to deal with online features. Usually knowledge transfer algorithms (like TD3BC(0,1) in the previous section) first train a teacher network and then distill the teacher's knowledge into a student network. The proposed algorithm is different, in the sense that, we train the teacher network and the student network simultaneously. The policy evaluation step of the teacher is the same as in TD3. The student network only learns the actor-network, since during inference, the agent only needs to use the actor-network π_ϕ to make a prediction.

Note that the proposed algorithm is an apt example for the inference at the edge challenge we discussed in Chapter 1. Here, despite being able to train an actor using TD3+BC using offline features, during inference due to the feature mismatch, the trained actor can not be used. Therefore, there is a need to adapt the trained actor to use the online features.

The key to the proposed algorithm is to introduce a transfer loss term that enables distilling the knowledge of the teacher network to the student network. Before that, let us denote the student actor network using the parameters $\phi^{\texttt{student}}$. Note that the teacher parameters are denoted by θ_1, θ_2, ϕ as earlier. Additionally, we also use $\pi_{\phi_{-1}} : \mathcal{S} \to \mathbb{R}^{d_e}$, to denote the output of the last but one layer of the actor network, where d_e is the dimension of the last but one layer. The transfer loss is then defined as the mean squared distance between the last layer embeddings of the teacher actor network (evaluated using the offline features) and the last layer embeddings of the student actor network (evaluated using the corresponding online features). Formally, we write the transfer loss as $\left\| \pi_{\phi_{-1}}(s) - \pi_{\phi^{\texttt{student}}_{-1}}(\hat{s}) \right\|^2$. The policy iteration step of the proposed algorithm can then be written as in (5.2).

$$\begin{aligned} \arg \max_{\phi, \phi^{\texttt{student}}} \frac{1}{N} \sum_{i=1}^{N} \Bigg[& \underbrace{\lambda Q_{\theta_1}(s_i, \pi_\phi(s_i)) - \Big(\pi_\phi(s_i) - a\Big)^2}_{\text{teacher loss}} \\ & + \underbrace{\lambda Q_{\theta_1}(\hat{s}_i, \pi_{\phi^{\texttt{student}}}(\hat{s}_i)) - \Big(\pi_{\phi^{\texttt{student}}}(\hat{s}_i) - a\Big)^2}_{\text{student loss}} \\ & + \underbrace{\left\| \pi_{\phi_{-1}}(s) - \pi_{\phi^{\texttt{student}}_{-1}}(\hat{s}) \right\|^2}_{\text{transfer loss}} \Bigg]. \end{aligned} \tag{5.2}$$

Here, the teacher loss is the same as in the policy iteration step of TD3+BC. In addition to the teacher loss, the teacher actor network ϕ is optimized to minimize the transfer loss as well. The student network $\phi^{\texttt{student}}$ is optimized to minimize the student loss and the transfer loss. The key insight behind the transfer loss is that, for a given observation, the embeddings for the teacher actor network (using full features) and student actor-network (using online features) must be close. The transfer loss regularizes the training of the teacher and student networks. The teacher

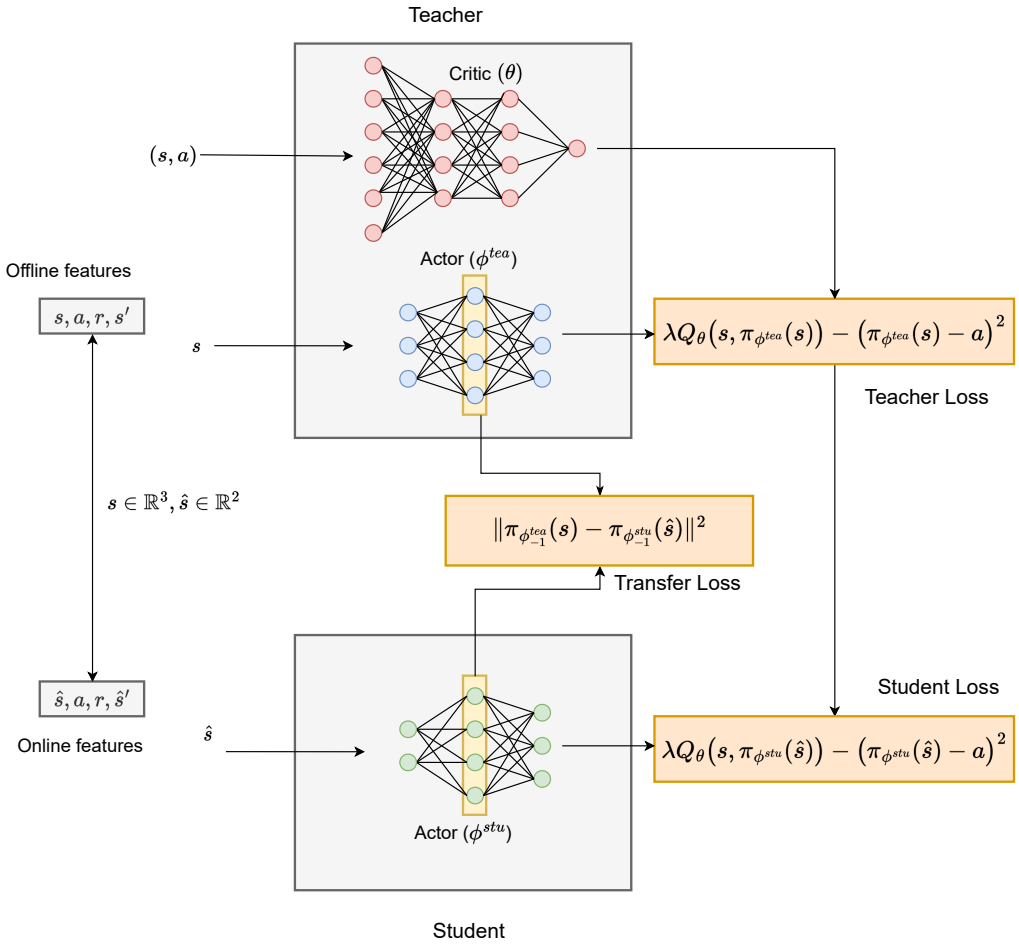


Figure 5.3: Simultaneous Transfer Policy Iteration

network is therefore incentivized to learn embeddings that can be represented using the information present in only the online features. The teacher actor network also influences the training of the teacher critic network (which in term features in the teacher loss and student loss terms). Therefore, the teacher's critic and actor networks are both trained to optimize the transfer loss. In this way, the student network learns better embeddings using the online features and we observe from experiments that

this is an effective way to perform the transfer. The complete algorithm is provided in Algorithm 5 and an illustration of the algorithm is provided in Figures 5.3.

Algorithm 5 Proposed transfer algorithm

Given: offline dataset D with full feature observations
Given: policy update frequency d; weighted average parameter τ, noise parameter $\bar{\sigma}$
Initialize teacher critic networks $Q_{\theta_1}, Q_{\theta_2}$ and an teacher actor network π_ϕ
Initialize student actor network $\pi_{\phi^{\texttt{student}}}$
Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi, \phi'^{\texttt{student}} \leftarrow \phi^{\texttt{student}}$.
Note that $\pi_{\phi_{-1}}(s)$ outputs the embeddings of the last but one layer of the actor-network $\forall \phi$.
for $t = 1, \cdots, T$ **do**
    Sample mini-batch of N transitions $(s, a, r, s') \in D$
    Online features for this transition is given as $(\hat{s}, a, r, \hat{s}')$
    $\tilde{a} \leftarrow \pi_{\phi'}(\hat{s}') + \min(\max(\epsilon, -c), c)$ where $\epsilon \sim \mathcal{N}(0, \tilde{\sigma})$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(\hat{s}', \tilde{a})$
    Update critics $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{N} \sum_{i=1}^{N} \left(y - Q_{\theta_i}(\hat{s}, a)\right)^2$
    if t `mod` d == 0 **then** Solve (5.2)
        Update target networks:
            $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
            $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
    end if
end for

5.5 Experimental Results

We designed our experimental analysis to answer the following questions.

- Do the proposed transfer algorithms outperform a baseline algorithm that is trained only on the limited online features? See Figures 5.4 and 5.5.

- How do the improvements offered by the transfer algorithms vary with the dataset difficulty? See Figures 5.7, 5.8, 5.9.

- How do the improvements offered by the transfer algorithms vary with the number of offline features that are left out of the online setting? See Figures 5.7, 5.8, 5.9.

We performed experiments on continuous control tasks, from the OpenAI gym MuJoCo [103] suite to study the resource-constrained setting. Before we discuss the resource-constrained setup used for the MuJoCo environments in Section 5.5.1, it is important to note that the data-collecting agent plays a vital role in determining the quality of the dataset (coverage of state-action pairs) and subsequently the performance of any offline RL algorithm. In the D4RL suite [178], datasets were collected using behavior policies of varying expertise to simulate the variation in data quality as follows

- medium: Data collected by deploying a policy that is not trained to expert level.
- expert: Data collected by deploying an expertly trained policy in the environment
- medium-replay: Data collected from the replay buffer of an agent trained to a medium level (approximately half of the expert score).
- medium-expert: Data collected by merging the medium-level dataset and expert-level dataset.

These dataset difficulty levels have an impact on the performance of an offline RL agent due to the level of exploration present in each dataset. In general, the quality of the dataset (in terms of average rewards observed in the dataset) follows the order: medium-replay $<$ medium $<$ medium-expert $<$ expert.

Note that, each of these behavior policies was trained online using the full feature set. Thus, these policies can use all of the information and explore the environment

online. The quality of the offline dataset collected by these behavior policies is thus relatively high thereby improving the performance of offline RL algorithms on them. In the resource-constrained setting, however, the behavior policy of the data-collecting agent does not have access to the full features online. The agent may only explore and navigate using the limited feature set during training. Using this behavior policy to collect data results in a relatively lower-quality dataset. However, for the scope of this work, we consider the case where the data-collecting agent is trained using the full offline features.

5.5.1 Simulation of Resource-Constrained Setting

We simulate the resource-constrained setting by reducing the feature space available during the deployment of the agent. We do this by dropping a fixed set of features from the full features available (this is possible in the system latency as well as the nano-satellite example). For instance, consider the MuJoCo environment Hopper-v2 where the original state space is 11 dimensional. We consider four scenarios where the online observable feature dimensions are reduced to 5, 7, 9, and 10 by randomly picking a subset of the features of the given dimension. For each of these scenarios, we consider 3 random seeds to simulate different features getting dropped in each seed. We summarize this setting for the three environments in Table 5.1.

Environment	Original Dim	Resource-Constrained Dimensions	Number of Seeds
Hopper	11	5, 7, 9, 10	3
HalfCheetah, Walker2d	17	9, 11, 13, 15	3

Table 5.1: Resource-Constrained Simulation using gym MuJoCo tasks

5.5.2 Training and Evaluation

When reporting the performance of the algorithm, we assume online access to the gym MuJoCo simulator for evaluation. The agents are restricted to using the limited features (defined by the combination of the offline dataset used during training) to make a decision. Given a policy to evaluate, we perform 10 different rollouts using this simulator with random initial states and compute the mean of the cumulative rewards. We perform one round of evaluation of the policy (student or baseline) during training at a fixed frequency. We take the average value of the last 10 evaluation rounds during training and report the mean and the standard deviation of the score. Lastly, since we run multiple trials (for each configuration), we compute the mean and standard deviation of the results of the three random trials for the given configuration. In order to facilitate easier understanding and comparison of the results across datasets and environments, we adopt the normalized score computation [179] [178]

$$\text{normalized score} = 100 \times \frac{\text{score - score of random policy}}{\text{score of expert online policy - score of random policy}}.$$

Comparison

We term as baseline the case where only the online features are available during deployment and we train the TD3+BC algorithm (Algorithm 4) using all the features. In addition to this baseline, we also evaluate a pure behavior cloning algorithm that takes a teacher policy as input and learns to imitate the teacher. The new policy is learnt by minimizing

$$\arg\min_{\phi} \mathbf{E}_{s_i \sim D}\Big[(\pi_{\text{teacher}}(s_i) - \pi_{\phi}(\hat{s}_i))^2\Big].$$

The policy π_ϕ uses a similar architecture as the teacher, with the exception that the input number of features is reduced due to the online features available.

We also consider an additional baseline that predicts the missing features (offline features) from the available online features. We do this by first training an autoencoder that takes the online features as input and predicts the offline features by minimizing the MSE loss between the predicted offline features and the actual offline features. The trained autoencoder is than passed to the offline RL algorithm (that is trained for deployment). During every step of training, the algorithm takes the online features, predicts the offline features using the autoencoder and uses the predicted features as the state observation. Similarly, during evaluation, the trained agent first predicts the features using online features and uses them to take an action.

We adopted the base hyperparameters from TD3 since hyperparameter tuning in offline RL is a difficult task without the access for the environment during training. We used the normalized score [178] for evaluating the algorithm.

From Figures 5.4 and 5.5, we can see that the proposed transfer algorithms significantly outperform the considered algorithms for all three environments considered. Similarly, in Fig 5.6, we observe that the proposed algorithms outperform the other algorithms for all difficulties of the datasets. Occasionally, we also observe that the PureBC or the Autoencoder algorithms are also on par with the proposed algorithms (and better in some cases). Since these algorithms also use some kind of transfer, it supports our view that using additional information from the offline features improves over the performance of just using the online features. A more detailed summary of the results for each environment, difficulty and online dimension combination is provided in Figures 5.7, 5.8 and 5.9. It can be observed that, when the online dimension is

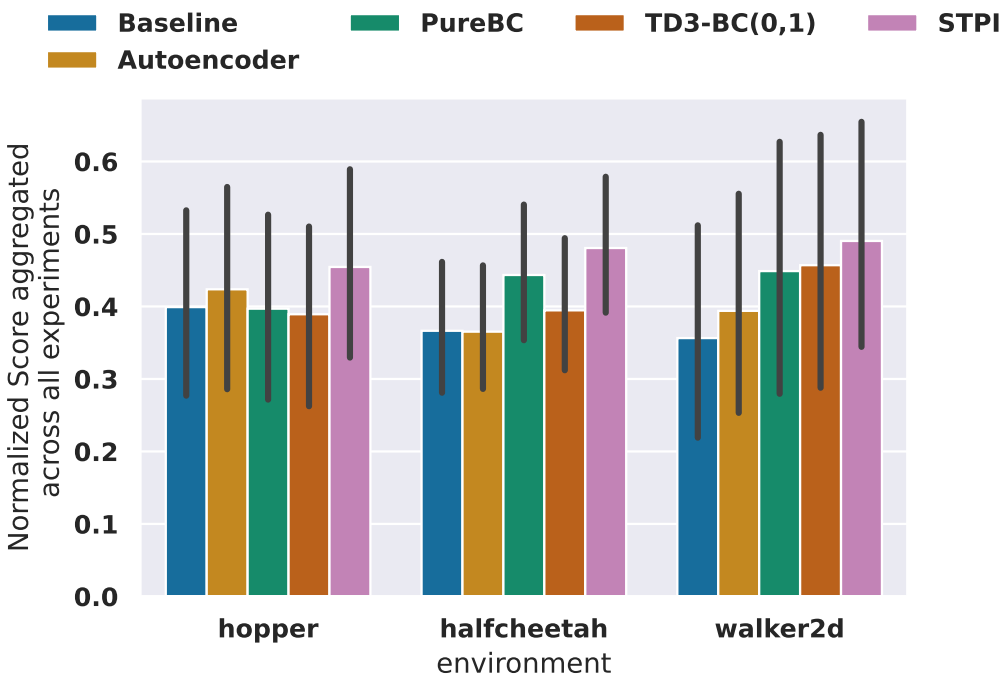


Figure 5.4: Comparison of normalized scores for all the experiments per environment

very small, the overall performance of all the algorithms considered is poor, which is expected. However, even in that case, the proposed algorithms provide a decent improvement over the baseline by leveraging the offline information.

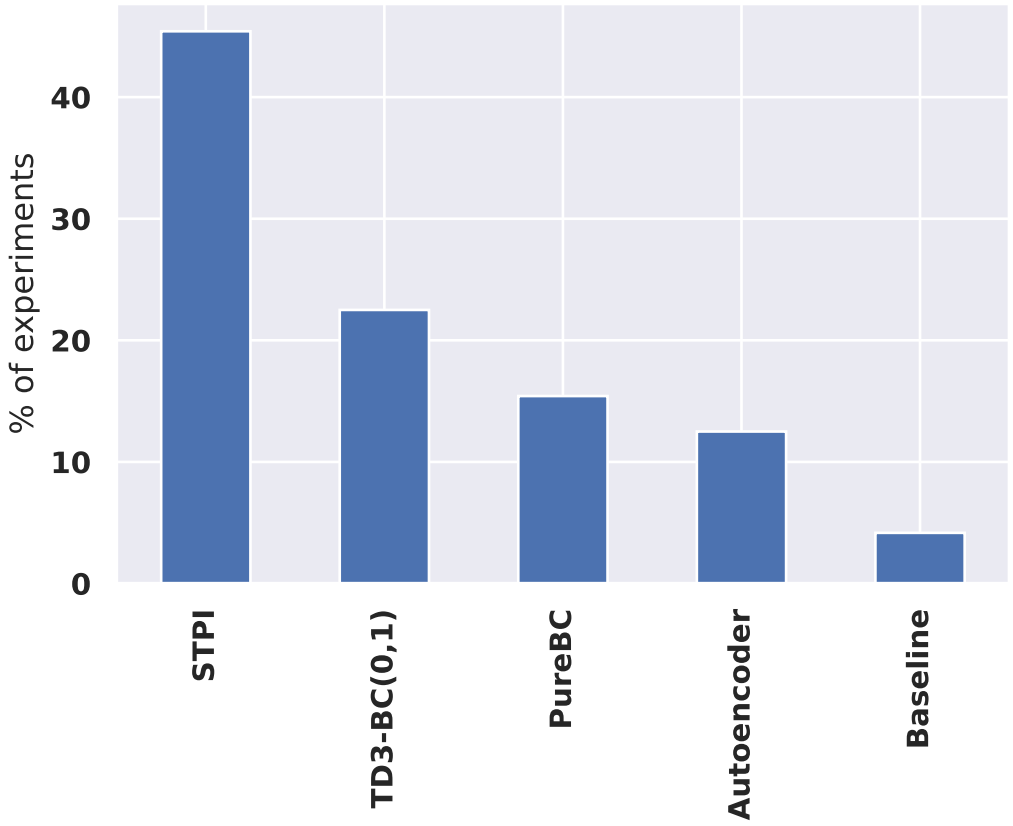


Figure 5.5: % of experiments where each algorithm achieves the highest performance as compared to the other algorithms

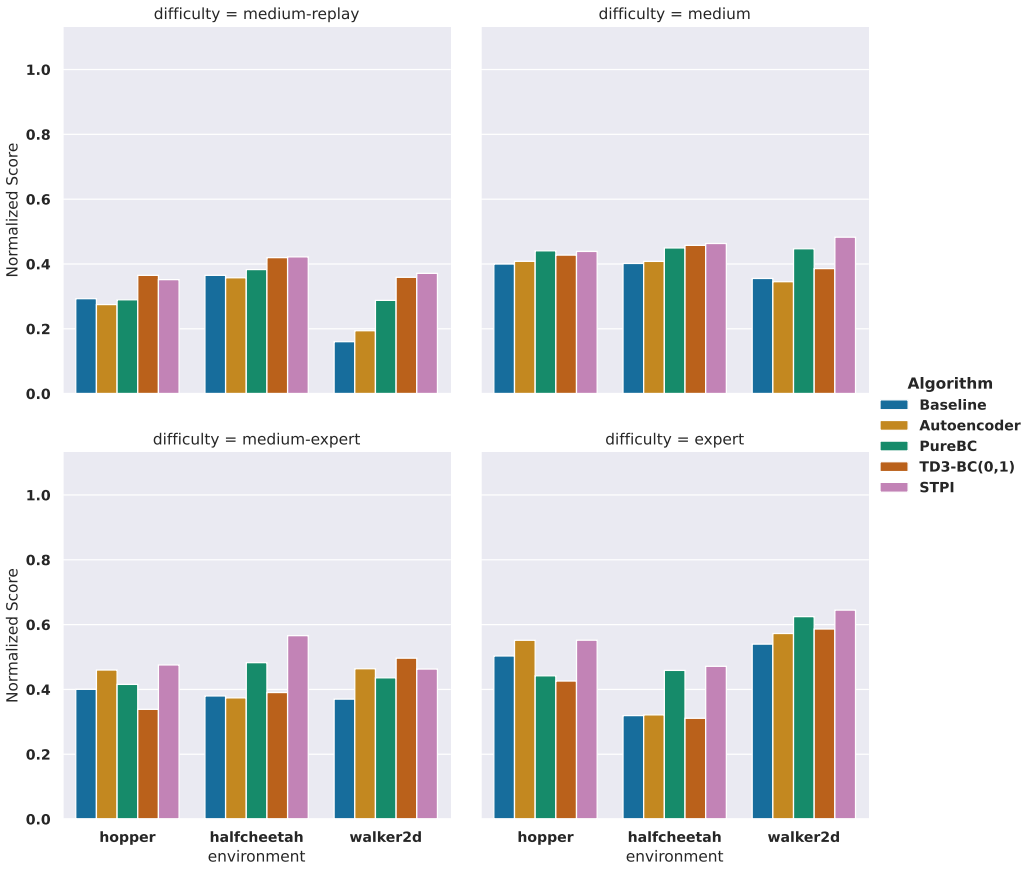


Figure 5.6: Comparison of the algorithms for different difficulties of the dataset

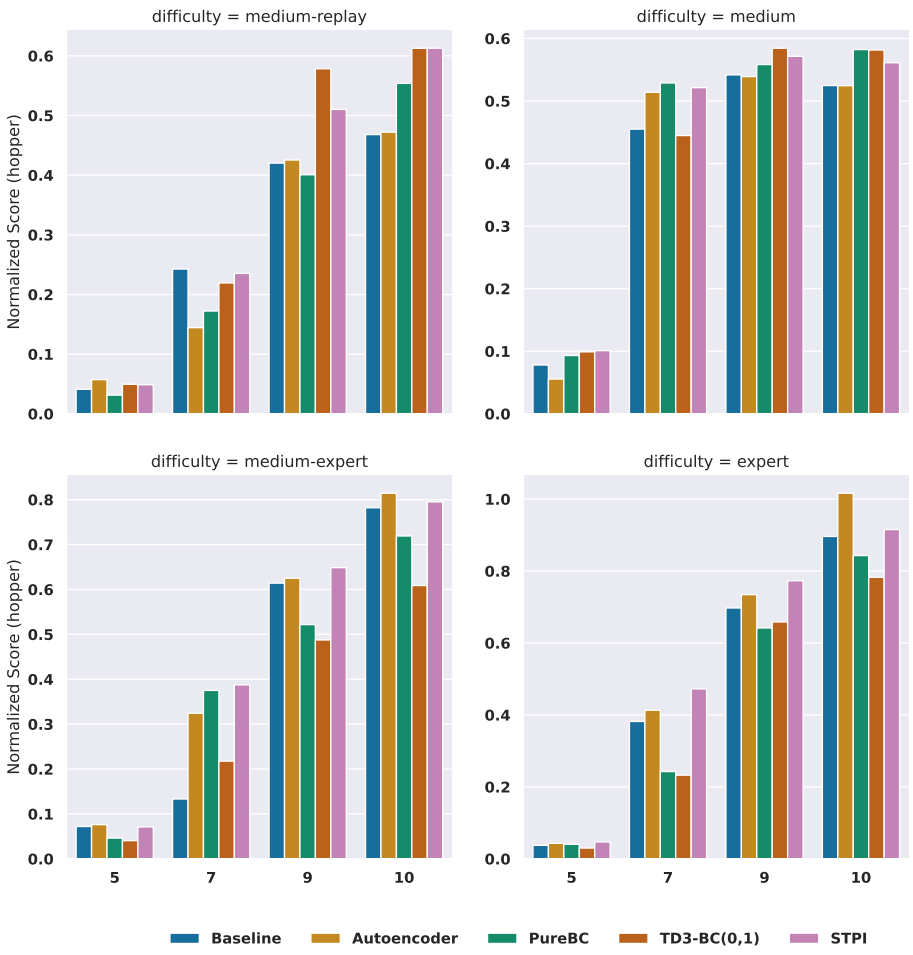


Figure 5.7: Summary of the algorithms performance on Hopper environment

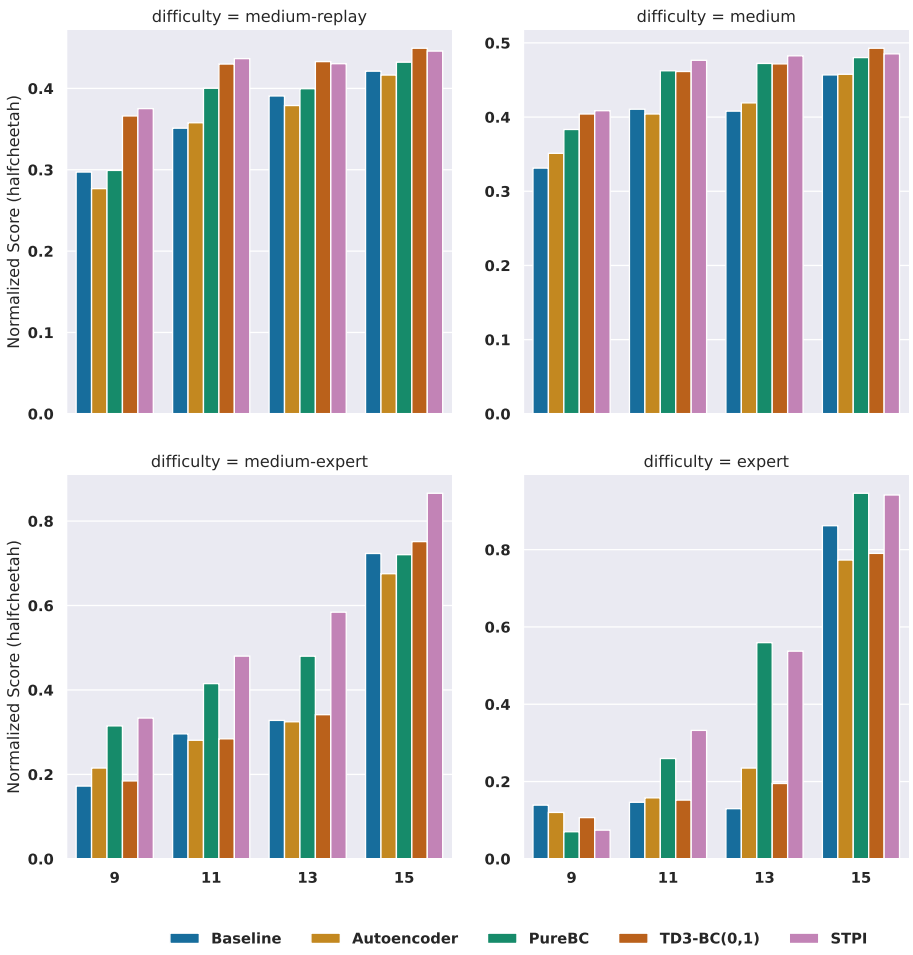


Figure 5.8: Summary of the algorithms performance on HalfCheetah environment

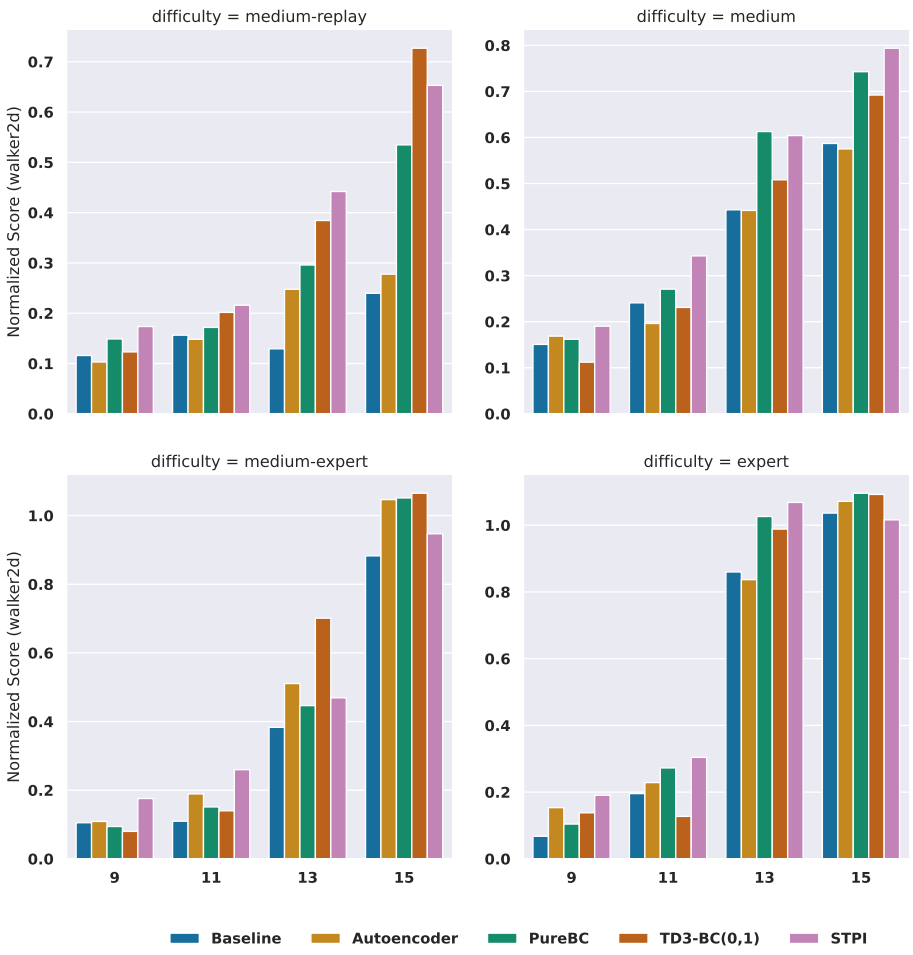


Figure 5.9: Summary of the algorithms performance on Walker2d environment

Chapter 6: Conclusion

In this thesis, I have discussed some challenges that arise out of the deployment of learning algorithms at the resource-constrained edge. In summary, the contributions are as follows:

1. Model heterogeneous FL setup: I studied this problem without assuming a public dataset, without imposing restrictions on the choice of the model architectures, and keeping the model architectures private. I proposed Fed-CMA based on conditional distribution alignment in the latent space in a federated way. I prove the convergence and generalization properties of the algorithm with an emphasis on the role of the learned representations. These insights are helpful in designing better FL algorithms and the proposed algorithm can be used along with existing FL algorithms in the weight-sharing setup, and/or use an adversarial loss to improve the performance. Moreover, our work opens up several intriguing questions: (i) What level of model heterogeneity can be tolerated to achieve a reasonable improvement in the FL setup?, (ii) What is the role of the latent space in tolerating this model heterogeneity?

2. Robust defenses for distributed setup: I propose reputation score-based aggregation for distributed machine learning that is resilient to any number of adversarial

workers. I showed that under reasonable assumptions, ByGARS++ converges to the optimal solution using results from two-timescale stochastic approximation theory. Through simulations, I showed that the proposed algorithms exhibit remarkable robustness properties even for non-convex problems under a wide range of Byzantine attacks. Although ByGARS and ByGARS++ are developed for the Byzantine attack setting, I believe that these algorithms serve a much more general purpose. This algorithm can be modified to train models in other cases such as learning from heterogeneous datasets, learning under privacy constraints, and other adversarial settings (such as adaptive adversaries).

3. Offline Reinforcement Learning for resource-constrained setup: In the resource-constrained setting that is motivated by real-world applications, the features available during training offline may be different than the limited features available online during deployment. I highlighted a performance gap between offline RL agents trained using only the online features and agents trained using all the offline features. To bridge this gap, I proposed a student-teacher-based policy transfer learning approach. The proposed algorithm improves over the baseline significantly even when the dataset quality is lacking in the resource-constrained setting. The simplicity of the approach (with just one additional hyperparameter) makes it easy to extend it to other offline RL algorithms. It would be interesting to study other transfer learning approaches (e.g., policy transfer with other divergence regularizations for stochastic policies) in the future. Moreover, I observe that the proposed approach benefits especially when the dataset quality is low which is often the case with real-world datasets. Despite

this, the performance gap with the teacher is still high (in the low-quality data regime) and this suggests more tailored approaches are required.

6.1 Future Work

Let us now spend some time understanding what are some interesting business problems that might spur new research directions in the future.

Assisted Learning Consider a small business entity that wants to deploy a specialized machine-learning model. For example, it could be an online store that wants to provide better search suggestions based on the users' inputs. It could be a small lender that operates in a niche lending market (debt consolidation or refinancing loans for very low credit score customers) that wants to predict that a new user defaults on the loan. It could be a small chain of retail stores (with a physical and online presence) that wants to forecast demand and stockpile an inventory distributed across its stores. It could be a new healthcare company that wants to integrate AI assistance to its radiologists. It could be a service company that provides Interactive Voice Response System for regional languages that require speech recognition for the respective language. It could a chatbot provider for mobile applications in a regional language. For example, India has 20+ officially recognized languages and there is a considerable user base for each regional language. A large multinational corporation that specializes in state-of-the-art chatbot services (in major languages like English, French, and Spanish) may not have the incentive and expertise to adapt its services to different regional languages. On the other hand, a small regional language chatbot provider may not have the huge data corpus (from previous interactions with millions

of users) that the larger corporation has. Note that the large corporation's model may not work directly for the smaller corporation.

In such cases, the entity concerned here is neither too big to have tonnes of data that can provide sharp insights, nor too small to ignore. Especially, if the entity is relatively new, it lacks a lot of historical data. However, there are larger corporations for each of these examples, that have huge amounts of data which the corporations are already monetizing by incorporating ML models for predictions, pattern recognition, etc. The larger corporation in question here can be a consortium of smaller companies/entities that collaborate together to train better models. For the scope of this discussion, we consider it to be a single entity that has huge amounts of data and an already trained model that performs relatively well for the larger corporation. The business opportunity here is that the larger entity can monetize its data/trained model (without actually competing in the smaller market) and the smaller entity can use the larger entity's model/data to benefit itself by paying some cost. Please note that the data and the model of the large entity are considered to be private, so they can not be shared with the smaller entity.

One business model is to provide an API to serve the smaller business where the smaller business pays per prediction to use the API. However, it has several drawbacks.

- The model for the large entity may not specialize to the smaller business. Consider the English chatbot with a Telugu chatbot. The high-level patterns of interactions (between the users and the chatbot) remain the same in a semantic space, but not in the actual speech space.

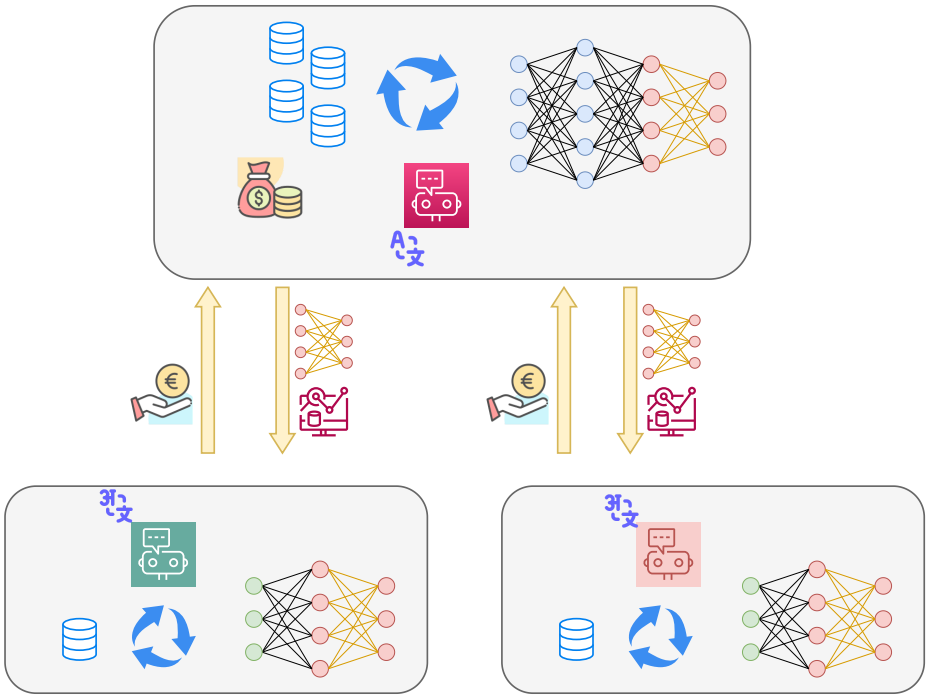


Figure 6.1: Illustration of Assisted Learning

- The smaller business may not want to reveal its data to the API and it may not be economical for the business to keep paying the larger entity for its predictions. Developing an in-house model may be cheaper.

[180] propose the assisted learning framework where the larger corporation provides a service that helps the smaller entities to train on their private data. See Figure 6.2 for a connection to the theme of this work. Consider that the larger corporation has a huge dataset $D_L = \{(x_i, y_i)\}_{i=1}^{N_L}$ and a well trained model $\mathbf{w}_L := (\mathbf{u}_L, \mathbf{v}_L) \in \mathcal{W}_L$ such that $\mathbf{u}_L \in \mathcal{U}_L$ and $\mathbf{v}_L \in \mathcal{V}$. The smaller entity has a dataset $D_S = \{(x_i, y_i)\}_{i=1}^{N_S}$ and the model space of the smaller corporation is $\mathcal{W}_S = (\mathcal{U}_S, \mathcal{V})$. The assumption here is that the latent space embedding dimension of both models is the same. The objective

is to solve

$$\arg \min_{\mathbf{u}_S \in \mathcal{U}_S, \mathbf{v}_s \in \mathcal{V}} \sum_{i=1}^{N_S} F(\mathbf{w}_S, \mathbf{v}_S, x_i, y_i, P(\mathbf{w}_L, D_L))$$

where $P(\mathbf{w}_L, D_L)$ is some private way to access the knowledge embedded inside the model $\mathbf{w}_L$ of the larger corporation and F is a function that incorporates this knowledge into the training loss for the smaller entity. One example of $P(\mathbf{w}_L, D_L)$ is to provide predictions on a data point that is available to both entities. Another example could be to share the class conditional mean embeddings of the larger entity with the smaller entity as in the FedCMA algorithm. See Figure 6.1 for an illustration of the interaction between the corporations. Algorithm 6 in Section A.4.7 provides an example algorithm for this assisted learning using the FedCMA prototype. From the results in Section A.4.7, we observe that FedCMA is a suitable fit for the assisted learning setup since FedCMA is designed to improve the generalization performance of the users. While in FedCMA, we observed that aligning the conditional moments in the latent space is sufficient, it is possible to improve the alignment process by using higher moments as well which remains to be explored.

However, this business opportunity opens avenues to some interesting and broad research problems on incentivizing users to participate in Federated Learning.

1. How should the new users/small business entities be charged for availing the learning services?

2. How can a small business entity determine if the large entity's model is indeed useful before paying for the service?

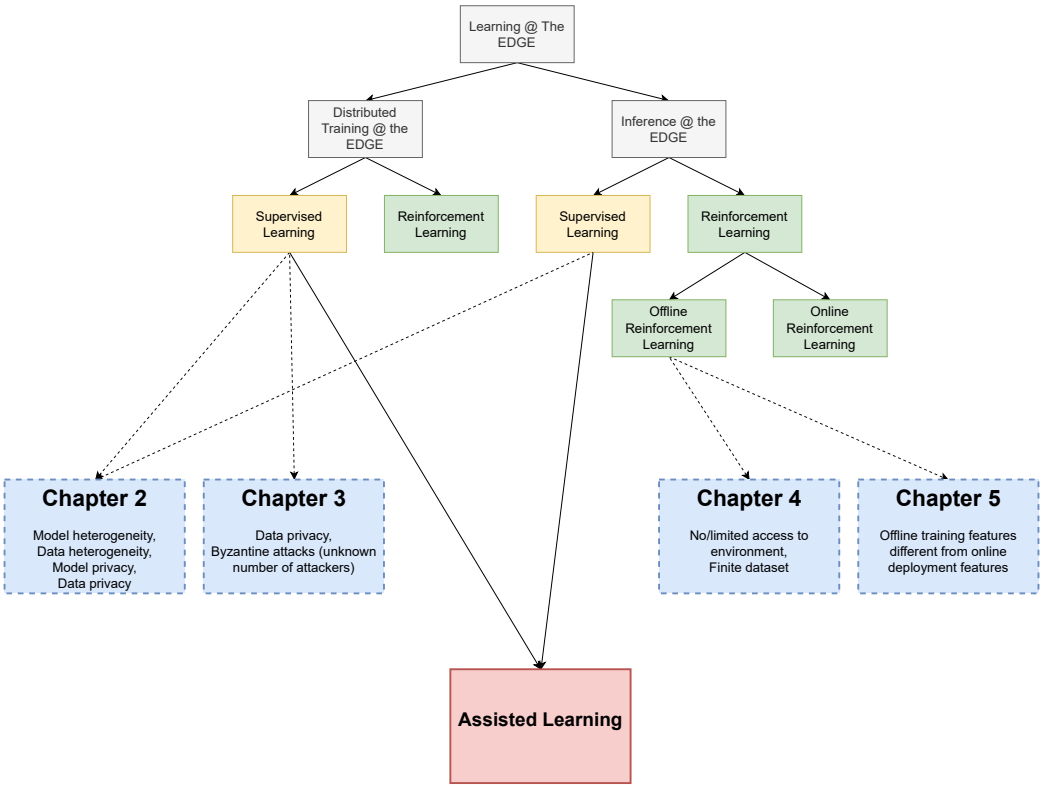


Figure 6.2: Illustration of future work

Appendix A: Chapter Two Proofs

A.1 Additional Preliminaries

MMD: Define a feature map $\phi : \mathcal{X} \to \mathcal{H}_k$ where $\mathcal{H}_k$ is a reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}_k}$. The Maximum Mean Discrepancy (MMD) between two distributions $\mathcal{D}, \mathcal{D}'$ is defined as $\text{MMD}(\mathcal{D}, \mathcal{D}') = \left\| \mathbb{E}_{x \sim \mathcal{D}}[\phi(x)] - \mathbb{E}_{x' \sim \mathcal{D}'}[\phi(x')] \right\|_{\mathcal{H}_k}$. The empirical estimate of the MMD given two datasets D, D' of size N each is given by $\text{MMD}(D, D') = \frac{1}{N} \sum_{x_i, x_j \in D} k(x_i, x_j) + \frac{1}{N} \sum_{x'_i, x'_j \in D'} k(x'_i, x'_j) - \frac{2}{N} \sum_{x_i \in D, x'_j \in D'} k(x_i, x'_j)$. When ϕ is an identity map, minimizing the MMD is equivalent to minimizing the first order moments and when ϕ is a gaussian kernel, then it is equivalent to minimizing all moments. In this work, we consider minimizing the distance between the first order moments (and the algorithm/analysis can be easily extended to a gaussian kernel case).

A.2 Convergence Results

Let us denote $\mathbf{a} = (\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$ and we can write the global objective as

$$\arg\min_{\mathbf{a}} \Phi(\mathbf{a}) := \arg \min_{\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \bar{\mathbf{e}}} \sum_{i=1}^{M} \Phi(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}). \tag{A.1}$$

$$\text{(Local update)} \begin{cases} \mathbf{w}_{(i,t+1)} & = \mathbf{w}_{(i,t)} - \eta_{(t)} \nabla_{\mathbf{w}_i} \widehat{\Phi}_i(\mathbf{w}_{(i,t)}, \bar{\mathbf{v}}_{(t)}, \mathbf{e}_{(t)}); \end{cases} \tag{A.2}$$

$$\text{(Global update)} \begin{cases} \bar{\mathbf{v}}_{(t+1)} & = \bar{\mathbf{v}}_{(t)}(1 - \lambda_1 \eta_{(t)}) + \lambda_1 \eta_{(t)} \Big(\sum_{i=1}^{M} p_i \mathbf{v}_{(i,t)} \Big); \\ e^k_{(t+1)} & = e^k_{(t)}(1 - \lambda_2 \eta_{(t)}) + \lambda_2 \eta_{(t)} \left(\sum_{i=1}^{M} p_i^k e_i^k \big(\mathbf{u}_{(i,t)} \big) \right), \end{cases} \tag{A.3}$$

Let us restate the assumptions for completeness.

Assumption 10. *The local loss functions* $\{L_i(\mathbf{w}_i)\}_{i=1}^N$ *are* β_l *Lipschitz continuous, and* β_s *smooth in* $\mathbf{w}_i$*. Note that* $\|\mathbf{w}_i\|^2 = \|\mathbf{u}_i, \mathbf{v}_i\|^2 = \|\mathbf{u}_i\| + \|\mathbf{v}_i\|$*. The loss functions* L_i *are lower bounded uniformly by a scalar* L_{inf}*.*

Assumption 11. *The function* $g_i(\mathbf{u}_i, x)$ *is* β_e *Lipschitz continuous and* β_g *smooth* $\forall i \in [M]$ *with respect to* $\mathbf{u}_i$*. Moreover, we assume that* $g_i(\mathbf{u}_i, x) \in [0,1]^{d_e}, \forall i$*.*

Assumption 12. *The empirical gradient is an unbiased gradient of the population loss i.e.,* $\mathbb{E}[\nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})] = \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$*. The variance of the stochastic gradient is bounded, i.e.,* $\mathbb{E}\Big[\Big\| \nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) - \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) \Big\|^2 \Big] \leq G_1 \Big\| \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) \Big\|^2 + G_2^2$*.*

A.2.1 Proof of Convergence Result

We now state some useful lemmas towards showing the convergence of the proposed algorithm. Unless otherwise specified, $\|\cdot\|$ is $\|\cdot\|_2$ for vectors and $\|\cdot\|_{\text{F}}$ for matrices. When we consider $\mathbf{a}$, we assume that all $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}$ are vectorized and concatenated such that $\mathbf{a} \in \mathbb{R}^{Md_i + 2(d_e+1)K}$.

Lemma 13. *When Assumptions 10,11 are satisfied, the function* $\Phi(\mathbf{a})$ *is* C_0-*smooth function wrt* $\mathbf{a}$ *with* $C_0 = \Big(\beta_s + \lambda_2 K \beta_e(\beta_e + 1) + \lambda_2 K \beta_g \sqrt{d_e} + 4\lambda_1 + \sqrt{K} \lambda_2 (1 + \beta_e) \Big)$*.*

Proof. We will show that $\Phi(\mathbf{a})$ is smooth wrt $\mathbf{a}$ by showing that $\nabla_{\mathbf{a}}\Phi$ is Lipschitz, i.e. $\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}')\| \leq C_0\|\mathbf{a} - \mathbf{a}'\|$. Consider $\mathbf{a} = (\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$ and $\mathbf{a}' = \{\mathbf{w}_i'\}_{i=1}^M, \bar{\mathbf{v}}', \mathbf{e}'$. To show this, we need to show that $\nabla\Phi$ is Lipschitz with respect to $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}$ and $\mathbf{e}$.

Firstly, observe that $\nabla_{e^k}\Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) = \lambda_2 \sum_{i=1}^M \frac{p_i^k}{b_i^k} \sum_{x\in\mathcal{B}_i^k}(g_i(\mathbf{u}_i, x) - e^k)$. Clearly,

$$\begin{aligned}\|\nabla_{e^k}\Phi(\mathbf{a}) - \nabla_{e^k}\Phi(\mathbf{a}')\| &\leq \lambda_2\|\mathbf{e} - \mathbf{e}'\| + \sum_{i=1}^M \sum_{x\in\mathcal{B}_i^k} \frac{\lambda_2 p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - g_i(\mathbf{u}_i', x)\| \\ &\leq \lambda_2(\|\mathbf{e} - \mathbf{e}'\| + \beta_e\|\mathbf{u}_i - \mathbf{u}_i'\|) \leq \lambda_2(1 + \beta_e)\|\mathbf{a} - \mathbf{a}'\|.\end{aligned}$$

where the second inequality follows from the Lipschitz property of g_i in Assumption 11, and since $\|e^k - e'^k\| \leq \|\mathbf{e} - \mathbf{e}'\| \leq \|\mathbf{a} - \mathbf{a}'\|$ and $\|\mathbf{u}_i - \mathbf{u}_i'\| \leq \|\mathbf{a} - \mathbf{a}'\|$. Therefore, we can write

$$\|\nabla_{\mathbf{e}}\Phi(\mathbf{a}) - \nabla_{\mathbf{e}}\Phi(\mathbf{a}')\|^2 = \sum_{k=1}^K \|\nabla_{e^k}\Phi(\mathbf{a}) - \nabla_{e^k}\Phi(\mathbf{a}')\|^2 \leq K\lambda_2^2(1 + \beta_e)^2\|\mathbf{a} - \mathbf{a}'\|^2. \tag{A.4}$$

Therefore, Φ is smooth in $\mathbf{e}$ with parameter $\lambda_2(1 + \beta_e)\sqrt{K}$. Similarly, consider $\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) = \sum_{i=1}^M p_i\lambda_1(\bar{\mathbf{v}} - \mathbf{v}_i)$. Therefore,

$$\|\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) - \nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}')\| \leq \lambda_1\|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\| + \lambda_1\sum_{i=1}^M p_i\|\mathbf{v}_i - \mathbf{v}_i'\| \leq 2\lambda_1\|\mathbf{a} - \mathbf{a}'\|, \tag{A.5}$$

since $\|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\| \leq \|\mathbf{a} - \mathbf{a}'\|$ and $\|\mathbf{v}_i - \mathbf{v}_i'\| \leq \|\mathbf{a} - \mathbf{a}'\|$. Therefore Φ is $2\lambda_1$ smooth in $\bar{\mathbf{v}}$.

Now, let us show that Φ is smooth in every $\mathbf{w}_i$. Clearly, $\nabla_{\mathbf{w}_i}\Phi(\mathbf{a}) = \nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e})$. Therefore, let us consider smoothness of Φ_i with respect to $\mathbf{w}_i$. Since $\mathbf{w}_i = (\mathbf{u}_i, \mathbf{v}_i)$, note that

$$\begin{aligned}&\nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}_i, \mathbf{e}) \\ &= p_i\nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i) + \nabla_{\mathbf{w}_i}\Big(p_i\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2 + \frac{\lambda_2}{2}\sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) \\ &= p_i\nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i) + \begin{bmatrix}\nabla_{\mathbf{u}_i}\Big(\frac{\lambda_2}{2}\sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) \\ p_i\nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2\big)\end{bmatrix}.\end{aligned} \tag{A.6}$$

We know from the smoothness of L_i from Assumption 10 that $\|\nabla_{\mathbf{w}_i} L_i(\mathbf{w}_i) - \nabla_{\mathbf{w}_i} L_i(\mathbf{w}_i')\| \leq \beta_s \|\mathbf{w}_i - \mathbf{w}_i'\|$. Also, it is easy to note that $\|\nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2\big) - \nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i' - \bar{\mathbf{v}}'\|^2\big)\| \leq (\lambda_1\|\mathbf{v}_i - \mathbf{v}_i'\| + \|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\|) \leq 2\lambda_1\|\mathbf{a} - \mathbf{a}'\|$.

Now consider the term

$$\nabla_{\mathbf{u}_i}\Big(\frac{\lambda_2}{2}\sum_{k=1}^{K}\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) = \sum_{k=1}^{K}\sum_{x\in\mathcal{B}_i^k}\frac{\lambda_2 p_i^k}{b_i^k}\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k).$$

We can write

$$\begin{aligned}
&\Big\|\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k) - \nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i', x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\
&\overset{(i)}{\leq} \Big\|\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k) - \nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\
&\qquad + \Big\|\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i', x) - e^{'k}) - \nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i', x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\
&\overset{(ii)}{\leq} \|\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)\|\|(g_i(\mathbf{u}_i, x) - e^k) - (g_i(\mathbf{u}_i', x) - e^{'k})\| \\
&\qquad + \|\nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i, x)^T - \nabla_{\mathbf{u}_i} g_i(\mathbf{u}_i', x)\|\|(g_i(\mathbf{u}_i', x) - e^{'k})\| \\
&\overset{(iii)}{\leq} \beta_e(\beta_e\|\mathbf{u}_i - \mathbf{u}_i'\| + \|\mathbf{e} - \mathbf{e}'\|) + \beta_g\|\mathbf{u}_i - \mathbf{u}_i'\|\sqrt{d_e} \\
&\overset{(iv)}{\leq} \Big(\beta_e(\beta_e + 1) + \beta_g\sqrt{d_e}\Big)\|\mathbf{a} - \mathbf{a}'\|.
\end{aligned} \tag{A.7}$$

where (i) follows from adding and subtracting a term and applying triangle inequality, (ii) follows from submultiplicativity of the norm, (iii) follows from the smoothness and Lipschitz properties of g_i from Assumption 11 and that $g_i(\mathbf{u}_i, x), e^k \in [0,1]^{d_e}$, and (iv) follows because $\|\mathbf{u}_i - \mathbf{u}_i'\| \leq \|\mathbf{a} - \mathbf{a}'\|$.

Substituting these smoothness results in (A.6), we get

$$\begin{aligned}
&\Big\|\nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) - \nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i', \bar{\mathbf{v}}', \mathbf{e}')\Big\| \\
&\leq \beta_s\|\mathbf{a}_i - \mathbf{a}_i'\| + \lambda_2 K\Big(\beta_e(\beta_e + 1) + \beta_g\sqrt{d_e}\Big)\|\mathbf{a} - \mathbf{a}'\| + 2\lambda_1\|\mathbf{a} - \mathbf{a}'\| \\
&\leq (\beta_s + \lambda_2 K\beta_e(\beta_e + 1) + \lambda_2 K\beta_g\sqrt{d_e} + 2\lambda_1)\|\mathbf{a} - \mathbf{a}'\|.
\end{aligned} \tag{A.8}$$

Therefore, finally from (A.4), (A.5), (A.8), we have

$$
\begin{aligned}
&\Big\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}')\Big\| \\
&\leq \sum_{i=1}^{M}\Big\|\nabla_{\mathbf{w}_i}\Phi(\mathbf{a}) - \nabla_{\mathbf{w}_i}\Phi(\mathbf{a}')\Big\| + \|\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) - \nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}')\| + \|\nabla_{\mathbf{e}}\Phi(\mathbf{a}) - \nabla_{\mathbf{e}}\Phi(\mathbf{a}')\| \\
&\leq \Big((\beta_s + \lambda_2 K\beta_e(\beta_e+1) + \lambda_2 K\beta_g\sqrt{d_e} + 2\lambda_1) + 2\lambda_1 + \sqrt{K}\lambda_2(1+\beta_e)\Big)\|\mathbf{a} - \mathbf{a}'\|.
\end{aligned} \tag{A.9}
$$

Hence, Φ is C_0-smooth in $\mathbf{a}$. □

Note that, we use boundedness of $g_i(\mathbf{u}_i, x), e^k$ in (A.7)(iii) to show that this term is smooth. However, it is not necessary to assume that $g_i(\mathbf{u}_i, x)$ is bounded in Assumption 11. Instead, we can modify the loss as

$$
\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) = L_i(\mathbf{w}_i) + \frac{\lambda_1}{2}p_i\Big\|\mathbf{v}_i - \bar{\mathbf{v}}\Big\|_2^2 + \frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\Big\|\sigma\big(g_i(\mathbf{u}_i, x)\big) - e^k\Big\|_2^2, \tag{A.10}
$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the element wise sigmoid function with values in $[0,1]$. The update equation in (A.3) then becomes

$$
e^k_{(t+1)} = e^k_{(t)}(1 - \lambda_2\eta_{(t)}) + \lambda_2\eta_{(t)}\left(\sum_{i=1}^{M}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\sigma\big(g_i(\mathbf{u}_{(i,t)}, x)\big)\right).
$$

The smoothness result in (A.7) follows with a little more effort.

Lemma 14. *When Assumptions 10, 11, 12 are satisfied, we have*

$$
eq \leq \Phi(\mathbf{a}_{(t)}) - \left(\eta_{(t)} - \frac{\eta_{(t)}^2 C_0(G_1+1)}{2}\right)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0 G_2^2}{2} \tag{A.11}
$$

where $\mathcal{F}_t$ is a filtration accounting for the randomness (in sampling the mini-batches) until time t.

Proof. Using the smoothness property of Φ from Lemma 13, we can write

$$
\Phi\big(\mathbf{a}_{(t+1)}\big) \leq \Phi\big(\mathbf{a}_{(t)}\big) + \nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\big(\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\big) + \frac{C_0}{2}\|\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\|^2.
$$

Taking expectation with respect to randomness at time t, we get

$$\begin{aligned}
&\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] \\
&\leq \Phi(\mathbf{a}_{(t)}) + \mathbb{E}\Big[\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\big(\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\big)|\mathcal{F}_t\Big] + \frac{C_0}{2}\mathbb{E}\Big[\|\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\|^2|\mathcal{F}_t\Big] \\
&\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\mathbb{E}\Big[\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)})|\mathcal{F}_t\Big] + \frac{\eta_{(t)}^2 C_0}{2}\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)})\|^2|\mathcal{F}_t\Big] \\
&\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0}{2}\Big(\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2|\mathcal{F}_t\Big] \\
&\qquad\qquad + \|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\Big) \\
&\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0}{2}\Big((G_1+1)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + G_2\Big) \\
&\leq \Phi(\mathbf{a}_{(t)}) - \left(\eta_{(t)} - \frac{\eta_{(t)}^2 C_0(G_1+1)}{2}\right)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0 G_2^2}{2}
\end{aligned} \tag{A.12}$$

where the second inequality follows from the update rule and the third inequality follows from Assumption 12. □

We state the theorem here for completeness.

Theorem 7. *Let Assumptions 10, 11, and 12 hold and run the algorithm with T timesteps. If we chose a constant learning rate $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ that satisfies $0 < \eta \leq \min\left\{\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \frac{1}{C_0(G_1+1)}\right\}$, where $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, and C_0 is as defined in Lemma 13, then we have $\min_{t:1,\cdots,T}\mathbb{E}\big[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\big] \leq \sqrt{\frac{2C_1C_0G_2^2}{T}}$.*

Proof. Taking expectation on both sides of (A.12) from Lemma 14 and since $\frac{\eta C_0(G_1+1)}{2} \leq \frac{1}{2}$, we have

$$\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})\Big] \leq \mathbb{E}\Big[\Phi(\mathbf{a}_{(t)})\Big] - \frac{\eta}{2}\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\Big] + \frac{\eta_{(t)}^2 C_0 G_2^2}{2}.$$

By summation over $t = 1$ to T followed by telescoping the sum and rearranging, we can write

$$\sum_{t=1}^{T} \frac{\eta}{2}\mathbb{E}[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] + \sum_{t=1}^{T} \eta^2\Big(\frac{C_0 G_2^2}{2}\Big).$$

Choosing a constant learning rate η and dividing both sides by T, we get

$$\sum_{t=1}^{T} \frac{\mathbb{E}[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2]}{T} \leq \frac{2\Big(\mathbb{E}[\Phi(\mathbf{a}_{(1)})]) - \mathbb{E}[\Phi(\mathbf{a}_{(T)})]\Big)}{\eta T} + \eta C_0 G_2^2.$$

Now, if $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, we can choose $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ which results in convergence to a first order stationary point with a rate of $\frac{1}{\sqrt{T}}$. $\square$

Observe that the rate depends on C_0 where $C_0 = \Big(\beta_s + \lambda_2 K\beta_e(\beta_e+1) + \lambda_2 K\beta_g\sqrt{d_e} + 4\lambda_1 + \sqrt{K}\lambda_2(1+\beta_e)\Big)$. For non-zero values of λ_1 and λ_2, the convergence rate gets affected. For non-zero values of λ_2 (the moment alignment term), the convergence rate slows down for high values of K and d_e. We study the performance of the algorithm by varying the size of the embedding dimension d_e in the experimental section.

A.2.2 Continuing FL in the latent space

Let $\big(\{(\mathbf{u}_i^*, \mathbf{v}_i^*)\}_{i=1}^M, \bar{\mathbf{v}}^*, \mathbf{e}^*\big)$ be the output from solving (2.8). We now freeze the representation functions $\{\mathbf{u}_i^*\}_{i=1}^M$ for all the clients. For every dataset $D_i = \{x_j, y_j\}_{j=1}^{N_i}$, we define a projected dataset $\tilde{D}_i = \{\big(z_j, y_j\big) | z_j = g_i(\mathbf{u}_i^*, x_j)\ \forall (x_j, y_j) \in D_i\}$. We now define the local objective as $\arg\min F_i(\mathbf{v}) := \frac{1}{N_i}\sum_{(x,y)\in D_i; z=g_i(\mathbf{u}_i^*,x)} l(\mathbf{v}, z, y)$. The global objective is then defined as

$$\arg\min_{\mathbf{v}} F(\mathbf{v}) := \sum_{i=1}^{M} p_i F_i(\mathbf{v}). \tag{A.13}$$

To solve (A.13), we initialize with $\mathbf{v}_{(i,0)} = \bar{\mathbf{v}}^*$, follow the FedAvg algorithm [61] and update the local weights $\mathbf{v}_{(i,t)}$ at $i-$th worker and the global weights $\mathbf{v}$ as

$$\mathbf{v}_{(i,t+\tau+1)} = \mathbf{v}_{(i,t+\tau)} - \eta_{(t+\tau)}\nabla\widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)}); \qquad \mathbf{v}_{(t+E)} = \sum_{i=1}^{M} p_i \mathbf{v}_{(i,t+E)}, \tag{A.14}$$

where τ is the local iteration number, $\nabla \widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)})$ is the stochastic gradient at time $t+\tau$ by sampling a local minibatch $\xi_{(i,t+\tau)}$ and $\eta_{(t+\tau)}$ is the learning rate. After every E local updates, the server aggregates the local weights and for simplicity, we assume full device participation in updating the weights. After the global update, the server assigns $\mathbf{v}_{(i,t+E)} = \mathbf{v}_{(t+E)}$ to every client. Before stating the convergence theorem, we make the following assumptions.

Assumption 13. *The local loss functions F_i are all $L-$smooth and μ strongly convex for all $i \in [M]$.*

Assumption 14. *The second moment of the stochastic gradients of the $i-th$ worker at time t is bounded as: $\mathbb{E}\Big[\Big\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)}) - \nabla F_i(\mathbf{v}_{(i,t)})\Big\|^2\Big] \leq \sigma_i^2$, for all $i \in [M]$.*

Assumption 15. *The expected squared norm of the stochastic gradients is uniformly bounded as $\mathbb{E}\Big[\Big\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)})\Big\|^2\Big] \leq G^2$ for all $i \in [M]$.*

The problem we solve in (A.13) is convex since the cross entropy loss is convex and we are only optimizing the final layer (classification) weights $\mathbf{v}$ (keeping the $\mathbf{u}_i^*$ fixed). Moreover, we can assume that each local objective uses an L2 regularization term, thus satisfying strong convexity. The Assumptions 13, 14, 15 are standard for analyzing FedAvg for non-iid scenarios [61].

Definition 5 ([61])**.** *Let F^*, F_i^* be the minimum values of F and F_i respectively. The degree of non-iid (heterogeneity) is defined as $\Gamma = F^* - \sum_{i=1}^{M} p_i F_i^*$.*

When the data $\tilde{D}_i$ are all i.i.d, then the degree of heterogeneity Γ goes to zero as the sample size increases. Here, the degree of heterogeneity between $\{\tilde{D}_i\}_{i=1}^M$ is determined by the degree of heterogeneity between $\{D_i\}_{i=1}^M$ and the learned $\{\mathbf{u}_i^*\}_{i=1}^M$. We now state the convergence theorem.

Theorem 8 (Theorem 1 [61]). *Let Assumptions 13 to 15 hold and L, μ, σ_i, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_{(t)} = \frac{2}{\mu(\gamma+t)}$. Then solving* (A.14) *with full client participation satisfies $\mathbb{E}[F(\mathbf{v}_{(T)})] - F^* \leq \frac{\kappa}{\gamma+T-1}\Big(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}[\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2]\Big)$, where $B = \sum_{i=1}^{M} p_i^2\sigma_i^2 + 6L\Gamma + 8(E-1)^2G^2$, $\mathbf{v}^*$ is the unique minimizer of* (A.13).

Theorem 8 states that applying FedAvg to the problem (A.13), the resultant iterate $\mathbf{v}_{(T)}$ convergences with a rate of $\mathcal{O}(\frac{1}{T})$. The faster rate compared to Theorem 7 is due to the simplification of the problem to the strongly convex case by freezing the $\{\mathbf{u}_i\}_{i=1}^M$. Moreover, unlike Theorem 7, where the algorithm converges to a local optima, here the convergence is to the global optimum of (A.13). However, (A.13) itself depends on the frozen representations at which the latent space training took place. The rate of convergence slows with higher values of Γ, thus implying that the frozen $\{\mathbf{u}_i^*\}$ from solving (2.8) also play a role in the convergence of $\mathbf{v}_{(T)}$. Also, observe that the convergence rate depends on $\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2$. Suppose, the freezing is done when $\mathbf{u}_i$ are far away from minimizing (2.8), then $\bar{\mathbf{v}}^*$ is also far away from the optimal point and therefore it slows down convergence. Note that, while this dependence on $\mathbf{u}_i^*$ is not desirable from the convergence perspective, we want to highlight that we are dealing with model heterogeneity and this decoupling of $\mathbf{u}_i, \mathbf{v}_i$ is required to show a reasonable generalization result.

A.3 Generalization Result

A.3.1 Preliminaries

For this section, we consider the space $\mathcal{Z}$ as the input space and consider the prediction function $h(\mathbf{v}, \cdot) : \mathcal{Z} \to \mathcal{Y}$ for $\mathbf{v} \in \mathcal{V}$. We first present the divergence

between two distributions for a given model space. Note that, the notation is slightly changed to be consistent with the rest of the notation in this paper. This divergence is presented in [44] as $\mathcal{H}-$ divergence.

Definition 6 ([44]). *Let $\mathcal{A}_\mathcal{V}$ be the set of subsets of $\mathcal{Z}$ that are the support of some model in $\mathcal{V}$, i.e., $\mathcal{A}_\mathcal{V} = \{B \subset \mathcal{Z} : \forall z \in B, h(\mathbf{v}, z) = 1\}$. Given two distributions $\mathcal{D}, \mathcal{D}'$ on $\mathcal{Z}$, we define*

$$d_\mathcal{V}(\mathcal{D}, \mathcal{D}') = 2 \sup_{\mathcal{A} \in \mathcal{A}_\mathcal{V}} |\mathbb{P}_\mathcal{D}[\mathcal{A}] - \mathbb{P}_{\mathcal{D}'}[\mathcal{A}]| = 2 \sup_{\mathbf{v} \in \mathcal{V}} |\mathbb{P}_\mathcal{D}[\mathbb{I}(h(\mathbf{v}, \cdot))] - \mathbb{P}_{\mathcal{D}'}[\mathbb{I}(h(\mathbf{v}, \cdot))]|.$$

where $\mathbb{I}$ is the indicator function.

From the above definition for $\mathcal{H}-$ divergence, we can similarly write the divergence between two induced distributions $\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)$ induced by two different representation functions $\mathbf{u}_1, \mathbf{u}_2$ respectively as $d_\mathcal{V}(\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)) = 2 \sup_{\mathcal{A} \in \mathcal{A}_\mathcal{V}} |\mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}[\mathcal{A}] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}[\mathcal{A}]|$.

Definition 7 (Symmetric Hypothesis space). *Given a hypothesis space/model space $\mathcal{V}$, we define the symmetric hypothesis space $\mathcal{V}\Delta\mathcal{V}$ as follows*

$$g \in \mathcal{V}\Delta\mathcal{V} \iff g(z) = h(\mathbf{v}, z) \oplus h(\mathbf{v}', z), \text{ for some } \mathbf{v}, \mathbf{v}' \in \mathcal{V}.$$

The indicator function of such g is the set of all $z \in \mathcal{Z}$ where $\mathbf{v}, \mathbf{v}'$ disagree.

We can now derive the relation between the difference in loss on two distributions and the divergence between the distributions as follows.

Lemma 15. *For $\mathbf{v}, \mathbf{v}' \in \mathcal{V}$, given two distributions $\mathcal{D}_1, \mathcal{D}_2$ and the respective representation functions $\mathbf{u}_1, \mathbf{u}_2$, we have $|L_{1,\mathbf{u}_1}(\mathbf{v}, \mathbf{v}') - L_{2,\mathbf{u}_2}(\mathbf{v}, \mathbf{v}')| \leq \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)\big)$.*

Proof. Recall that $L_{1,\mathbf{u}_1}(\mathbf{v},\mathbf{v}') = \mathbb{E}_{z\sim\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\big[|h(\mathbf{v},z)-h(\mathbf{v}',z)|\big]$. Therefore,

$$\begin{aligned}
&|L_{1,\mathbf{u}_1}(\mathbf{v},\mathbf{v}') - L_{2,\mathbf{u}_2}(\mathbf{v},\mathbf{v}')| \\
&= \left|\mathbb{E}_{z\sim\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[|h(\mathbf{v},z)-h(\mathbf{v}',z)|\Big] - \mathbb{E}_{z\sim\tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[|h(\mathbf{v},z)-h(\mathbf{v}',z)|\Big]\right| \\
&= \left|\mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[\mathbb{I}_{\{z:h(\mathbf{v},z)\neq h(\mathbf{v}',z)\}}\Big] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[\mathbb{I}_{\{z:h(\mathbf{v},z)\neq h(\mathbf{v}',z)\}}\Big]\right| \\
&\leq \sup_{g\in\mathcal{V}\Delta\mathcal{V}}\left|\mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[\mathbb{I}(g)\Big] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[\mathbb{I}(g)\Big]\right| \\
&= \frac{1}{2}d_{\mathcal{V}\Delta\mathcal{V}}(\tilde{\mathcal{D}}_1(\mathbf{u}_1),\tilde{\mathcal{D}}_2(\mathbf{u}_2)).
\end{aligned}$$

The first two equalities follow from the definition. The inequality follows from the definition of the symmetric difference hypothesis space and taking a supremum. □

The following result gives a uniform convergence bound for the empirical weighted error across multiple sources relative to the true error.

Lemma 16 ([58]). *For each $i \in \{1,\cdots,M\}$, let D_i be a labeled sample of size $\beta_i N$ generated by drawing $\beta_i N$ points from $\mathcal{D}_i$ and labeling them according to c. For any fixed weight vector $\boldsymbol{\alpha}$, let $\hat{L}_{\boldsymbol{\alpha}}(\mathbf{w})$ be the empirical α-weighted error of some fixed model $\mathbf{w}$ on this sample, and let $L_{\boldsymbol{\alpha}}(\mathbf{w})$ be the true $\alpha-$weighted error. Then for any $\delta\in(0,1)$, with probability at least $1-\delta$,*

$$\mathbb{P}\Big[|\hat{L}_{\boldsymbol{\alpha}}(\mathbf{w}) - L_{\boldsymbol{\alpha}}(\mathbf{w})| \geq \epsilon\Big] \leq 2\exp\left(\frac{-2N\epsilon^2}{\sum_{i=1}^{N}\frac{\alpha_i^2}{\beta_i}}\right).$$

The following result provides a way to empirically estimate the $d_{\mathcal{V}\Delta\mathcal{V}}$ divergence based on the data samples.

Lemma 17 ([44],[58]). *Let $\mathcal{V}$ be a hypothesis space on $\mathcal{Z}$ with VC dimension $d_{\mathcal{V}}$. If $\mathcal{U}$ and $\mathcal{U}'$ are samples of size n from $\mathcal{D}$ and $\mathcal{D}'$ (two distributions in the space $\mathcal{Z}$)*

respectively. Then, for any $\delta \in (0,1)$ *with probability at least* $1-\delta$,

$$d_{\mathcal{V}\Delta\mathcal{V}}(\mathcal{D},\mathcal{D}') \leq 2\Big(1 - \min_{\mathbf{v}\in\mathcal{V}}\Big[\frac{1}{n}\sum_{z:h(\mathbf{v},z)=0}\mathbb{I}[x\in\mathcal{U}] + \frac{1}{n}\sum_{z:h(\mathbf{v},z)=1}\mathbb{I}[x\in\mathcal{U}']\Big]\Big) + 4\sqrt{\frac{d_{\mathcal{V}}\log(2n)+\log(\frac{2}{\delta})}{n}}.$$

We train a classifier to identify the distribution to which the data points belong. The divergence term is large, when the learnt function is able to classify properly between the distributions, and the divergence term is small when we are unable to learn a hypothesis that can distinguish between the distributions. We generalize this result to multiple number of clients and show the results in Fig 9 of main paper.

[49] provides a relation between the $d_{\mathcal{H}\Delta\mathcal{H}}$ divergence and two sample testing using MMD in Theorem 1 (eq 11). However, the bound appears vacuous, since the risk is bounded by 1 but there is a term with value 2 in eq 11.

A.3.2 Proof of main result

The generalization result for the model homogeneous case is a special case of the result for the heterogeneous case where $\mathbf{u}_i = \mathbf{u},\ \forall i \in [M]$. Therefore, we only show the proof for the model heterogeneous case. We now state the Theorem formally.

Theorem 9. *For private representations* $\{\mathbf{u}_i\}_{i=1}^M$ *and* j *as the target, let* $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v})$ *and* $\mathbf{v}_j^* = \arg\min_{\mathbf{v}\in\mathcal{V}} L_{j,\mathbf{u}_j}(\mathbf{v},\tilde{c}_{\mathbf{u}_j})$ *be the minimizer of the true target risk. Then for any* $\delta > 0$, *w.p.* $> 1-\delta$, *we have*

$$L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}},\tilde{c}_{\mathbf{u}_j}) - L_{j,\mathbf{u}_j}(\mathbf{v}_j^*,\tilde{c}_{\mathbf{u}_j}) \leq 4\sqrt{\sum_{i=1}^M \frac{\alpha_i^2}{p_i}}\sqrt{\frac{2d_{\mathcal{V}}\log(2(N+1))+\log(\frac{8}{\delta})}{N}}\Big) + 2\Big(\lambda_\alpha + \frac{1}{2}\sum_{i=1}^M \alpha_i d_{\mathcal{V}\Delta\mathcal{V}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i),\tilde{\mathcal{D}}_j(\mathbf{u}_j))\Big),$$

where $\lambda_\alpha = \sum_{i=1}^{M} \alpha_i \min\Big\{\mathbb{E}_{z\in\tilde{\mathcal{D}}_i(\mathbf{u}_i)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big], \mathbb{E}_{z\in\tilde{\mathcal{D}}_j(\mathbf{u}_j)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big]\Big\}$.

Proof. The proof follows that of Theorem 4 in [58] with some changes. Note that, we have induced distributions $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$ instead of the actual distributions $\mathcal{D}_i$. We write the result in terms of the induced distributions using Lemma 15 as follows. For every $i \in [M]$, we have $\forall \mathbf{v} \in \mathcal{V}$,

$$\begin{aligned} L_{j,\mathbf{u}_j}(\mathbf{v}, \tilde{c}_{\mathbf{u}_j}) &\leq L_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_j}) + \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_j(\mathbf{u}_j), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big) \\ &\leq L_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_i}) + L_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j}) + \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_j(\mathbf{u}_j), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big), \end{aligned}$$

where the first inequality follows from Lemma 15 and second inequality follows from triangle inequality. Similarly, by first using the triangle inequality and then Lemma 15, we can write,

$$L_{j,\mathbf{u}_j}(\mathbf{v}, \tilde{c}_{\mathbf{u}_j}) \leq L_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_i}) + L_{j,\mathbf{u}_j}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j}) + \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_j(\mathbf{u}_j), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Therefore, combining both the equations, we have

$$L_{j,\mathbf{u}_j}(\mathbf{v}) \leq L_{i,\mathbf{u}_i}(\mathbf{v}) + \min\{L_{j,\mathbf{u}_j}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j}), L_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j})\} + \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_j(\mathbf{u}_j), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big). \tag{A.15}$$

This is similar to (2) in the main paper, but we have the induced distributions and the induced labeling functions with respect to the private representations $\{\mathbf{u}_i\}_{i=1}^M$. This gives us the freedom to interpret the bounds in terms of the private representation functions of every client.

Now, we have

$$\big|L_{j,\mathbf{u}_j}(\mathbf{v}) - L_{\boldsymbol{\alpha}}(\mathbf{v})\big| = \big|L_{j,\mathbf{u}_j}(\mathbf{v}) - \sum_{i=1}^{M} \alpha_i L_{i,\mathbf{u}_i}(\mathbf{v})\big| \leq \sum_{i=1}^{M} \alpha_i \big|L_{j,\mathbf{u}_j}(\mathbf{v}) - L_{i,\mathbf{u}_i}(\mathbf{v})\big|,$$

and substituting (A.15), we get

$$\begin{aligned}\left|L_{j,\mathbf{u}_j}(\mathbf{v}) - L_{\boldsymbol{\alpha}}(\mathbf{v})\right| \leq & \underbrace{\sum_{i=1}^{M} \alpha_i \min\{L_{j,\mathbf{u}_j}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j}), L_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_j})\}}_{A} \\ & + \frac{1}{2} \underbrace{\sum_{i=1}^{M} \alpha_i \; d_{\mathcal{V}\Delta\mathcal{V}}\big(\tilde{\mathcal{D}}_j(\mathbf{u}_j), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big)}_{B}\end{aligned} \tag{A.16}$$

Now, taking the one side inequality, we get

$$L_{j,\mathbf{u}_j}(\mathbf{v}) \leq L_{\boldsymbol{\alpha}}(\mathbf{v}) + A + \frac{B}{2}, \tag{A.17}$$

and now we apply the usual recipe for uniform convergence for empirical risk minimizers (apply Lemma 16 to the $L_{\boldsymbol{\alpha}}(\mathbf{v})$ term and bound the growth function using the VC dimension $d_{\mathcal{V}}$). We get for $\delta \in (0,1)$, with probability $> 1 - \delta/2$,

$$L_{j,\mathbf{u}_j}(\mathbf{v}) \leq \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}) + 2\sqrt{\sum_{i=1}^{M} \frac{\alpha_i^2}{p_i} \left(\frac{2d_{\mathcal{V}} \log(2(N+1)) + \log(\frac{8}{\delta})}{N} \right)} + A + \frac{B}{2}. \tag{A.18}$$

Suppose $\alpha_i = p_i$, the bound can be written as (ignoring log terms)

$$L_{j,\mathbf{u}_j}(\mathbf{v}) \leq \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}) + 2\mathcal{O}\left(\sqrt{\frac{\log(\frac{1}{\delta})}{N}} \right) + A + \frac{B}{2}.$$

This bound shows us that, if we have good private representations $\{\mathbf{u}_i\}_{i=1}^{M}$, such that the terms A and B are small, then one can achieve a good generalization error by solving for $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v})$ where $\widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}) = \sum_{i=1}^{M} \alpha_i \widehat{L}_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_i})$. Compare this with the results in equations (3, 4) in the main paper; we allow model heterogeneity which is not possible in (3) and we show the $\frac{1}{\sqrt{N}}$ dependence which is not shown in (4). The minimize the bound in (3), we solve $\arg\min_{\mathbf{w}\in\mathcal{W}} \widehat{L}_{\boldsymbol{\alpha}}(\sum_{i=1}^{M} \alpha_i \mathbf{w}_i)$. However, since the objective is non-convex, it is not possible to guarantee convergence to a global optima.

However, while solving for $\arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v})$, we have fixed representation weights $\{\mathbf{u}_i\}_{i=1}^M$ which allows us to transform the problem into a strongly-convex objective (refer to Section A.2.2) which can guarantee the convergence of $\arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v})$ to the global minima.

We shall now show the bound in terms of $\mathbf{v}_j^* := \arg\min_{\mathbf{v}\in\mathcal{V}} L_{\boldsymbol{\alpha}}(\mathbf{v})$ which is the minimizer (in the latent space) of the combined induced true risk (due to the private representations $\{\mathbf{v}_i\}_{i=1}^M$). Since $\widehat{\mathbf{v}}$ is the empirical minimizer of $\widehat{L}_{\boldsymbol{\alpha}}$, we have $\widehat{L}_{\boldsymbol{\alpha}}(\widehat{\mathbf{v}}) \leq \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}_j^*)$, then with probability $> 1 - \delta/2$,

$$L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}}) \leq \widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}_j^*) + 2\sqrt{\sum_{i=1}^M \frac{\alpha_i^2}{p_i}\left(\frac{2d_{\mathcal{V}}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\right)} + A + \frac{B}{2}.$$

Applying Lemma 16 again to $\widehat{L}_{\boldsymbol{\alpha}}(\mathbf{v}_j^*)$, with probability $> 1 - \delta$,

$$L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}}) \leq L_{\boldsymbol{\alpha}}(\mathbf{v}_j^*) + 4\sqrt{\sum_{i=1}^M \frac{\alpha_i^2}{p_i}\left(\frac{2d_{\mathcal{V}}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\right)} + A + \frac{B}{2}.$$

Finally, substituting $\mathbf{v}_j^*$ in (A.16) (taking the one sided inequality), we get with probability $> 1 - \delta$,

$$L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}}) \leq L_{j,\mathbf{u}_j}(\mathbf{v}_j^*) + 4\sqrt{\sum_{i=1}^M \frac{\alpha_i^2}{p_i}\left(\frac{2d_{\mathcal{V}}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\right)} + 2A + B.$$

The proof is complete. □

Please note that, in the result, we show the error $L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}})$ relative to the true error (induced by $\mathbf{u}_j$) $L_{j,\mathbf{u}_j}(\mathbf{v}_j^*)$. However, $L_{j,\mathbf{u}_j}(\mathbf{v}_j^*)$ may not be optimal if $\mathbf{u}_j$ is not a good representation. Let $\mathbf{w}^* = \arg\min L_j(\mathbf{w})$ where $\mathbf{w}^* := (\mathbf{u}_j^*, \mathbf{v}_j^{**})$. Then, using

Lemma 15, we can write,

$$L_{j,\mathbf{u}_j}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*}) - L_{j,\mathbf{u}_j^*}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*}) \leq \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\Big(\tilde{D}_j(\mathbf{u}_j), \tilde{D}_j(\mathbf{u}_j^*)\Big) \tag{A.19}$$

$$L_{j,\mathbf{u}_j}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*}) \leq L_{j,\mathbf{u}_j^*}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*}) + \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\Big(\tilde{D}_j(\mathbf{u}_j), \tilde{D}_j(\mathbf{u}_j^*)\Big) \tag{A.20}$$

$$L_{j,\mathbf{u}_j}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j}) \leq L_{j,\mathbf{u}_j^*}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*}) + L_{j,\mathbf{u}_j}(\tilde{c}_{\mathbf{u}_j}, \tilde{c}_{\mathbf{u}_j^*}) \tag{A.21}$$

$$+ \frac{1}{2} d_{\mathcal{V}\Delta\mathcal{V}}\Big(\tilde{D}_j(\mathbf{u}_j), \tilde{D}_j(\mathbf{u}_j^*)\Big). \tag{A.22}$$

The first inequality follows from Lemma 15, the third inequality follows from triangle inequality. Therefore, we can write the result of the theorem as, with probability at least $1 - \delta$,

$$\begin{aligned} L_{j,\mathbf{u}_j}(\widehat{\mathbf{v}}) \leq & \underbrace{L_{j,\mathbf{u}_j^*}(\mathbf{v}_j^*, \tilde{c}_{\mathbf{u}_j^*})}_{\text{True error}} \\ & + \underbrace{L_{j,\mathbf{u}_j}(\tilde{c}_{\mathbf{u}_j}, \tilde{c}_{\mathbf{u}_j^*})}_{\text{distance between labeling functions induced by } \mathbf{u}_j, \mathbf{u}_j^*} \\ & + \frac{1}{2} \underbrace{d_{\mathcal{V}\Delta\mathcal{V}}\Big(\tilde{D}_j(\mathbf{u}_j), \tilde{D}_j(\mathbf{u}_j^*)\Big)}_{\text{divergence between distributions induced by } \mathbf{u}_j, \mathbf{u}_j^*} \\ & + 4\mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{N}}\right) + 2A + B. \end{aligned}$$

This shows that, the representations should be learnt such that $\mathbf{u}_j$ is closer to $\mathbf{u}_j^*$ and also to minimize the term B.

A.4 Experiments

In this section, we explain the experimental setup and provide additional analysis to understand the algorithm. Specifically, we aim to answer the following questions.

1. Is there an advantage of minimizing MMD on the conditional distributions instead of the marginal distributions? See Section A.4.2.

2. Is communicating both $\bar{\mathbf{v}}$ and $\mathbf{e}$ with the server important for the improved performance? What happens if only one of them is communicated? See Section A.4.3.

3. How does the dimension d_e of the embedding space affect the algorithm? See Section A.4.5.

Before we address these questions, we first describe the experimental setup and the choice of the hyperparameters in Section A.4.1. We will revisit some of these questions again for the synthetic data experiments and provide a more thorough summary in Section A.4.6.

A.4.1 Experimental Setup

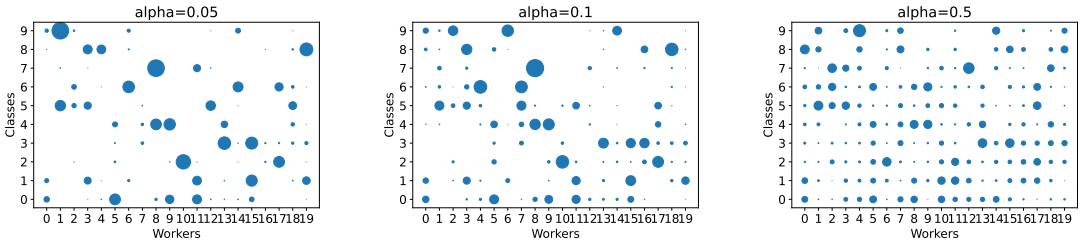


Figure A.1: Dirichlet non-iid split for 20 workers for different values of α

MNIST, EMNIST, FEMNIST

In this section, we provide the setup for comparing Fed-CMA with FedGen, FedAvg and Local training. We follow the training setup of FedGen [40] available at `https://github.com/zhuangdizhu/FedGen`. FedGen uses a batch size of 32 and performs 20 local iterations at every client before the global round. For MNIST

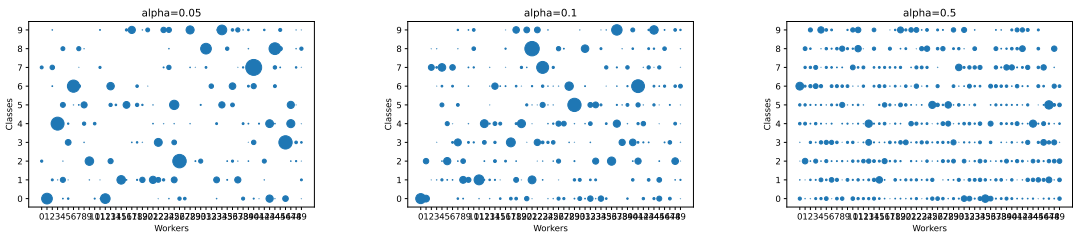
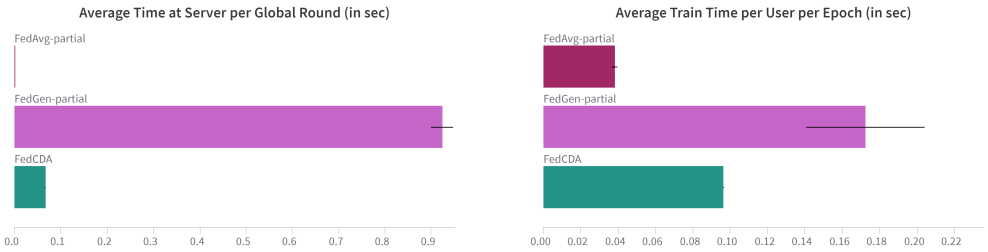


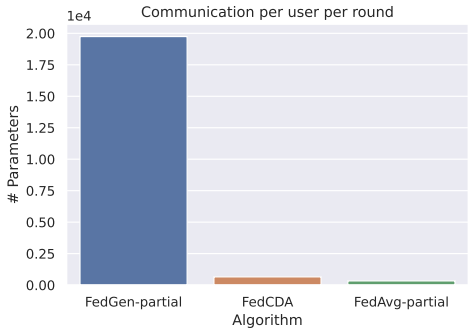
Figure A.2: Dirichlet non-iid split for 50 workers for different values of α

and EMNIST, we only consider 20 clients, and assume full client participation. For FEMNIST however, we assume that roughly 50% of the 923 clients participate at every round. We follow the same architecture used in FedGen here: `Conv2d(6,3,2)` – `BatchNorm2d()` – `ReLU` – `Conv2d(16,3,2)` – `BatchNorm2d`() – `ReLU` – `Flatten` – `Linear(784,32)` – `Linear(32,10)`. Here the latent space dimension is 32. Note that all the users have the same architecture in this experiment (following FedGen) but the users do not share the feature extractor weights and can only share the classification layer (i.e.,`Linear(32,10)`). We run every algorithm for 400 global iterations and three random seeds. We present the results of our experiments in Table 2 of main paper. To highlight that every algorithm only shares the final classification layer with the server, we denote FedAvg-partial, FedGen-partial, FedCMA-partial. We also provide a comparison of the runtimes and communication complexity for the algorithms in Fig A.3. FedGen-partial takes the most time at both the server and user, because it involves training a generator. Fed-CMA has only a small runtime overhead as compared to FedAvg since we compute the class conditional mean embeddings and update the global variables. FedGen-partial needs to communicate the generator between the users and the server, whereas in Fed-CMA we only need to communicate the final layer



(a) Local Training

(b) Fed ($\bar{\mathbf{v}}$)



(c) Communication Complexity

Figure A.3: Comparison of runtimes and communication complexity of Fed-CMA , FedAvg-partial and FedGen-partial.

weights and the mean embeddings resulting in an improved communication complexity. Please note that while the convergence analysis uses one local update and one global update simultaneously, in the experiments we consider multiple local updates per global update. Showing the convergence result for the multiple local update case is considered as future work.

CIFAR-10 comparison with pFedHN

For comparison with [63], we follow the same data split, model architecture and hyperparameters followed in the pFedHN paper. While pFedHN considers just one client is available at every communication round, we consider full client participation for FedCMA and compare the amount of communication required during training.

Despite needing full client availability, FedCMA requires much less communication as compared to pFedHN.

CIFAR-10 with model heterogeneity

The open source code for pFedHN does not include training with model heterogeneity. Therefore, to perform empirical analysis for the model heterogeneous case, we omit pFedHN and consider the following setup. To simulate **data heterogeneity**, we follow [62] where a Dirichlet distribution $\text{Dir}_M(\alpha)$ is used to partition the CIFAR-10 dataset [181] among M clients for $M \in \{10, 20, 50, 100\}$. Larger values of α indicate a more homogeneous distribution (Fig A.1,A.2). While allotting the train dataset to each user, we allot 90% of the data as train data and 10% of the data as validation data randomly uniformly. Note that the train data and validation data have the same distribution which is different than the test distribution (since in the test distribution all labels have equal number of data points).

In the experiments, we assume full user availability (i.e., every user participates at every global update), however each user runs a different number of local updates due to the difference in the size of the datasets. This simulates user heterogeneity in terms of the number of local updates at every global update. In the implementation, instead of weighing the local losses L_i with p_i, we give an equal weightage of $\frac{1}{M}$ to avoid changing the hyperparameters for every different choice of $M, \alpha,$ `seed`.

Each client's test set is comprised of the original test data points for all the classes in the client's training set. To simulate **model heterogeneity,** we consider the four networks: `CNN1`: `Conv2D(3,6,5)`$-$`Conv2D(6,16,5)`$-$`Lin(400,120)`$-$`Lin(120,84)`$-$`Lin(84,10)`, `CNN2`: `Conv2D(3,6,5)`$-$`Conv2D(6,16,5)`$-$`Lin(400,84)`$-$`Lin(84,10)`, `MLP1`: `Lin(924,512)`$-$`Lin(512,84)`$-$`Lin(84,10)`, `MLP2`: `Lin(924,256)`$-$`Lin(256,84)`$-$

`Lin(84,10)`. In all the networks, we use `sigmoid` in the last but one layer to keep the embeddings bounded and `ReLU` activation for all other layers. We consider two setups: (i) **Setup I** clients use either `CNN1` or `CNN2` and (ii) **Setup II** clients use any of the four networks at random, i.e., Setup I: the clients models $\in \{\texttt{LeNet1}, \texttt{LeNet2}\}$. Setup II: the clients models $\in \{\texttt{LeNet1}, \texttt{LeNet2}, \texttt{TwoLayer1}, \texttt{TwoLayer2}\}$. For each setup, we run Fed-CMA and local training in which the clients do not communicate. We compute the average of the local classification accuracy of all the clients . We repeat the experiments for three seeds and present the summary in Table A.1.

We can observe that for different levels of data heterogeneity, Fed-CMA outperforms local training. The improvement achieved by Fed-CMA in Setup II is less due to higher model heterogeneity since the capacity of the function class of CNNs and MLPs are different and therefore the alignment of distributions in the latent space shared between these networks is a tougher task than when the models consisted only of CNNs. In Figure A.4a, we plot the histogram of improvement in classification accuracy of every client (compared to local training) for Setup I with 100 clients and $\alpha = 0.1$. We observe that almost every client benefits from participating in the FL setup as compared to training locally. **Number of clients:** In Figure A.4b, we measure the norm of the updates in (A.3) to understand how fast a consensus is reached on $\mathbf{e}$ and $\bar{\mathbf{v}}$ and we observe that as the number of clients increases, the norm of the updates converges faster and to a smaller value showing the advantage of having more clients in the setup. **Communication:** In KD based approaches (such as KT-pFL [33] that communicate logits), the number of parameters communicated per client per communication round is $K \times |P|$ where P is the public dataset (KT-pFL uses $|P| = 3000$); for weight aggregation it is $\approx dim(\mathcal{H}_i)$; and for Fed-CMA it is $2K \times d_e$.

		Setup I		Setup II	
M	α	Local	Fed-CMA	Local	Fed-CMA
10	0.05	51.9 (0.6)	**54.1** (0.8)	39.3 (12.6)	**40.3** (13.0)
10	0.10	50.3 (0.2)	**52.9** (0.5)	38.3 (12.6)	**39.1** (13.3)
20	0.05	50.5 (0.1)	**53.9** (0.9)	47.8 (0.9)	**48.7** (1.1)
20	0.10	40.3 (1.9)	**45.7** (1.9)	36.5 (1.2)	**39.6** (0.9)
50	0.05	56.0 (0.4)	**59.8** (0.2)	52.8 (0.7)	**55.6** (0.8)
50	0.10	42.1 (0.3)	**47.5** (0.4)	38.0 (1.0)	**40.9** (0.9)
100	0.05	58.8 (0.4)	**63.1** (0.3)	56.7 (0.3)	**59.4** (0.4)
100	0.10	43.3 (0.1)	**48.0** (0.1)	41.0 (0.4)	**44.3** (0.5)

Table A.1: Summary of the results on CIFAR-10 dataset.

In general, d_e is much smaller than $dim(\mathcal{H}_i)$ and is comparable or smaller than $|P|$, hence very little communication overhead for Fed-CMA .

We implemented Fed-CMA using Pytorch and can be easily used with other existing algorithms. We used an Nvidia Quadro RTX 6000 GPU for the simulations, and the GPU RAM required to run any experiment is less than 2GB and can be run on even smaller GPUs. We simulate the distributed setup regarding dataset availability, but run the user updates sequentially so that the experiments can finish on a single GPU without requiring a cluster (since we run for different numbers of users upto 100). We will include the data splits and the code upon acceptance. Please note that, we use this experimental setup and use the CIFAR-10 dataset in all the experiments, unless otherwise specified. For all other ablations in the next sections, we use the setting with $M = 20, \alpha = 0.1, \texttt{seed} = 0$, and Setup I model heterogeneity.

To choose the hyperparameters, we selected the setup ($M = 20, \alpha = 0.1, \texttt{seed} = 0$) and used the model heterogeneous Setup I. We can this as the `hyperparam` setup. We fixed the batch size for the experiments to be 256. We chose the learning rate as 0.01 by performing a grid search over $[0.001, 0.01, 0.1]$ during Local only training and selected the value with the best validation accuracy. These hyperparameters are then fixed for all different setups (varying M, α, `seed` and for Setup I, II).

We chose $\mathcal{B}_i^k$ at the beginning by randomly sampling a mini-batch of size 256 from the training dataset of each worker. Then we split the chosen minibatch for every class k. We chose the values λ_1, λ_2 through hyperparameter selection by training Fed-CMA for the `hyperparam` setup. We performed grid search over the values $\lambda_1 \in [0.5, 1, 5]$ and $\lambda_2 \in [0.5, 1, 5]$. We chose a different learning rate for the user model parameters ($\eta = 0.01$) and the global variables $\bar{\mathbf{v}}, \mathbf{e}$. The learning rate of λ_1, denoted by η_{λ_1} is fixed to be 0.2 and the learning rate of λ_2, denoted by η_{λ_2} is fixed to be 0.18. These values were chosen such that $\eta_{\lambda_1}\lambda_1 \in [0, 1]$ and $\eta_{\lambda_2}\lambda_2 \in [0, 1]$.. The values for λ_1 and λ_2 are selected such that the average validation accuracy across all the clients is higher (good classification accuracy) and simultaneously $\|\mathbf{e}_{t+1} - \mathbf{e}_t\|, \|\bar{\mathbf{v}}_{t+1} - \bar{\mathbf{v}}_t\|$ are small (reach consensus on $\mathbf{e}, \bar{\mathbf{v}}$). The selected values $\lambda_1 = 5.0$, $\lambda_2 = 1.0$ are used throughout all the CIFAR-10 experiments in this paper.

A.4.2 Conditional Distribution Alignment vs Marginal Distribution Alignment

In this section, we shall examine the importance of performing conditional distribution alignment as compared to just aligning the marginal distributions of the representations. For this experiment, we consider the setting with 20 users, $\alpha = 0.1$, and use the Setup I model heterogeneity. The rest of the experiment setup is the

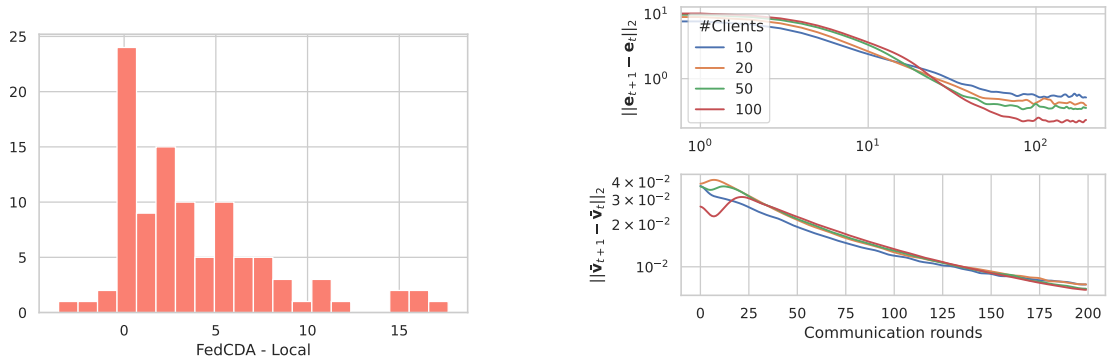


Figure A.4: (a) Histogram of per worker improvements for Setup I, (b) Norm of the global updates

same as described in Section A.4.1, and we run only for one seed. From Table A.2, we can observe that minimizing MMD on marginal distributions does not improve the performance, whereas minimizing MMD on conditional distributions provides a boost in performance.

	#Clients	α	Average Accuracy
Conditional (Fed-CMA)	20	0.1	**47.16**
Marginal	20	0.1	39.0
Local	20	0.1	41.6

Table A.2: Comparison of conditional distribution alignment and marginal alignment

A.4.3 Importance of $\bar{v}$ and e

In this Section, we perform an ablation on sharing $\bar{\mathbf{v}}$ and $\mathbf{e}$ in Fed-CMA . We follow the same experiment setup described in Section A.4.2. In the algorithm where $\bar{\mathbf{v}}$ is not

	#Clients	α	Average Accuracy
Fed-CMA	20	0.1	**47.16**
Fed-CMA (e)	20	0.1	39.56
Fed ($\bar{v}$)	20	0.1	35.03
Local	20	0.1	41.6

Table A.3: Ablation on sharing only $\bar{v}$ and only $\mathbf{e}$.

shared (only $\mathbf{e}$ is shared), we only minimize the $\frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\left\|g_i(\mathbf{u}_i,x)-e^k\right\|_2^2$ along with L_i at each client. Note that $\bar{\mathbf{v}}$ is not updated at the global level since it does not play any role. We denote this algorithm by Fed-CMA ($\mathbf{e}$). Similarly, in the algorithm where $\mathbf{e}$ is not shared (we only share $\bar{\mathbf{v}}$), we only minimize $\lambda_1\|\bar{\mathbf{v}}-\mathbf{v}_i\|$ at each client. We denote this algorithm Fed($\bar{\mathbf{v}}$) since we are only sharing the final classification layer weights. We compare these two algorithms with Fed-CMA and Local training and present the results in Table A.3. We can observe that these two algorithms perform poorly compared to Local training only. This is possible since, in Fed ($\bar{\mathbf{v}}$), the latent space distributions of all the clients are not homogenized/aligned, therefore learning a classifier for these heterogeneous distributions does not result in a good classifier and the clients are better off not participating in the FL algorithm. In the case of Fed-CMA ($\mathbf{e}$), the latent space distributions are aligned, however the final classification layer weights are not shared, therefore the clients do not have a chance to improve their decision boundaries by using the classification weights of the other clients. Note that in this case, the performance is slightly affected compared to Local only training, and this may be avoided by a better choice of the hyperparameters.

A.4.4 Pathogenic non-iid split

We present the results for a pathogenic non-iid split here. We sample 10% of the CIFAR10 dataset and divide the dataset among two workers such that the first worker gets classes 0,1,2,3,4 and the second worker gets classes 5,6,7,8,9. The following table shows that even in this pathogenic non-iid scenario, FedCMA provides an improvement over local only training.

Num Workers	Local	FedCMA
2	74.6	76

Table A.4: Pathogenic non-iid split

A.4.5 Dimension of the embedding space

The latent space plays an important role in aligning the conditional distributions and subsequently learning the classification weights. Therefore, it is important to understand what impact the latent space has on the algorithm, especially since we are not sharing the feature extractor weights and the data is heterogeneous at all clients. We vary the dimension d_e of the latent space and evaluate Fed-CMA and Local training and present the results in Table A.5.

We also plot the norms of the updates to the global variables for different values of the latent space dimension d_e, i.e., we plot $\|\bar{\mathbf{v}}_{t+1} - \bar{\mathbf{v}}_t\|$ and $\|\mathbf{e}_{t+1} - \mathbf{e}_t\|$. We observe from Fig A.5, $\bar{\mathbf{v}}$ and $\mathbf{e}$ converge differently for different values of d_e. **Norm of the classifier update $\bar{\mathbf{v}}$**: For higher values of d_e, the convergence is faster for $\bar{\mathbf{v}}$. In other words, a richer space enables a faster convergence of the classification weights. **Norm**

Algorithm	8	16	32	64	84	128	200
Fed-CMA	43.7	46.21	47.18	46.72	47.17	45.29	45.57
Local	36.69	37.72	40.37	41.22	41.61	41.93	41.84

Table A.5: Varying the dimension of embedding space

of the mean embeddings update e: For higher values of d_e, the convergence is slower for $\mathbf{e}$. In other words, it is easier to obtain a consensus on the mean embeddings when the latent space is small. While a small latent space (small d_e) makes it easier to push the mean embeddings closer, it may not lead to a good classification boundary since the complexity of the hypothesis space is reduced, therefore the convergence of the classification weights is slowed down. On the other hand, for a relatively large d_e, the complexity of the hypothesis space is increased and the quality of the decision boundaries increase making the convergence of $\bar{\mathbf{v}}$ faster. However, since the latent space is now relatively larger, it becomes harder to bring the mean embeddings close to each other, slowing down the convergence of $\mathbf{e}$. This tradeoff in the convergence of the global variables shows us that choosing the latent space dimension is very important to the success of the algorithm. Note that, the performance of Fed-CMA is relatively stable across all the different values of d_e chosen as compared to that of Local training only.

A.4.6 Synthetic Data Experiments

We first generated four 2D Gaussian distributions with $\mu_1 = (0, 2.5), \Sigma_1 = [(0.5, 0), (0, 1)]$, $\mu_2 = (2.5, 0), \Sigma_2 = [(1, 0), (0, 0.5)]$, $\mu_3 = (-2, 2), \Sigma_3 = [(1, -0.5), (-0.5, 1)]$ and $\mu_4 = (2, 2), \Sigma_4 = [(0.5, 1), (1, 0.5)]$. We consider each Gaussian distribution to be

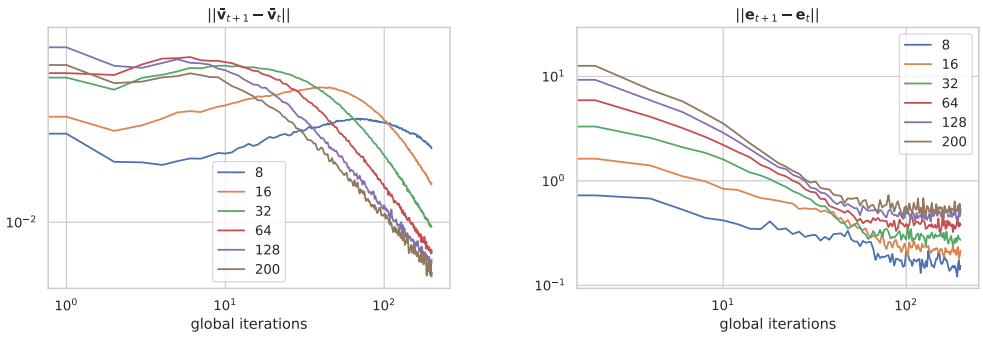
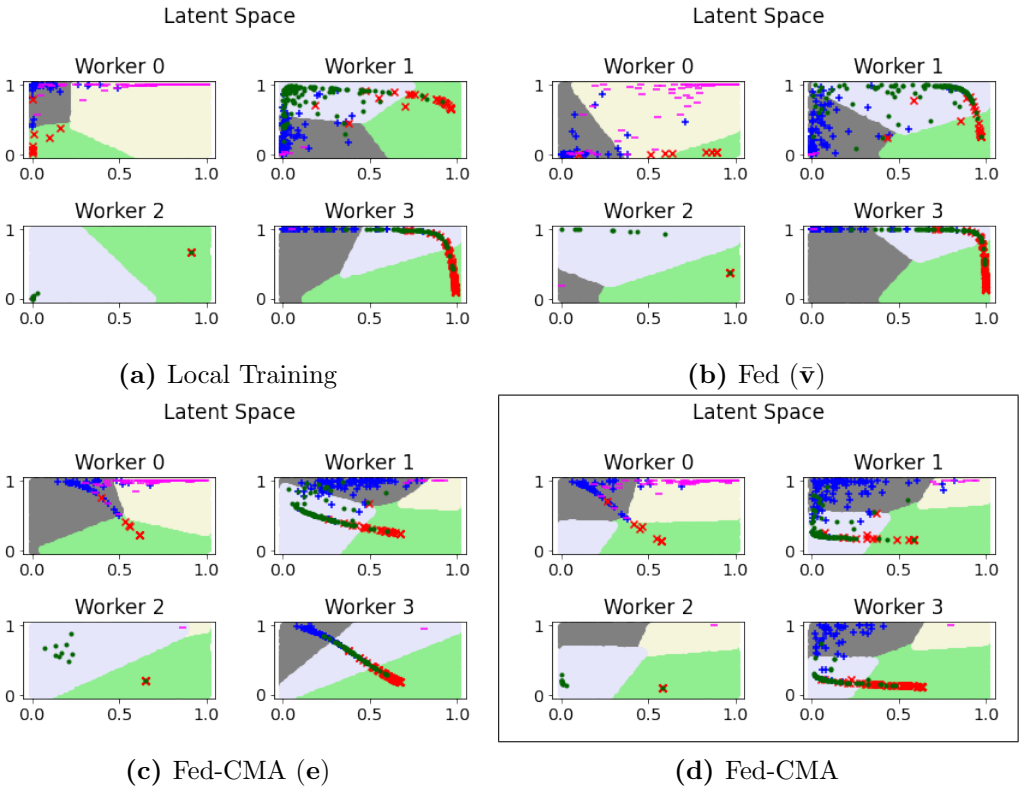


Figure A.5: Norm of the updates to $\bar{v}$ and e for different values of the dimension of the latent space d_e.

representing a class in a 4 class classification problem. The four classes are represented using the symbols [`+,x,-,.`]. The data is then split across the four workers using Dirichlet allocation as shown in Fig 4, 7 of the main paper. Note that, not all workers have the data of all four labels. In some extreme cases, some workers may posses very few data points of a particular class (see class `-` for Workers 2 & 3). We use `Linear(2,2)` $\rightarrow$ `Linear(2,2)` $\rightarrow$ `Linear(2,4)` at every client for simplicity. We train the algorithms for 3000 epochs. For every epoch, each worker has a different number of iterations since the number of data points at every worker is different. At the end of the epoch, the workers communicate with the server. We chose a learning rate of 0.1 and batch size of 64 for the synthetic experiments. We chose $\mathcal{B}_i^k$ at the beginning by randomly sampling a mini-batch of size 256 from the training dataset of each worker. Then we split the chosen minibatch for every class k. We chose $\lambda_1 = 0.01, \lambda_2 = 1.0$ for Fed-CMA . The learning rates for $\bar{\mathbf{v}}, \mathbf{e}$ are chosen separately as $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}$ respectively. We also train Fed-CMA ($\mathbf{e}$) and Fed ($\bar{\mathbf{v}}$) similar to Section A.4.3 and present the results in Table A.6.

Num Workers	Local	Fed ($\bar{\mathbf{v}}$)	Fed-CMA ($\mathbf{e}$)	Fed-CMA
4	53.15 (5.97)	52.65 (5.39)	55.58 (4.95)	**58.44** (7.6)

Table A.6: Summary of Synthetic data experiments



(a) Local Training

(b) Fed ($\bar{\mathbf{v}}$)

(c) Fed-CMA ($\mathbf{e}$)

(d) Fed-CMA

Figure A.6: Visualizing decision boundaries in latent space

Similar to the observations in Section A.4.3, we can see that sharing both $\mathbf{e}$ and $\bar{\mathbf{v}}$ is important to get a significant boost in performance. In this case however, Fed ($\bar{\mathbf{v}}$) and Fed-CMA ($\mathbf{e}$) perform better than Local training, but significantly worse than Fed-CMA . In Figures A.6, we illustrate the decision boundaries learnt by the respective clients. From Fig A.6a and A.6b, we can see that the latent space distributions of the

clients are not aligned properly. In Fig A.6c, the latent space distributions are aligned, however, since $\bar{\mathbf{v}}$ is not shared and the decision boundaries of every client are different and are overfit to their local training data. In Fig A.6d, the decision boundaries also align well along with the latent space distributions, thus resulting in a much better overall decision boundary in the input space (see Fig 7 of main paper). Observe that, for workers 2 & 3, even with very little data for the - class, the corresponding decision boundary is well formed and aligns with that of worker 0 who has an abundance of - class. Note that, for Fed ($\bar{\mathbf{v}}$), despite the latent space decision boundaries aligning roughly, the input space decision boundaries are poorly formed since the latent space distributions are not aligned.

A.4.7 Inference and adding new clients

Consider the inference problem at the edge. There is an existing federation of some users which are trained together using FedCMA. Now, a new user is deployed at the edge, which is previously not a part of the federation. In this case, the new user model may be lying in a different space than the models of all the other users. Therefore, the new user faces the inference challenge, since it needs the knowledge of the trained models in the federation. Below, we shall propose a method to add a new client to the federation, or transfer the knowledge of the federation to a new client for inference.

For every new client that is added, the server first communicates the global mean embeddings and the global final layer weights to the new client. The client then trains locally using the proposed algorithm until the local mean embeddings are ϵ away from the global mean embeddings (where ϵ is determined by the server based on the global

training statistics). After this point, the client is free to participate in the federation. The detailed algorithm is presented in Algorithm 6.

Below, we present results when new clients are added to the federation using the CIFAR dataset setup in the previous section. For example, there are $N-2$ clients in the beginning of training and 2 new clients join towards the end of the training. The 2 new clients train locally using the global mean embeddings and final layer weights using the proposed algorithm (they do not communicate with each other or with the server yet). From Table A.7, we see that, using the information provided by the server, the new clients were able to significantly improve their performance as compared to training locally. These results show that Fed-CMA is an effective way to add new clients to a federation, and to efficiently transfer knowledge for inference at the resource constrained edge.

Algorithm 6 Fed-CMA with new clients

Time steps T elapsed with existing clients $[M]$.
New Client $M+1$ joins the federation.
Given: Mean embeddings $\{e^k_{(T)}\}_{k=1}^K$ and final classification layer weights $\bar{\mathbf{v}}_{(T)}$
Server: Broadcast $\{e^k_{(T)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(T)}$ to client $M+1$. Send threshold parameters ϵ_1 and ϵ_2.
At new client $M+1$**:**
for $t = 1, \cdots, T$ **do**
    Pick random minibatch $\xi_{(t)}$ from D_{M+1} and compute $\nabla\widehat{\Phi}_{M+1}$.
    Update $\mathbf{w}_{(M+1,t)}$ according to (2.6)
    Compute $\{e^k_{(M+1,t)}\}_{k=1}^K$ using $e^k_{M+1} = \frac{1}{b^k_{M+1}} \sum_{x \in \mathcal{B}^k_{M+1}} g_{M+1}(\mathbf{u}_{(M+1,t)}, x)$ for $k \in [K]$.
    Continue until: $\sum_{k=1}^K \|e^k_{(M+1,t)} - e^k_{(T)}\|_2 \leq \epsilon_1$ and $\|\mathbf{v}_{(M+1,t)} - \bar{\mathbf{v}}_{(T)}\|_2 \leq \epsilon_2$.
end for
Join Federation and share with server: mean embeddings $\{e^k_{(M+1,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(M+1,t)}$.

Num Workers (N)	Local	Fed-CMA
20	32.1 (1.7)	37.5 (0.9)
50	18.0 (1.5)	26.9 (1.5)
100	25.5 (0.1)	34.8 (0.2)

Table A.7: Accuracy of the new clients when trained locally vs using Algorithm 6 for different values of N.

Appendix B: Chapter Four Proofs

B.1 Proof of Lemma 4

By the optimality of $\widehat{f}$, we can write

$$\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}\widehat{f},\widehat{f}) \leq \mathcal{L}_D(f^*;f^*) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f^*;f^*)$$

Adding and subtracting $\mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f})$ on LHS and $\mathcal{L}_D(\mathcal{T}f^*,f^*)$ on RHS, we get

$$\begin{aligned}
&\mathcal{L}_D(\widehat{f};\widehat{f}) - \mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f}) + \mathcal{L}_D(\mathcal{T}\widehat{f},\widehat{f}) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}\widehat{f},\widehat{f}) \\
&\leq \mathcal{L}_D(f^*;f^*) - \mathcal{L}_D(\mathcal{T}f^*,f^*) \\
&\qquad + \mathcal{L}_D(\mathcal{T}f^*,f^*) - \mathcal{L}_D(\widehat{\mathcal{T}}_{\mathcal{G}}f^*;f^*)
\end{aligned}$$

The result is obtained by rearranging the terms.

B.2 Capacity of Function Classes and Results from Empirical Process Theory

Definition 8 (Covering Number)**.** *Let $(\mathcal{M},d)$ be a pseudo-metric space, and let $\epsilon>0$. Let $M_1,\cdots,M_k$ be balls of radius $\epsilon>0$ in $\mathcal{M}$. We say that $\{M_i\}_{i=1}^k$ is a covering of $(\mathcal{M},d)$ if $\mathcal{M}\subseteq\cup_{i=1}^k M_i$. The covering number $N(\epsilon,\mathcal{M},d)$ is defined as the smallest k such that the set of ϵ balls $\{M\}_1^k$ is a covering of $(\mathcal{M},d)$. If no such finite k exists, then the covering number is ∞.*

Definition 9 (Empirical Covering Number)**.** *Let $\mathcal{H}$ be a class of functions with domain $\mathcal{R}$, and let points $R^{1:n} := (R_1, \cdots, R_n)$ be points in $\mathcal{R}$. The empirical covering number is defined with respect to the pseudo metric $l_{R^{1:N}}(f,g) = \frac{1}{N}\sum_{i=1}^{N} |f(R_i) - g(R_i)|; \quad g \in \mathcal{F}$ and denoted by $N_1(\epsilon, \mathcal{F}, R^{1:N})$.*

Definition 10 (VC dimension[182])**.** *Let $\mathcal{H}$ denote a class of functions from $\mathcal{X} \to \{0,1\}$. The growth function is defined as, for any non-negative m,*

$$s(\mathcal{H}, m) := \max_{x_1, \cdots, x_m \in \mathcal{X}} |\{(h(x_1), \cdots, h(x_m)) : h \in \mathcal{H}\}|.$$

If for any $\{x_1, \cdots, x_m\}$, $|\{(h(x_1), \cdots, h(x_m)) : h \in \mathcal{H}\}| = 2^m$, we say $\mathcal{H}$ shatters the set $\{x_1, \cdots, x_m\}$. The VC dimension of $\mathcal{H}$ is defined as the largest number of points m that it can shatter, i.e.,

$$\texttt{VC-dim}(\mathcal{H}) := \sup\{m \in \mathbb{N} : s(\mathcal{H}, m) = 2^m\}.$$

For a real-valued function class, the capacity is defined in terms of the pseudo-dimension.

Definition 11 (pseudo-dimension[182])**.** *Let $\mathcal{F} : \{f : \mathcal{X} \to \mathbb{R}\}$. The pseudo-dimension $\texttt{d}_{\mathcal{F}}$ is defined as the largest integer m for which there exists $(x_1, \cdots, x_m, y_1, \cdots, y_m) \in \mathcal{X}^m \times \mathbb{R}^m$ such that for any $(b_1, \cdots, b_m) \in \{0,1\}^m$ there exists $f \in \mathcal{F}$ such that $\forall i : f(x_i) > y_i \iff b_i = 1$. For cases where the function class $\mathcal{F}$ is generated by a neural network with a fixed architecture and activation function, we can also write $d_{\mathcal{F}} = \texttt{VC-dim}(sign(\mathcal{F}))$, where $sign(\mathcal{F}) = \{sign(f) : f \in \mathcal{F}\}$ and $sign(x) = 1$ if $x \geq 0$ and $sign(x) = 1$ if $x < 0$.*

The following lemma relates the pseudo dimension of a function class to the empirical covering number.

Lemma 18 (see [183]). *Consider a set Z and a class $\mathcal{F} \subset \{f : Z \to [0, \bar{C}]\}$ of functions on Z with pseudo-dimension $\mathtt{d}_{\mathcal{F}} < \infty$. For any points $z^{1:n} \in Z^n$ and $\epsilon > 0$,*

$$N_1(\epsilon, \mathcal{F}, z^{1:n}) \leq e(\mathtt{d}_{\mathcal{F}} + 1)\Big(\frac{2e\bar{C}}{\epsilon}\Big)^{\mathtt{d}_{\mathcal{F}}}.$$

Lemma 19 (Pollard's tail inequality: Theorem 9.1 [137]). *Let $\mathcal{H}$ be a class of functions that map $\mathcal{R}$ into $[-B/2, B/2]$, and let μ be a probability measure on $\mathcal{R}$. Let $R_1, \cdots, R_n$ are i.i.d with distribution μ. For every $\epsilon > 0$,*

$$\begin{aligned}&\mathbb{P}\bigg\{\sup_{h\in\mathcal{H}}\Big|\frac{1}{n}\sum_{i=1}^{n} h(R_i) - \mathbb{E}[h(R)]\Big| > \epsilon\bigg\}\\ &\leq 8\mathbb{E}\big[N_1(\epsilon/8, \mathcal{H}, R^{1:n})\big]\exp\Big(-\frac{n\epsilon^2}{128B^2}\Big)\end{aligned}$$

where $N_1(\cdot, \mathcal{H}, R^{1:n})$ is the empirical covering number of $\mathcal{H}$ given data points $R^{1:n}$.

Remark 4. *Note that, during the proofs, we often consider*

$$\mathbb{P}\bigg\{\sup_{h\in\mathcal{H}}\Big(\frac{1}{n}\sum_{i=1}^{n} h(R_i) - \mathbb{E}[h(R)]\Big) > \epsilon\bigg\}$$

instead of the absolute value. Observe that,

$$\sup_{h\in\mathcal{H}}\Big(\sum_{i=1}^{n}\frac{h(R_i) - \mathbb{E}[h(R)]}{n}\Big) \leq \sup_{h\in\mathcal{H}}\Big|\sum_{i=1}^{n}\frac{h(R_i) - \mathbb{E}[h(R)]}{n}\Big|$$

and this shows that

$$\begin{aligned}&\bigg\{\sup_{h\in\mathcal{H}}\Big(\frac{1}{n}\sum_{i=1}^{n} h(R_i) - \mathbb{E}[h(R)]\Big) > \epsilon\bigg\}\\ &\subseteq \bigg\{\sup_{h\in\mathcal{H}}\Big|\frac{1}{n}\sum_{i=1}^{n} h(R_i) - \mathbb{E}[h(R)]\Big| > \epsilon\bigg\}.\end{aligned}$$

Therefore, we simply use

$$\begin{aligned}&\mathbb{P}\bigg\{\sup_{h\in\mathcal{H}}\Big(\frac{1}{n}\sum_{i=1}^{n} h(R_i) - \mathbb{E}[h(R)]\Big) > \epsilon\bigg\}\\ &\leq 8\mathbb{E}\big[N_1(\epsilon/8, \mathcal{H}, R^{1:n})\big]\exp\Big(-\frac{n\epsilon^2}{128B^2}\Big).\end{aligned}$$

B.3 Difference between $\mathcal{L}_\mu(f;f)$ and $\|f-\mathcal{T}f\|_{2,\mu}^2$

In this section, we show that $\mathcal{L}_\mu(f;f)$ overestimates $\|f-\mathcal{T}f\|_{2,\mu}^2$ by a variance term. First, we have

$$\begin{aligned}
\mathcal{L}_\mu(f,f) &= \mathbb{E}[\mathcal{L}_D(f,f)] \\
&= \mathbb{E}\Big[f(s,a) - R(s,a) - \gamma V_f(s')\Big]^2 \\
&= \mathbb{E}\Big[\big(f(s,a) - R(s,a)\big)^2 + \gamma^2 V_f^2(s') \\
&\quad - 2\gamma f(s,a)V_f(s') + 2\gamma R(s,a)V_f(s')\Big] \\
&= \mathbb{E}_\mu\Big[\big(f(s,a) - R(s,a)\big)^2\Big] + \gamma^2\mathbb{E}[V_f^2(s')] \\
&\quad - 2\gamma\mathbb{E}[f(s,a)V_f(s')] + 2\gamma\mathbb{E}[R(s,a)V_f(s')].
\end{aligned}$$

Further, we note that $\|f-\mathcal{T}f\|_{2,\mu}^2$ is expanded as

$$\begin{aligned}
&\|f-\mathcal{T}f\|_{2,\mu}^2 \\
&= \int \Big(f(s,a) - R(s,a) - \gamma\mathbb{E}_{s'\sim P(s,a)}[V_f(s')]\Big)^2 d\mu \\
&= \int (f(s,a) - R(s,a))^2 d\mu \\
&+ \gamma^2 \int \big(\mathbb{E}_{s'\sim P(s,a)}[V_f(s')]\big)^2 d\mu \\
&\quad - 2\gamma \int f(s,a)\ \mathbb{E}_{s'\sim P(s,a)}[V_f(s')]\ d\mu \\
&\quad + 2\gamma \int R(s,a)\ \mathbb{E}_{s'\sim P(s,a)}[V_f(s')] d\mu \\
&= \int (f(s,a) - R(s,a))^2 d\mu \\
&+ \gamma^2 \int \big(\mathbb{E}_{s'\sim P(s,a)}[V_f(s')]\big)^2 d\mu \\
&\quad - 2\gamma\mathbb{E}[f(s,a)\ V_f(s')] + 2\gamma\mathbb{E}[R(s,a)\ V_f(s')].
\end{aligned}$$

The two equations above yield the desired results.

Bibliography

[1] "US online advertising revenue since 2000." https://www.statista.com/statistics/183816/us-online-advertising-revenue-since-2000/, 2022. [Online; accessed 19-September-2022].

[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[3] "qCT-Lung: Catching lung cancer early." https://qure.ai/qct-lung-catching-lung-cancer-early/. Accessed: 2022-09-07.

[4] "AI is essential for solving the climate crisis." https://www.bcg.com/publications/2022/how-ai-can-help-climate-change. Accessed: 2022-09-07.

[5] "The good, the bad and the ugly uses of machine learning in election campaigns." https://www.centreforpublicimpact.org/insights/good-bad-ugly-uses-machine-learning-election-campaigns. Accessed: 2022-09-07.

[6] "2017 Equifax data breach." https://en.wikipedia.org/wiki/2017_Equifax_data_breach, 2022. [Online; accessed 17-September-2022].

[7] "Cambridge Analytica data scandal." https://en.wikipedia.org/wiki/Cambridge_Analytica, 2022. [Online; accessed 17-September-2022].

[8] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*, pp. 160–164, IEEE, 1982.

[9] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*, pp. 1–19, Springer, 2008.

[10] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 493–502, 2010.

[11] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18, IEEE, 2017.

[12] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[13] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 1–11, 2013.

[14] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[15] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," *arXiv preprint arXiv:1706.07880*, 2017.

[16] J. Regatti, G. Tendolkar, Y. Zhou, A. Gupta, and Y. Liang, "Distributed sgd generalizes well under asynchrony," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 863–870, IEEE, 2019.

[17] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pp. 20–27, 2004.

[18] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

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

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

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

[22] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

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

[25] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, *et al.*, "A field guide to federated optimization," *arXiv preprint arXiv:2107.06917*, 2021.

[26] J. Hanna and P. Stone, "Grounded action transformation for robot learning in simulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[27] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810, IEEE, 2018.

[28] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[29] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *arXiv preprint arXiv:2006.04779*, 2020.

[30] Y. Jin, Z. Yang, and Z. Wang, "Is pessimism provably efficient for offline rl?," *arXiv preprint arXiv:2012.15085*, 2020.

[31] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*, pp. 104–114, PMLR, 2020.

[32] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[33] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized knowledge transfer for personalized federated learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[34] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[35] Y. J. Cho, A. Manoel, G. Joshi, R. Sim, and D. Dimitriadis, "Heterogeneous ensemble knowledge transfer for training large models in federated learning," *arXiv preprint arXiv:2204.12703*, 2022.

[36] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[37] D. Yao, W. Pan, Y. Wan, H. Jin, and L. Sun, "Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization," *arXiv preprint arXiv:2111.14655*, 2021.

[38] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: an efficient federated learning framework for heterogeneous mobile clients," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 420–437, 2021.

[39] O. Litany, H. Maron, D. Acuna, J. Kautz, G. Chechik, and S. Fidler, "Federated learning with heterogeneous architectures using graph hypernetworks," *arXiv preprint arXiv:2201.08459*, 2022.

[40] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," *arXiv preprint arXiv:2105.10056*, 2021.

[41] K. Pillutla, K. Malik, A.-R. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, "Federated learning with partial model personalization," in *International Conference on Machine Learning*, pp. 17716–17758, PMLR, 2022.

[42] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," *arXiv preprint arXiv:2001.01523*, 2020.

[43] H. Zhao, R. T. Des Combes, K. Zhang, and G. Gordon, "On learning invariant representations for domain adaptation," in *International Conference on Machine Learning*, pp. 7523–7532, PMLR, 2019.

[44] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *VLDB*, vol. 4, pp. 180–191, Toronto, Canada, 2004.

[45] F. D. Johansson, D. Sontag, and R. Ranganath, "Support and invertibility in domain-invariant representations," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 527–536, PMLR, 2019.

[46] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, 2006.

[47] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, pp. 2200–2207, 2013.

[48] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[49] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, PMLR, 2015.

[50] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[51] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[52] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," *Advances in neural information processing systems*, vol. 29, 2016.

[53] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176, 2017.

[54] R. Tachet des Combes, H. Zhao, Y.-X. Wang, and G. J. Gordon, "Domain adaptation with conditional distribution matching and generalized label shift," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19276–19289, 2020.

[55] C. Shui, Z. Li, J. Li, C. Gagné, C. X. Ling, and B. Wang, "Aggregating from multiple target-shifted sources," in *International Conference on Machine Learning*, pp. 9638–9648, PMLR, 2021.

[56] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," *arXiv preprint arXiv:1911.02054*, 2019.

[57] Y. J. Cho, J. Wang, T. Chiruvolu, and G. Joshi, "Personalized federated learning for heterogeneous clients with clustered knowledge transfer," *arXiv preprint arXiv:2109.08119*, 2021.

[58] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

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

[60] M. Balog, I. Tolstikhin, and B. Schölkopf, "Differentially private database release via kernel mean embeddings," in *International Conference on Machine Learning*, pp. 414–422, PMLR, 2018.

[61] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[62] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[63] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *International Conference on Machine Learning*, pp. 9489–9502, PMLR, 2021.

[64] L. Lamport, "The weak byzantine generals problem," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 668–676, 1983.

[65] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, pp. 203–226, 2019.

[66] N. A. Lynch, *Distributed algorithms*. Elsevier, 1996.

[67] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, pp. 4613–4623, 2018.

[68] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[69] P. Blanchard, R. Guerraoui, J. Stainer, *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

[70] N. Gupta and N. H. Vaidya, "Byzantine fault-tolerant parallelized stochastic gradient descent for linear regression," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 415–420, IEEE, 2019.

[71] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," *arXiv preprint arXiv:1803.09877*, 2018.

[72] G. Damaskinos, E. M. E. Mhamdi, R. Guerraoui, R. Patra, and M. Taziki, "Asynchronous byzantine machine learning (the case of SGD)," *arXiv preprint arXiv:1802.07928*, 2018.

[73] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous SGD," *arXiv preprint arXiv:1903.07020*, 2019.

[74] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1544–1551, 2019.

[75] S. P. Karimireddy, L. He, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," *arXiv preprint arXiv:2006.09365*, 2020.

[76] Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019.

[77] Z. Yang, A. Gang, and W. U. Bajwa, "Adversary-resilient inference and machine learning: From distributed to decentralized," *arXiv preprint arXiv:1908.08649*, 2019.

[78] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, and S. Rouault, "SGD: Decentralized byzantine resilience," *arXiv preprint arXiv:1905.03853*, 2019.

[79] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer," *arXiv preprint arXiv:1912.11279*, 2019.

[80] A. Portnoy and D. Hendler, "Towards realistic Byzantine-robust federated learning," *arXiv preprint arXiv:2004.04986*, 2020.

[81] C. Xie, S. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation," *arXiv preprint arXiv:1903.03936*, 2019.

[82] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, pp. 8635–8645, 2019.

[83] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," *arXiv preprint arXiv:1802.07927*, 2018.

[84] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint arXiv:1803.01498*, 2018.

[85] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signSGD with majority vote is communication efficient and fault tolerant," *arXiv preprint arXiv:1810.05291*, 2018.

[86] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," in *International Conference on Machine Learning*, pp. 5311–5319, PMLR, 2021.

[87] X. Cao and L. Lai, "Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers," *IEEE Transactions on Signal Processing*, vol. 67, no. 22, pp. 5850–5864, 2019.

[88] C. Xie, O. Koyejo, and I. Gupta, "Zeno: Byzantine-suspicious stochastic gradient descent," *arXiv preprint arXiv:1805.10032*, 2018.

[89] R. Jin, X. He, and H. Dai, "Distributed Byzantine tolerant stochastic gradient descent in the era of big data," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[90] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[91] J. Ji, X. Chen, Q. Wang, L. Yu, and P. Li, "Learning to learn gradient aggregation by gradient descent," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization*, vol. 7, pp. 2614–2620, 2019.

[92] J. Chee and P. Toulis, "Convergence diagnostics for stochastic gradient descent with constant learning rate," in *International Conference on Artificial Intelligence and Statistics*, pp. 1476–1485, 2018.

[93] V. S. Borkar, "Stochastic approximation with two time scales," *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

[94] V. S. Borkar, *Stochastic approximation: A dynamical systems viewpoint*, vol. 48. Springer, 2009.

[95] V. B. Tadic, "Almost sure convergence of two time-scale stochastic approximation algorithms," in *Proceedings of the 2004 American Control Conference*, vol. 4, pp. 3802–3807, IEEE, 2004.

[96] H. Kushner and G. G. Yin, *Stochastic approximation and recursive algorithms and applications*, vol. 35. Springer Science & Business Media, 2003.

[97] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM Journal on Control and Optimization*, vol. 38, no. 2, pp. 447–469, 2000.

[98] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[99] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *arXiv preprint arXiv:1509.01240*, 2015.

[100] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[101] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[102] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[103] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[104] "Opening up a physics simulator for robotics." `https://www.deepmind.com/blog/opening-up-a-physics-simulator-for-robotics`. Accessed: 2021-10-18.

[105] S. Desai, H. Karnan, J. P. Hanna, G. Warnell, and P. Stone, "Stochastic grounded action transformation for robot learning in simulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6106–6111, IEEE, 2020.

[106] A. Afzal, D. S. Katz, C. L. Goues, and C. S. Timperley, "A study on the challenges of using robotics simulators for testing," *arXiv preprint arXiv:2004.07368*, 2020.

[107] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[108] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*, pp. 45–73, Springer, 2012.

[109] G. J. Gordon, *Approximate solutions to Markov decision processes*. Carnegie Mellon University, 1999.

[110] C. Szepesvári and R. Munos, "Finite time bounds for sampling based fitted value iteration," in *Proceedings of the 22nd international conference on Machine learning*, pp. 880–887, 2005.

[111] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration.," *Journal of Machine Learning Research*, vol. 9, no. 5, 2008.

[112] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European conference on machine learning*, pp. 317–328, Springer, 2005.

[113] A. Antos, C. Szepesvári, and R. Munos, "Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, no. 1, pp. 89–129, 2008.

[114] A. Lazaric, M. Ghavamzadeh, and R. Munos, "Finite-sample analysis of least-squares policy iteration," *Journal of Machine Learning Research*, vol. 13, pp. 3041–3074, 2012.

[115] A. M. Farahmand, R. Munos, and C. Szepesvári, "Error propagation for approximate policy and value iteration," in *Advances in Neural Information Processing Systems*, 2010.

[116] J. Chen and N. Jiang, "Information-theoretic considerations in batch reinforcement learning," in *International Conference on Machine Learning*, pp. 1042–1051, PMLR, 2019.

[117] H. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," in *International Conference on Machine Learning*, pp. 3703–3712, PMLR, 2019.

[118] T. Nguyen-Tang, S. Gupta, S. Venkatesh, *et al.*, "Sample complexity of offline reinforcement learning with deep relu networks," *arXiv preprint arXiv:2103.06671*, 2021.

[119] M. J. Khojasteh, V. Dhiman, M. Franceschetti, and N. Atanasov, "Probabilistic safety constraints for learned high relative degree system dynamics," in *Learning for Dynamics and Control*, pp. 781–792, PMLR, 2020.

[120] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 4093–4099, IEEE, 2020.

[121] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3387–3395, 2019.

[122] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 777–783, IEEE, 2020.

[123] O. Shelke, V. Baniwal, and H. Khadilkar, "Anticipatory decisions in retail e-commerce warehouses using reinforcement learning," in *8th ACM IKDD CODS and 26th COMAD*, pp. 272–280, 2021.

[124] H. Li, S. Shao, and A. Gupta, "Fitted value iteration in continuous mdps with state dependent action sets," *IEEE Control Systems Letters*, vol. 6, pp. 1310–1315, 2021.

[125] Z. Zhu, S. Gupta, A. Gupta, and M. Canova, "A deep reinforcement learning framework for eco-driving in connected and automated hybrid electric vehicles," *arXiv preprint arXiv:2101.05372*, 2021.

[126] Z. Zhu, N. Pivaro, S. Gupta, A. Gupta, and M. Canova, "Safe model-based off-policy reinforcement learning for eco-driving in connected and automated hybrid electric vehicles," *IEEE Transactions on Intelligent Vehicles*, 2022.

[127] S. R. Deshpande, S. Gupta, D. Kibalama, N. Pivaro, M. Canova, G. Rizzoni, K. Aggoune, P. Olin, and J. Kirwan, "In-vehicle test results for advanced propulsion and vehicle system controls using connected and automated vehicle information," *SAE International Journal of Advances and Current Practices in Mobility*, vol. 3, no. 2021-01-0430, pp. 2915–2930, 2021.

[128] S. R. Deshpande, S. Gupta, A. Gupta, and M. Canova, "Real-time ecodriving control in electrified connected and autonomous vehicles using approximate dynamic programing," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 1, 2022.

[129] M. Guériau and I. Dusparic, "Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1558–1563, IEEE, 2018.

[130] Y. Deng, H. Chen, S. Shao, J. Tang, J. Pi, and A. Gupta, "Multi-objective vehicle rebalancing for ridehailing system using a reinforcement learning approach," *Journal of Management Science and Engineering*, 2021.

[131] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising.," *Journal of Machine Learning Research*, vol. 14, no. 11, 2013.

[132] J. R. Regatti, A. A. Deshmukh, F. Cheng, Y. H. Jung, A. Gupta, and U. Dogan, "Offline RL with resource constrained online deployment," *arXiv preprint arXiv:2110.03165*, 2021.

[133] O. Hernández-Lerma and J. B. Lasserre, *Discrete-time Markov control processes: basic optimality criteria*, vol. 30. Springer Science & Business Media, 2012.

[134] K. Hinderer, "Lipschitz continuity of value functions in markovian decision processes," *Mathematical Methods of Operations Research*, vol. 62, no. 1, pp. 3–22, 2005.

[135] A. W. Vaart and J. A. Wellner, "Weak convergence," in *Weak convergence and empirical processes*, pp. 16–28, Springer, 1996.

[136] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[137] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk, *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[138] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[139] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[140] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[141] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[142] A. A. Deshmukh, S. Sharma, J. W. Cutler, M. Moldwin, and C. Scott, "Simple regret minimization for contextual bandits," *arXiv preprint arXiv:1810.07371*, 2018.

[143] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *arXiv preprint arXiv:2106.06860*, 2021.

[144] K. J. Åström, "Optimal control of markov processes with incomplete state information i," *Journal of Mathematical Analysis and Applications*, vol. 10, pp. 174–205, 1965.

[145] V. Vapnik, R. Izmailov, *et al.*, "Learning using privileged information: similarity control and knowledge transfer.," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 2023–2049, 2015.

[146] Y. Wu, S. Zhai, N. Srivastava, J. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh, "Uncertainty weighted actor-critic for offline reinforcement learning," *arXiv preprint arXiv:2105.08140*, 2021.

[147] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *arXiv preprint arXiv:2106.01345*, 2021.

[148] D. Brandfonbrener, W. F. Whitney, R. Ranganath, and J. Bruna, "Offline rl without off-policy evaluation," *arXiv preprint arXiv:2106.08909*, 2021.

[149] J. Buckman, C. Gelada, and M. G. Bellemare, "The importance of pessimism in fixed-dataset policy optimization," *arXiv preprint arXiv:2009.06799*, 2020.

[150] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, pp. 2052–2062, PMLR, 2019.

[151] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with fisher divergence critic regularization," in *International Conference on Machine Learning*, pp. 5774–5783, PMLR, 2021.

[152] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.

[153] Y. Guo, S. Feng, N. Le Roux, H. Lee, M. Chen, *et al.*, "Batch reinforcement learning through continuation method," in *International Conference on Learning Representations*, 2020.

[154] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[155] G. Blanchard, A. A. Deshmukh, Ü. Dogan, G. Lee, and C. Scott, "Domain generalization by marginal transfer learning.," *J. Mach. Learn. Res.*, vol. 22, pp. 2–1, 2021.

[156] A. A. Deshmukh, Y. Lei, S. Sharma, U. Dogan, J. W. Cutler, and C. Scott, "A generalization error bound for multi-class domain generalization," *arXiv preprint arXiv:1905.10392*, 2019.

[157] A. Kumar, J. Fu, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *arXiv preprint arXiv:1906.00949*, 2019.

[158] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *arXiv preprint arXiv:2005.05951*, 2020.

[159] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[160] L. Wang and K.-J. Yoon, "Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[161] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey.," *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[162] L. T. Liu, U. Dogan, and K. Hofmann, "Decoding multitask dqn in the world of minecraft," in *The 13th European Workshop on Reinforcement Learning (EWRL) 2016*, 2016.

[163] G. Konidaris and A. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 489–496, 2006.

[164] T. J. Perkins, D. Precup, *et al.*, "Using options for knowledge transfer in reinforcement learning," tech. rep., Citeseer, 1999.

[165] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," in *European Conference on Machine Learning*, pp. 412–424, Springer, 2005.

[166] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[167] A. Li, H. Hu, P. Mirowski, and M. Farajtabar, "Cross-view policy learning for street navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8100–8109, 2019.

[168] P.-A. Kamienny, K. Arulkumaran, F. Behbahani, W. Boehmer, and S. Whiteson, "Privileged information dropout in reinforcement learning," *arXiv preprint arXiv:2005.09220*, 2020.

[169] C. Cang, A. Rajeswaran, P. Abbeel, and M. Laskin, "Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl," *arXiv preprint arXiv:2106.09119*, 2021.

[170] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[171] W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1331–1340, PMLR, 2019.

[172] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.

[173] A. X. Lee, C. M. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, *et al.*, "Beyond pick-and-place: Tackling robotic stacking of diverse shapes," in *5th Annual Conference on Robot Learning*, 2021.

[174] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, N. Díaz-Rodríguez, and D. Filliat, "Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer," *arXiv preprint arXiv:1906.04452*, 2019.

[175] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[176] R. Ortner, D. Ryabko, P. Auer, and R. Munos, "Regret bounds for restless markov bandits," in *International conference on algorithmic learning theory*, pp. 214–228, Springer, 2012.

[177] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn, "Offline reinforcement learning from images with latent space models," in *Learning for Dynamics and Control*, pp. 1154–1168, PMLR, 2021.

[178] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[179] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[180] C. Chen, J. Zhou, J. Ding, and Y. Zhou, "Assisted learning for organizations with limited data," *arXiv preprint arXiv:2109.09307*, 2021.

[181] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[182] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 2285–2301, 2019.

[183] D. Haussler, "Sphere packing numbers for subsets of the boolean n-cube with bounded vapnik-chervonenkis dimension," *Journal of Combinatorial Theory, Series A*, vol. 69, no. 2, pp. 217–232, 1995.