ABSTRACT

Title of Dissertation: ARTIFICIAL INTELLIGENCE ON THE EDGE: FROM LEARNING TO DEPLOYMENT

Marco Bornstein
Doctor of Philosophy, 2025

Dissertation Directed by: Associate Professor Furong Huang
Department of Computer Science

As Artificial Intelligence (AI) has grown in use, so too have the computational, memory, and data resources required to train state-of-the-art AI models. AI models, and the data they are trained on, are traditionally stored centrally, in-house. However, data is increasingly collected by decentralized devices, like smart sensors and mobile phones, which are at the *edge* of a network. Primarily due to privacy constraints, via impending laws and regulation, it is increasingly difficult to transfer user data from such edge devices to a central, in-house server.

This emerging impediment has spurred interest and research into *edge learning*, a distributed and collaborative paradigm, where AI models are trained closer to, or on, the devices where data originates. While promising, edge learning faces obstacles to overcome before attaining widespread adoption.

Part I of this dissertation addresses the first obstacle, the ability of participating devices, known as edge devices, to *learn* AI models. Edge devices lack access to large memory and computational power sources that are common in traditional, centralized settings. To solve these is-

sues, Part I proposes: (i) an asynchronous decentralized edge-learning algorithm that removes the training-time dependence on the slowest device, improving training speed, (ii) a locality-sensitive hashing framework that allows devices, with the help of a server, to train a large-scale AI model without storing the entire model, and (iii) a compressed-sensing algorithm that shrinks the size of both training data and AI models, with minimal performance degradation, while improving training and inference runtime.

Part II of this dissertation addresses the second obstacle, the inability to successfully *deploy* both fair collaborative edge-learning frameworks and safely trained, compliant AI models. Current edge-learning frameworks lack incentives. Thus, edge-learning participants may not always be incentivized to act in accordance with how the designee intended their frameworks to function, resulting in unfair and unintended outcomes. To solve these issues, Part II proposes: (i) two separate mechanisms to eliminate device free riding, one to incentivize device truthfulness when reporting information to a central server and a second to incentivize greater training contribution than by training alone, and (ii) a rigorous framework for AI regulation that incentivizes devices to both deploy compliant models and participate in the regulation process.

ARTIFICIAL INTELLIGENCE ON THE EDGE: FROM LEARNING TO DEPLOYMENT

by

Marco Bornstein

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Associate Professor Furong Huang, Chair/Advisor
Professor Tom Goldstein
Associate Professor Haizhao Yang
Assistant Professor Amrit Singh Bedi
Professor Nikhil Chopra

© Copyright by
Marco Bornstein
2025

Dedication

To Iris Brumer, whose humor, love, and resilience are guiding lights.

Acknowledgments

First and foremost, I want to thank my advisor, Dr. Furong Huang, for her mentorship and support at the University of Maryland. You took a chance on me, when I possessed no background in machine learning, during the darkness of the COVID-19 pandemic, and I am grateful to have learned so much from you. I have become a better thinker, writer, and academic because of you. I am in awe of your dedication and intelligence, and you inspire all of your students to reach for new heights.

I also want to thank Dr. Amrit Bedi for his mentorship. Ever since you began collaborating with our lab, my confidence and achievements have taken off. This is testament to your guidance and patience, which have been influential in the development of my mathematical maturity and creativity to formulate important research problems. I hope to continue our any-time-of-day brainstorming sessions.

Thank you to the amazing staff and directors within the Math Department. Liz Wincek, you always put a smile on my face and made my time on campus a joy. Jessica Sadler, Ruth Yun, Stephanie Padgett, and Jacqueline Dwyer, thank you for always answering my logistical questions, easing my transition into graduate school, and making my graduate career easier. Dr. Howard Elman, I am grateful to have had you as an academic advisor, teacher, and fellow Mets fan. Your advice has always been sound, and you have always pointed me towards the right path. You gave me the chance to become an AMSC graduate student, and I will never forget that.

I have been lucky to have been surrounded by and participate with wonderful co-authors and collaborators as a graduate student. Thank you to Dr. Tahseen Rabbani, Mucong Ding,

Souradip Chakraborty, Zora Che, Suhas Julapalli, Dr. Nawaf Nazir, Dr. Brian Gravelle, Dr. Abdirisak Mohamed, Dr. Ján Drgoňa, and Dr. Anit Sahu. A big thank you to my two older academic friends and mentors, Dr. Tahseen Rabbani and Dr. Avi Schwarzschild. Tahseen, I am thankful for our collaboration over the years and learning the ropes within machine learning from you. Avi, I have followed your exact footsteps ever since I touched campus. I am grateful for your academic advice and I hope to continue following in your footsteps.

Thank you to my graduate school cohort, roommates, and friends: Victoria, Michael, Luis, Kevin, Jameson, Shantam, Rolando, Dan, and Giuliano. I am so fortunate to have met you all, and am thankful to have braced the COVID-years with you.

I would like to acknowledge the support from the Laboratory of Physical Sciences (LPS) for many of my projects detailed herein, and much of my graduate career. Thank you, Dr. Brian Gravelle, for your mentorship and advice. I will miss our weekly discussions, and I hope to continue our collaboration!

I cannot be appreciative enough of my family and friends. Mom, Dad, Rebecca, and Miriam, thank you all for your love and guidance. I look up to each of you, and I would not have made it this far without you. Thanks for being the best proofreaders in the world! Thank you, Amit, Ariella, Daniel, Elie, Jake, Leah, Max, Nadav, Yoni, Isaac, and Jacob. It is rare to have maintained an amazing group of friends from such a young age. Your humor, love, and support have kept me sane.

Last, but most definitely not least, is Becca. Thank you so much for being my better half, and the one who has motivated, supported, and kept me going throughout graduate school. You and your family (thanks Tom, Liz, Jo, Zach, and Katy!) have provided a home away from home. I love you so much, and I am so grateful to have you in my life.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents v

List of Tables ix

List of Figures x

List of Abbreviations xvi

Chapter 1: Introduction 1
- 1.1 Edge-Learning Preliminaries . . . 5
  - 1.1.1 Centralized Federated Learning . . . 6
  - 1.1.2 Decentralized Federated Learning . . . 7
- 1.2 Summary of Contributions . . . 8
  - 1.2.1 Part I: Efficient Learning on Resource-Constrained Edge Devices . . . 9
  - 1.2.2 Part II: Mechanism Design for Edge Learning Fairness & AI Regulation . 10

I Efficient Learning on Resource-Constrained Edge Devices 13

Chapter 2: SWIFT: Rapid Decentralized Federated Learning via Wait-Free Model Communication 15
- 2.1 Preliminaries . . . 17
  - 2.1.1 Problem Formulation . . . 17
  - 2.1.2 Inter-Device Communication in Decentralized Federated Learning . . . 18
  - 2.1.3 Related Works . . . 21
- 2.2 Shared Wait-Free Transmission (SWIFT) Federated Learning . . . 23
- 2.3 SWIFT Theoretical Analysis . . . 29
- 2.4 Experiments . . . 32
  - 2.4.1 Baseline Comparison . . . 33
  - 2.4.2 Varying Heterogeneities . . . 34
  - 2.4.3 Varying Numbers of Devices & Network Topologies . . . 37

Chapter 3: Large-Scale Distributed Learning via Private On-Device Locality-Sensitive Hashing **39**
3.1 Preliminaries . . . 41
3.1.1 Problem Formulation . . . 41
3.1.2 Locality-Sensitive Hashing . . . 41
3.1.3 Related Works . . . 44
3.2 PGHash . . . 47
3.2.1 Theoretical Analysis . . . 48
3.2.2 PGHash Algorithm . . . 52
3.3 Experiments . . . 54
3.3.1 Sensitivity Analysis . . . 54
3.3.2 Large-Scale Recommender Network Training . . . 56

Chapter 4: Shrinking the Size of Deep Extreme Multi-Label Classification **62**
4.1 Preliminaries . . . 63
4.1.1 Problem Formulation . . . 63
4.1.2 Related Works . . . 64
4.2 Compressed Labeling & Recovery Estimation . . . 65
4.2.1 Label Compression via SVD Approximation . . . 65
4.2.2 Label Recovery . . . 66
4.3 Experiments . . . 68

II Mechanism Design for Edge Learning Fairness & AI Regulation **71**

Chapter 5: FACT or Fiction: Can Truthful Mechanisms Eliminate Federated Free Riding? **73**
5.1 Preliminaries . . . 75
5.1.1 The Challenge of Free Riding in Federated Learning . . . 75
5.1.2 Related Works . . . 78
5.2 Eliminating Free Riding via Penalization . . . 82
5.3 FACT: Eliminating Free-Riding With Untruthful Devices . . . 86
5.4 Experiments . . . 90
5.4.1 Free Rider Elimination With Untruthful Devices . . . 90
5.4.2 Real-World Case Study: Inter-Hospital Skin Cancer Diagnosis . . . 94

Chapter 6: Towards Realistic Mechanisms That Incentivize Federated Participation and Contribution **96**
6.1 Preliminaries . . . 99
6.1.1 Problem Formulation . . . 99
6.1.2 Related Works . . . 102
6.2 Modeling Realistic Utility . . . 106
6.3 REALFM: A Step Towards Realistic Federated Mechanisms . . . 110
6.4 Experiments . . . 114

Chapter 7: Auction-Based Regulation for Artificial Intelligence **120**
7.1 Preliminaries . . . 123
7.1.1 Problem Formulation . . . 123
7.1.2 Related Works . . . 127
7.2 Reserve Thresholding: Bare Minimum Regulation . . . 129
7.3 Compliance-Incentivized Regulatory-Centered Auction (CIRCA) . . . 130
7.4 Experiments . . . 135
7.4.1 Compliance-Cost Ablation Study . . . 138

III Conclusion, Future Work, Appendices, and Bibliography **141**

Chapter 8: Conclusion and Future Work **142**

Chapter A: Appendix to Chapter 2 **146**
A.1 Experimental Setup . . . 146
A.1.1 Network Topologies . . . 146
A.1.2 Data Partitioning . . . 148
A.2 SWIFT Properties . . . 149
A.2.1 Properties of Communication Matrices . . . 149
A.2.2 Instantaneous Communication and Overcoming Computational Delay . . . 150
A.2.3 Optimal Step-Size Under Uniform Device Influence . . . 151
A.3 Proof of the Main Theorem . . . 151
A.4 Supplemental Lemmas . . . 166
A.5 Lemma Proofs . . . 166
A.5.1 Proof of Lemma 1 . . . 166
A.5.2 Proof of Lemma 6 . . . 169
A.5.3 Proof of Lemma 7 . . . 170
A.5.4 Proof of Lemma 8 . . . 171
A.5.5 Proof of Lemma 9 . . . 172
A.5.6 Proof of Lemma 10 . . . 173

Appendix B: Appendix to Chapter 3 **174**
B.1 Experimental Setup . . . 174
B.2 Amazon-670K and Wiki-325K Experiment Analysis . . . 175
B.3 PGHash: Angle Versus Hamming Distance . . . 177
B.4 Proofs . . . 179
B.4.1 Proof of Theorem 3 . . . 179
B.4.2 Proof of Proposition 1 . . . 181
B.4.3 Additional Theory . . . 183

Appendix C: Appendix to Chapter 5 **186**
C.1 Experimental Setup . . . 186
C.2 Proofs . . . 190
C.2.1 Proof of Theorem 4 . . . 190

C.2.2 Proof of Theorem 5 . . . 190
C.2.3 Proof of Theorem 6 . . . 191
C.2.4 Proof of Lemma 2 . . . 191
C.2.5 Proof of Theorem 7 . . . 193
C.2.6 Proof of Lemma 3 . . . 194
C.2.7 Proof of the Main Theorem . . . 195
C.3 Impact & Limitations . . . 197

Appendix D: Appendix to Chapter 6 **199**
D.1 Experimental Setup . . . 199
D.2 Additional Experiments . . . 201
D.2.1 8 Device Experiments . . . 201
D.3 Additional Properties . . . 207
D.4 Proofs . . . 208
D.4.1 Proof of Theorem 9 . . . 208
D.4.2 Proof of Theorem 10 . . . 211
D.4.3 Proof of Theorem 11 . . . 213
D.4.4 Proof of the Main Theorem . . . 216

Appendix E: Appendix to Chapter 7 **220**
E.1 Additional Experiments . . . 220
E.2 Proofs . . . 223
E.2.1 Proof of Theorem 13 . . . 223
E.2.2 Proof of Theorem 14 . . . 224
E.2.3 Proof of Corollary 4 . . . 227
E.2.4 Proof of Corollary 5 . . . 231
E.3 Repeating CIRCA Auctions . . . 235
E.4 AI Regulation Challenges . . . 239

Bibliography **241**

List of Tables

2.1 Rate and Complexity Comparisons for Decentralized Federated Learning Algorithms. . . . 17
2.2 Comparison of Average Epoch and Communication Times. . . . 33
2.3 Average CIFAR-10 Epoch & Communication Times for 16 Device Ring (Slowdown). . . . 34
2.4 Average CIFAR-10 Epoch & Communication Times with Varying Devices (Ring). 38
2.5 Average CIFAR-10 Epoch & Communication Times for Varying Topologies. . . . 38

3.1 Extreme Multi-Label Dataset Overview. . . . 53
3.2 **Average LSH time for PGHash over a range of tables.** The average μ time (and standard deviation σ) it takes for PGHash to perform *vanilla sampling* (exact matches) between the hash codes of sample x and each weight w_i in the final dense layer. Times are sampled for PGHash on Delicious-200k for batch size $M = 128$, $k = 9$, and $c = 8$ for one device. . . . 57

4.1 DECLARE's Test Performance vs. (Un)compressed Baselines. . . . 67

7.1 Equalized Odds Performance as Minority Class Data Increases. . . . 140

A.1 Hyper-parameters for all SWIFT Experiments. . . . 146

B.1 Hyper-parameters for federated experiments (PGHash & Federated SLIDE). . . . 175
B.2 Hyper-parameters for compression experiments (PGHash). . . . 175
B.3 **PGHash and SLIDE performance on Amazon-670K using SimHash.** Accuracy across the first 5,000 iterations for a single device. Batch size $M = 1024$, $k = 8$, and $c = 8$. . . . 176

C.1 Hyper-parameters for CIFAR-10 FACT Experiments. . . . 187
C.2 Hyper-parameters for MNIST FACT Experiments. . . . 187

D.1 Hyper-parameters for CIFAR-10 Experiments. . . . 200
D.2 Hyper-parameters for MNIST Experiments. . . . 200

List of Figures

1.1 **The Rise of Big Data**. Total data creation worldwide from 2010-2028 [1]. 2

2.1 Depiction of Varying Local Updates I_1 and Averaging Steps I_2 [2]. 20
2.2 SWIFT Schematic with $\mathcal{C}_s = \mathcal{C}_1$ (*i.e.,* Devices Aggregate Every Two Local Updates). 23
2.3 Baseline Performance Comparison on CIFAR-10 for 16 Devices (Ring). 34
2.4 Average CIFAR-10 Test Loss for Varying non-IIDness on 10 Devices (ROC-3C). 35
2.5 SWIFT vs. D-SGD for CIFAR-10 in 16 Device (Ring) with Varying Slowdown. . 36
2.6 Average Communication and Epoch Times for Increasing Numbers of Devices. . 37
2.7 Average Test Loss for Varying Network Topologies on CIFAR-10. 38

3.1 **Hash-based Dropout.** Two procedures for conducting cosine similarity estimation between full weight matrix $W \in \mathbb{R}^{d \times n}$ (column w_i denotes the weight of neurons i) and an input $x \in \mathbb{R}^d$. **(a)** SimHash generates a hash table via left-multiplication by a randomized rectangular Gaussian matrix $S \sim \mathcal{N}(0, I_d)$ onto the *fully-sized* W. **(b)** PGHash generates a hash table via left-multiplication by a randomized square Gaussian matrix $S \sim \mathcal{N}(0, I_c)$ onto a base projection $BW \in \mathbb{R}^{c \times n}$ of W. In both procedures, weight w_i is selected for activation if its signed hash matches the signed hash of x. 46
3.2 **Correlation Between Angle and Hamming Distance.** The average Hamming distance (x-axis) between a PG/SimHash hashed fixed vector x and collection of vectors $\mathcal{V}$ versus their true angles (y-axis) is plotted. Vectors are unit, length 100, and hashed to dimension $k = 25$ binary vectors via $\tau = 10$ or 100 hash tables. PGHash's sketch dimension is $c = 25$. PGHash and SimHash show strong correlation between Hamming distance and angular similarity. 55
3.3 **Compressed PGHash.** Model accuracy of a large recommendation system is recorded on an extreme classification task (Delicious-200K) using PGHash for varying compression rates (CR). Compressed PGHash, even at 90% compression, is competitive with full training (*without even including effects of sparsity-induced neuronal drop out*). 58
3.4 **Federated PGHash on Delicious-200K.** Model accuracy of a large recommendation system on an extreme classification task (Delicious-200K) trained in a federated setting is plotted. PGHash achieves competitive accuracies compared with Federated SLIDE and FedAvg. In fact, PGHash converges quicker to near-optimal accuracy. 59

3.5 **Federated Amazon-670K PGHash-D.** Model accuracy of a large recommendation system is recorded on Amazon-670K trained in a federated setting. PGHash-D matches the convergence of Federated SLIDE without requiring LSH to be performed by the central server. 60
3.6 **PGHash(-D) Computational Efficiency.** In Figure **3.6a**, PGHash(-D) activates only a fraction of the total final layer neurons *even without compression*. Through this induced sparsity, PGHash(-D) greatly reduces the computational complexity of forward and backward training compared to full training. In Figures **3.6b** and **3.6c**, PGHash(-D) is compared with the Sampled Softmax heuristic [3] (randomly sampling 10% of the total neurons). PGHash(-D) outperforms the Sampled Softmax baseline, selecting a better set of activated neurons, via LSH, to more efficiently train the recommender system. 61

4.1 **Simultaneous Performance and Efficiency Improvement**. DECLARE outperforms its uncompressed counterpart by 5.28% and 18.36% on Delicious-200K (top) and Wiki10-31K (bottom) while reducing the final output dimension by 99.975% and 99.677% respectively. DECLARE outperforms the baselines in terms of test accuracy (left) and convergence (right). 69

5.1 FACT Schematic. 82
5.2 **Enforcement of Device Truthfulness.** Average net improvement in loss over local training is plotted for iid (dotted line) and two non-iid device distributions (D-0.6: dashed, D-0.3: solid). For both CIFAR10 (left) and MNIST (right), devices maximize their net improvement in loss when truthful (0% added) about their true cost. This matches the results in Theorem 8. 88
5.3 **Reduction in Device Loss.** The average device loss for baselines on CIFAR10 (top row) and MNIST (bottom row) under IID (left), and two non-IID data distributions (center: D-0.6, right: D-0.3). Traditional federated learning (FL) is an upper bound on device loss (if devices did not free ride). FACT improves device loss over local training by up to a factor of 3x for CIFAR10 and 4x for MNIST. . 91
5.4 **Elimination of Free Riding via Penalty.** The penalty term $P_{fr}(m_i)$ plus data collection costs $c_i m_i$ is plotted for CIFAR10 (left) and MNIST (right) for varying data contributions m_i. These combined costs are minimized at the local optimum $m^*_{i,l}$, as predicted by Theorem 6. 93
5.5 **FACT Eliminates Free Riding in Realistic Settings**. When training an image classifier for diagnosing skin cancer, devices participating in FACT achieve much lower loss (66% less) than if they did not participate (left). Devices maximize their improvement in loss over local training when they are truthful; reporting inflated or deflated costs diminishes improvement in loss (middle). Devices minimize penalties when using their locally optimal amount of data ($m^* = 801$) for training (right). 94

6.1 **Federated Mechanism Diagram. (A) Decision Phase for Device Participation**. Each device decides whether it wants to participate in the mechanism. If so, the quantity of data points used by each agent m_i is sent to the server (no data is shared). Note that rational devices would participate in REALFM; the utility gained by participating is *never* less than what agents attain locally. **(B) Federated Training Phase**. Devices upload their updates and receive feedback from the server in an iterative manner. **(C) Accuracy & Monetary Reward Distribution Phase.** Upon completion of federated training, the server distributes both accuracy $a^r(m_i)$ and monetary $R(m_i)$ rewards to device i. These rewards, the crux of REALFM, incentivize device participation and data contribution. 98

6.2 **Local Optimal Data Contribution for Varying Payoff Functions.** The optimal data contribution across different payoff functions are compared. Realistic power payoff functions, $\phi_i(a) = \frac{1}{(1-a)^2} - 1$, result in greater optimal contribution compared to linear payoff functions, $\phi_i(a) = a$. The accuracy function $\hat{a}_i(m)$ used is defined in Equation 6.4, with $a^i_{opt} = 0.95$ and multiple k values. 107

6.3 **Utility Functions for Varying Cost and Payoff Functions.** Using both linear, $\phi_i(a) = a$, and power, $\phi_i(a) = \frac{1}{(1-a)^2} - 1$, payoff functions, the change of device utilities with rising costs is analyzed. Once marginal costs c_i become too high, the utility is always negative and devices will not collect data for training. The accuracy function $\hat{a}_i(m)$ used is defined in Equation 6.4, with $a^i_{opt} = 0.95$ and $k = 1$. 108

6.4 **Improved Server Utility on CIFAR-10 & MNIST.** REALFM increases server utility on CIFAR-10 (top row) and MNIST (bottom row) for 16 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a FL version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 115

6.5 **Increased Federated Contribution on CIFAR-10 & MNIST.** REALFM incentivizes devices to use more local data during federated training on CIFAR-10 (left column) and MNIST (right column) for 16 devices compared to relevant baselines. REALFM achieves upwards of 4 magnitudes more federated contributions than LINEAR REALFM across both uniform and various heterogeneous Dirichlet data distributions (top: uniform, middle: D-0.6, bottom: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 116

6.6 **Improved Device Utility on CIFAR-10 & MNIST.** REALFM increases device utility on CIFAR-10 (top row) and MNIST (bottom row) for 16 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a federated learning version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 118

7.1 **CIRCA Schematic.** (Step 0) The regulator sets a compliance threshold, ϵ, having corresponding price, p_ϵ, required to achieve ϵ. (Step 1) Agents evaluate their total value, V_i, from model deployment value (v_i^d) and potential regulator compensation (v_i^p). Agents only participate if their total value exceeds p_ϵ. (Step 2) Participating agents submit their models to the regulator, accompanied by their bid b_i, which reflects the amount spent to improve their model's compliance level. Models with bids below p_ϵ are automatically rejected. (Step 3) The submitted models are randomly paired, and the more compliant model (*i.e.,* the higher bid) in each pair wins. In this example, agent 3 wins since $b_3 > b_1$. (Step 4) Winning models receive both a premium and deployment value (*i.e.,* agent 3 wins premium v_3^p and deployment v_3^d values), while losing models receive only the deployment value (*i.e.,* agent 1 only wins deployment value v_1^d). 122

7.2 **Validation of Uniform Nash Bidding Equilibrium.** Agent utility is maximized when agents follow the theoretically optimal bidding function shown in Equation (7.8). Across varying compliance prices, $p_\epsilon = 0.25$ (left), 0.5 (middle), 0.75 (right), agents attain less utility when they deviate from the optimal bid (red line) derived in Corollary 4. 132

7.3 **Validation of Beta Nash Bidding Equilibrium.** Akin to the Uniform results, agent utility is maximized when agents follow the theoretically optimal bidding function shown in Equation (7.11). Across varying compliance prices, $p_\epsilon = 0.25$ (left), 0.5 (middle), 0.75 (right), agents attain less utility when they deviate from the optimal bid (red line) derived in Corollary 5. 135

7.4 **Improved Compliance and Participation with Uniform Values**. When total value stems from a Uniform distribution $V_i \sim U(0, 1)$, agents participate at a higher rate in SIRA than Reserve Thresholding (left) and spend more to train compliant models (right). 136

7.5 **Improved Compliance and Participation with Beta Values**. When total value stems from a Beta distribution $V_i \sim$ Beta$(\alpha = \beta = 2)$, agents participate at a higher rate in SIRA than Reserve Thresholding (left) and spend more to train compliant models (right). 137

7.6 **Strictly Monotonic Compliance-Cost Relationship.** As the percentage of minority class data increases (greater cost), equalized odds metric improves (greater compliance) on Fairface. 139

A.1 **Diagram of Utilized Network Topologies**. The main network topologies used within SWIFT's experiments are a simple ring topology as well as a ring-of-cliques. 147

B.1 **PGHash-D results on Wiki-325K.** Model accuracy of a large recommendation network is recorded on Wiki-325K trained in a federated setting. PGHash-D matches or outperforms the convergence of SLIDE without requiring LSH to be performed by the central server. 175

B.2 **Angle vs. Hamming Distance as a function of sketch dimension.** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^{100}$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x. Increasing sketch dimension c smooths and reduces the variance of the scatter towards linear correlation. Furthermore, the Hamming scales linearly with c, improving discernibility. (a)-(c) & (d)-(f) are independent series. 178
B.3 **Angle vs. Hamming Distance as a function of tables** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^100$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x and fixed sketch dimension $c = 10$. Increasing the number of tables reduces variance. 179
B.4 **Distribution of Folded Norms.** One hundred thousand randomly drawn unit vectors ($d = 128$) are folded down to length 16 by are usual (d, c)-folding procedure. Depicted is a binned histogram of the norms. As predicted by the statistical description of $||BX||^2$, where X is a randomly drawn unit vector, the mass is centered at 1, *i.e.,* most norms are preserved. Empirically one can observe that folded rarely exceed $\sqrt{128}16$, although the theoretical support is $[0, 8]$: this concurs with the pdf. 183

C.1 **HAM10000 Test Loss (Left) and Accuracy (Right) for Federated and Local Training.** Consistent with the experiments for CIFAR-10 and MNIST, federated learning outperforms local training even though the HAM10000 dataset is highly skewed. 187
C.2 **Test Loss for CIFAR-10 (left) and MNIST (right) in Heterogeneous Settings.** Federated learning outperforms local training on IID (top) and mild (center) & strong (bottom) non-IID Dirichlet settings. 188
C.3 **Test Accuracy for CIFAR-10 (left) and MNIST (right) in Heterogeneous Settings.** Federated learning outperforms local training on IID (top) and mild (center) & strong (bottom) non-IID Dirichlet settings. 189

D.1 **Accuracy Comparison on CIFAR-10 & MNIST.** The difference between local training (red line) and federated training (blue line) on CIFAR-10 (left column) and MNIST (right column) for 16 devices with uniform marginal costs and payoff functions is displayed. As expected, federated training always outperforms local training. Both uniform and various heterogeneous Dirichlet data distributions are shown above (top: uniform, middle: D-0.6, bottom: D-0.3). 202
D.2 **Accuracy Comparison on CIFAR-10 & MNIST.** The difference between local training (red line) and federated training (blue line) on CIFAR-10 (left column) and MNIST (right column) for 8 devices with uniform marginal costs and payoff functions is displayed. As expected, federated training always outperforms local training. Both uniform and various heterogeneous Dirichlet data distributions are shown above (top: uniform, middle: D-0.6, bottom: D-0.3). 203

D.3 **Improved Server Utility on CIFAR-10 & MNIST.** REALFM increases server utility on CIFAR-10 (top row) and MNIST (bottom row) for 8 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a FL version of [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 204
D.4 **Increased Federated Contribution on CIFAR-10 & MNIST.** REALFM incentivizes devices to use more local data during federated training on CIFAR-10 (left column) and MNIST (right column) for 8 devices compared to relevant baselines. REALFM achieves upwards of 4 magnitudes more federated contributions than a federated learning version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (top: uniform, middle: D-0.6, bottom: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 205
D.5 **Improved Device Utility on CIFAR-10 & MNIST.** REALFM increases device utility on CIFAR-10 (top row) and MNIST (bottom row) for 8 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a federated learning version of [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P). 206

E.1 **Uniform PDF and CDF Validation**. Numerical validation of the derivations for $f_v(v_i^p)$ and $F_v(v_i^p)$, where $v_i^p := V_i\lambda_i$, for V_i and λ_i coming from Uniform distributions (Corollary 4). The price of attaining ϵ is set as $p_\epsilon = 1/4$ (top row) and $p_\epsilon = 1/2$ (bottom row). 221
E.2 **Beta PDF and CDF Validation**. Numerical validation of the derivations for $f_v(v_i^P)$ and $F_v(v_i^P)$, where $v_i^p := V_i\lambda_i$, for V_i coming from a Beta distribution and λ_i from a Uniform distributions (Corollary 5). The price of attaining ϵ is set as $p_\epsilon = 1/4$ (top row) and $p_\epsilon = 1/2$ (bottom row). 222

List of Abbreviations

AI	Artificial Intelligence
CFL	Centralized Federated Learning
DFL	Decentralized Federated Learning
DNN	Deep Neural Network
DWTA	Densified Winner Take All
FL	Federated Learning
FLOP	Floating Point Operation
GDPR	General Data Protection Regulation
GB	Gigabytes
IID	Independent and Identically Distributed
IoT	Internet of Things
LLM	Large Language Model
LSH	Locality-Sensitive Hashing
MIPS	Maximum Inner Product Search
NNS	Nearest-Neighbor Search
SVD	Singular Value Decomposition
XMC	Extreme Multi-Label Classification
WTA	Winner Take All

Chapter 1: Introduction

The rise of Artificial Intelligence (AI) models reflect humankind's use of a digital machine to discern vast and complex patterns, undetected by the human eye, as a means to solve complicated tasks. The fuel for this machine is data; increased quantity and quality of training data (Big Data) have unlocked unforeseen AI model capability, with some models exhibiting human-level ability [5, 6, 7]. Coupled with the development of new model architectures, AI in the age of Big Data has revolutionized the fields of healthcare, natural language processing, and computer vision among others [8, 9, 10].

This initial success of AI has led to widespread AI adoption across a variety of industries. Currently, industry-grade AI models are built using a concentrated approach: models, as well as the data they are trained on, are stored and managed in-house [11]. However, within the past few years, this concentrated approach has faced a conundrum. The spike in size of AI models, coupled with exponential data growth, have made it difficult and costly to store everything in-house. Of greatest concern is the proliferation of data that increasingly stems from decentralized sources, namely mobile phones, sensors, and other internet of things (IoT) devices [12]. Data is no longer being collected by a single entity at the center of a network. Instead, user data is increasingly collected and generated by many devices located at the *edge* of a network [13, 14].

Data collection at the edge causes functional impediments when training AI models. These

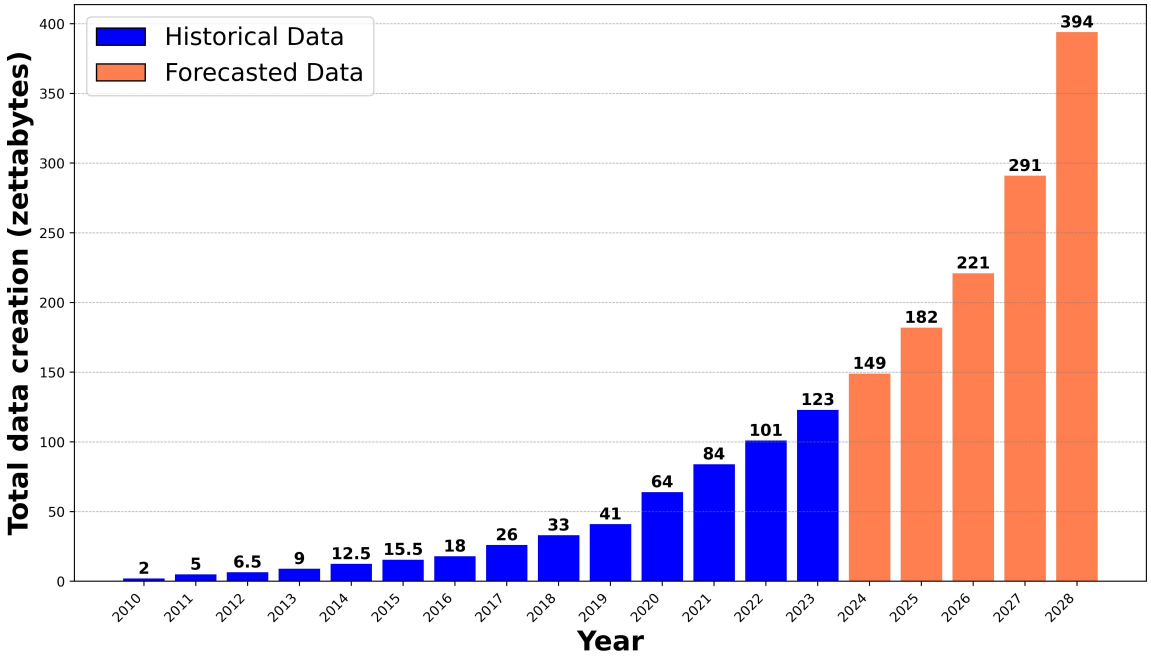


Figure 1.1: **The Rise of Big Data**. Total data creation worldwide from 2010-2028 [1].

include, without limitation, data security [15, 16, 17], network latency [18, 19], network bandwidth [20, 21], and general transmission costs [22, 23]. However, an emerging impediment poses the greatest threat: it may be difficult or impossible for data to be transferred from devices at the edge, known as *edge devices*, to the in-house center of the network for model training. This impediment occurs where data-privacy constraints exist. For example, many users may decline to send their private health data from a smartwatch to an online server. Issues surrounding the transfer of private data is only expected to grow [24, 25]. Laws and regulation to protect users' data privacy and security have begun to emerge from around the world. Most notably, in January 2024, the European Union enacted laws, the General Data Protection Regulation (GDPR) and Data Act, to improve data privacy and security protections [26, 27, 28, 29]. Impending laws and regulation to protect users' data privacy and security will prompt industries to move away from a concentrated approach and towards privacy-preserving methods compatible with edge devices.

With data increasingly emanating from the edge, and private-data transfer becoming more constricted, a new approach for training AI models has evolved to rival the traditional concentrated approach. This approach, known as *edge learning*, is a distributed and collaborative paradigm where models are trained closer to, or on, the devices where data originates [13, 14]. The benefits of this approach are: **(i)** lower bandwidth consumption and reduced latency (no need to transmit data back to a server), **(ii)** increased data privacy and security (personal data remains on-device), **(iii)** improved scalability and computational power (hundreds of thousands of devices can assist in training, boosting total resource power), **(iv)** resource efficiency (the server offloads computational costs to edge devices), and **(v)** offline operation (devices can function with limited or no internet access). A surge of research into edge learning over the past decade has produced algorithms and frameworks that have proven effective, efficient, and are competitive with the performance of traditional, concentrated approaches [30, 31, 32, 33].

Although its future possesses promise, edge learning faces obstacles to overcome before attaining widespread adoption. The first obstacle is the ability of edge devices to **learn** AI models. Edge devices lack access to large memory and computational power sources found in traditional, concentrated settings. In fact, many edge devices suffer severe memory and computational constraints [34]. As a result, edge devices inefficiently store and train AI models with billions of parameters, such as smaller-scale large-language models (LLMs). For instance, Meta's smallest open-source LLM, Llama 3, is eight billion parameters in size. This corresponds to approximately 32 gigabytes (GB) of memory if using 32-bit storage. Furthermore, simply running inference on these models, let alone training and back-propagation, is computationally expensive: billions of floating point operations (FLOPs) are required. Finally, as edge devices have widely varying computational power, training can be slow and often proceed at the pace of the slowest

device [31, 35].

The second obstacle that edge learning faces is the inability to successfully **deploy** both fair collaborative edge-learning frameworks and safely trained AI models. Current edge-learning frameworks lack incentives. Thus, edge-learning participants may not always be incentivized to act in accordance with how the designee intended their frameworks to function, resulting in unfair and unintended outcomes. In realistic settings, edge-learning participants often act in their own best interest. Consequently, when undertaking multi-device edge-learning frameworks, many participants are incentivized to take actions that either (a) reduce the total social welfare, or (b) are dishonest in order to attain greater benefit for themselves [4, 36, 37, 38, 39]. At a higher level, AI model deployment on the edge, as well as in general, is unguided, unsavory, and unregulated. Those who build AI models have limited incentive to produce and deploy safe, compliant AI models for public and private use.

My dissertation seeks to convert the promise of edge-learning frameworks into realistic performance. Its overarching goal is to improve the ability of edge learning to (i) **learn** powerful AI models in constrained settings and (ii) **deploy** frameworks that are fair in practice and improve AI safety. Particularly, I solve the following two questions:

In the age of Big Data and large-scale AI . . .

(1) How can memory- and computation-constrained edge devices more efficiently train large, high-performing AI models?

(2) How can edge-learning frameworks incorporate mechanisms to promote outcomes that benefit participants while simultaneously ensuring fairness and AI safety?

1.1 Edge-Learning Preliminaries

The preeminent edge-learning method is currently federated learning (FL) [32]. Within federated learning, an assortment of n edge devices collaboratively train an AI model $w \in \mathbb{R}^d$, of dimension d, with their own local data. Formally, federated learning solves the following optimization problem:

$$\min_{w \in \mathbb{R}^d} f(w) := \sum_{i=1}^{n} \alpha_i f_i(w). \tag{1.1}$$

The goal of federated learning is utilitarian, to find an optimal solution (model) w that performs well on all devices. The global objective function f shown in Equation 1.1 is the weighted sum of each device's local objective function f_i, where α_i is the weighting of device i's local objective function ($\sum_i \alpha_i = 1$). Each local objective function f_i often is non-convex. The local objective function consists of a loss function ℓ that uses data produced from device i's underlying distribution $\mathcal{D}_i$ to train the AI model,

$$f_i(w) := \mathbb{E}_{\xi \sim \mathcal{D}_i}\big[\ell(w; \xi)\big]. \tag{1.2}$$

Local objective functions differ only between devices when there exists a difference between their data distributions. When the data distribution $\mathcal{D}$ for all devices is identical, and thus $\mathcal{D}_i = \mathcal{D}\ \forall i$, then all local objective functions are identical as well. This scenario is defined in federated learning literature as the independent and identically distributed (IID) scenario.

Algorithm 1 Federated Averaging (FedAvg) [32].
Input: n devices; P_i is each device's local dataset; F is the device participation % each round; M is the batch size; E is the total epochs; w_0 is the initial model, $\boldsymbol{\alpha}$ are device weights (proportional to dataset size), and η is the learning rate.

Server Executes:
for each round $t = 1, 2, \ldots,$ **do**
  $m \leftarrow \max\{F * n, 1\}$
  $S_t \leftarrow$ (random set of m devices)
  for each device $i \in S_t$ in parallel **do**
    $w_i^{t+1} \leftarrow$ DeviceUpdate(i, w)
  $w^{t+1} \leftarrow \sum_{i=1}^{n} \alpha_i w_i^{t+1}$

DeviceUpdate(i, w)**:**
$\mathcal{B} \leftarrow$ (split P_i into batches of size M)
for each local epoch e from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

1.1.1 Centralized Federated Learning

The most commonly implemented and studied federated learning setting is the centralized federated learning (CFL) setting. In the centralized federated learning setting, a group of edge devices work together to train a model under the command of a centralized host. The foundational paper on federated learning, McMahan et al. [32], solves Equation 1.1 in an iterative centralized federated learning approach through (i) local gradient computations and (ii) gradient or model averaging amongst devices. This process is detailed algorithmically in Algorithm 1 and may be summarized by the following steps: **(1)** each device i computes a local gradient update $\nabla f_i(w^t)$ during round t, obtaining local model w_i^{t+1}, **(2)** once finished, each device sends its local model w_i^{t+1} (or gradient $\nabla f_i(w^t)$) to the central server, **(3)** after receiving all device models, the central server averages them together: $w^{t+1} = \sum_i w_i^{t+1}$, **(4)** the central server sends the resulting averaged model w^{t+1} back to all devices, and **(5)** steps (1)-(4) are repeated until convergence to a local optimum.

While centralized federated learning performs well in many applications [32, 40], it has drawbacks. The first drawback is that the central server is a single point of failure in the network

Algorithm 2 Decentralized Parallel SGD (D-PSGD) for the ith device [30].
Input: n devices i; C is the weight matrix; T is the total iterations; and η is the learning rate.

for each round $t = 0, 1, \ldots, T-1$ **do**
    Randomly sample local batch of data ξ_i^t
    Compute local SGD update with model w_i^t and data ξ_i^t: $\nabla f_i(w_i^t; \xi_i^t)$
    Compute neighborhood weighted average $w_i^{t+1/2} = \sum_{j=1}^{n} C_{i,j} w_j^t$
    Update local model $w_i^{t+1} \leftarrow w_i^{t+1/2} - \eta \nabla f_i(w_i^t; \xi_i^t)$
Output: $\frac{1}{n} \sum_{i=1}^{n} w_i^T$

(*i.e.* training stops if the central server fails). The second drawback is the inefficient scalability of centralized methods. Requiring a single server to send and receive messages can result in high communication latency that severely slows the run-time performance of an algorithm [30]. As the size of datasets and number of training devices multiply, central servers become inefficient due to increasing communication latency and scale poorly.

1.1.2 Decentralized Federated Learning

Within decentralized federated learning (DFL), the n edge devices are represented as the nodes of an arbitrary communication graph $\mathcal{G}$ with vertex set $\mathcal{V} = \{1, \ldots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ [30]. Each device i communicates with one-hop neighboring devices j such that $(i, j) \in \mathcal{E}$. Each device has a set of neighbors, defined as $\mathcal{N}_i := \{j |\ (i, j) \in \mathcal{E}\}$. It is important to note that every device is defined to be included within its own neighborhood $i \in \mathcal{N}_i$. Similar to centralized federated learning, each device i performs (i) local gradient updates and (ii) model averaging during each round t of decentralized federated learning,

$$w_i^{t+1/2} = w_i^t - \eta \nabla f_i(w_i^t), \quad \rightarrow \quad w_i^{t+1} = \sum_{j \in \mathcal{N}_i} c_{i,j} w_j^{t+1/2}. \tag{1.3}$$

Unlike centralized federated learning, neighboring devices communicate and average amongst themselves. Instead of a central server that averages all device updates, each device i averages the local model updates of *its neighbor's* $\mathcal{N}_i$. Therefore, it is possible in the decentralized setting for devices to have differing local models in the same round $w_i^t \neq w_j^t,\ i \neq j$. The model communication coefficient $c_{i,j}$ for each device j in device i's neighborhood ($\sum_{j \in \mathcal{N}_i} c_{i,j} = 1$) denotes the weighting that device i places on device j's model. All of the model communication coefficients can be represented as a non-negative communication matrix, denoted by $C \in \mathbb{R}^{n \times n}$. A positive (i, j)th entry signifies that devices i and j communicate with one another. Typically C is symmetric and doubly-stochastic to ensure that models are equally exchanged between devices. An example of a foundational decentralized federated learning algorithm, Decentralized Parallel Stochastic Gradient Descent (D-PSGD), is detailed in Algorithm 2.

Decentralized federated learning allays concerns found within its centralized counterpart by allowing devices to communicate with one another. Consequently, the high-communication-latency central server is discarded. By removing the central server, decentralized federated learning algorithms no longer have a single point of failure, can achieve superior run-time performance [30, 31, 41], and are shown to have improved data privacy [42, 43].

1.2 Summary of Contributions

The following dissertation is broken up into Parts I and II. Part I provides an answer for the first question I pose on Page 4. Part II answers the second. Below, I detail the organization of both Parts and then summarize their respective contributions.

1.2.1 Part I: Efficient Learning on Resource-Constrained Edge Devices

In Part I, I propose new algorithmic advancements within edge learning that improve the scalability as well as memory and computational efficiency over existing methods. For edge learning to become practiced, AI models must be smaller, training must be less expensive, and communication bottlenecks must be removed.

Chapter 2 proposes the first wait-free, asynchronous decentralized edge-learning algorithm. This algorithm, known as SWIFT, allows devices to conduct training at their own speed without having to pause for slower devices. Allowing wait-free, asynchronous training greatly improves training speed. Theoretically, as proven within the chapter, SWIFT matches the gold-standard iteration convergence rate $\mathcal{O}(1/\sqrt{T})$ of parallel stochastic gradient descent for convex and non-convex smooth optimization (total iterations T). Furthermore, theoretical results for IID and non-IID settings are provided *without* any bounded-delay assumption for slow devices. Such assumptions are required by previous asynchronous decentralized edge-learning algorithms. Finally, experimental results demonstrate that SWIFT reduces runtime speed while simultaneously matching or outperforming baseline algorithms.

Chapter 3 proposes a novel locality-sensitive hashing framework, PGHash, that may be deployed in edge-learning settings to dynamically prune model weights, thereby reducing both model size and total computations. Existing frameworks require the full model weights to be hashed, which is expensive to store and impractical for edge devices. Furthermore, conducting locality-sensitive hashing is unrealistic in a centralized edge-learning setting, since the process (i) is slow if the number of devices is large and (ii) requires access to input data that is forbidden in an edge-learning context. PGHash, on the other hand, allows for private, personalized, and memory-

efficient on-device locality-sensitive hashing. Hash tables are generated with a compressed set of the full weights, and can be serially generated and discarded if the process is memory-intensive. This allows devices to avoid maintaining (i) the fully sized model and (ii) large amounts of hash tables in local memory for locality-sensitive hashing analysis. Several statistical and sensitivity properties of PGHash are proven, and empirical results show that PGHash effectively trains large-scale recommender networks compared to baselines.

Finally, Chapter 4 presents a novel compressed-sensing method for training deep AI models with a large number of labels. Traditionally, the final output layer of these deep classifiers scales linearly with the size of the label set, which is often in the millions for realistic settings (*e.g.,* the number of products on Amazon). In order to deploy such recommender systems on-device, the training and storage of models must be more efficient and compressed. To address this, the DECLARE algorithm is proposed to first compress labels, via a pairwise label similarity preserving process, into a smaller dimension. Then, a deep neural network is trained to map features into the compressed label space. Consequently, this reduces training time and model-storage size. Since pairwise similarity is approximately preserved in the compressed space, DECLARE can infer the closest labels in the original space from a compressed label prediction. Empirically, DECLARE can compress labels by multiple magnitudes while maintaining uncompressed performance in certain instances.

1.2.2 Part II: Mechanism Design for Edge Learning Fairness & AI Regulation

The beginning chapters of Part II center on designing mechanisms, implemented by a central server, to incentivize devices to fully participate in edge-learning frameworks, thereby elim-

inating device free riding. The goal of this line of research is to develop superior edge-learning frameworks that successfully deal with realistic challenges arising post-implementation, like free riding. Part II ends by describing an initial push into the area of AI regulation.

Chapter 5 proposes a novel mechanism to eliminate federated free riding even when edge devices can be untruthful with a centralized server. Standard federated learning approaches are vulnerable to the free-rider dilemma: participating devices may contribute little to nothing, yet receive a well-trained aggregated model. While prior mechanisms attempt to solve the free-rider dilemma, none have addressed truthfulness. In practice, adversarial devices may provide false information to the server in order to avoid contributing to federated training. The proposed antidote, FACT, carefully designs a contract fee that penalizes devices from not contributing towards training. Furthermore, FACT implements a post-training reward payment system that awards participating devices, that have a median cost compared to its peers, the contract fees of other devices. It is theoretically proven, via a Nash Equilibrium, that the contract fees coupled with the new reward system dissuade free riding. Empirically, FACT avoids free-riding when devices are untruthful, and reduces device loss by over four times.

Chapter 6 likewise addresses the obviation of free riding via federated learning. Unlike Chapter 5, the mechanism proposed here, REALFM, provably incentivizes devices to contribute more towards federated training than they would by training solo. In effect, REALFM eliminates free riding because devices will never contribute less than what is optimal for them when training locally. Compared to previous federated mechanisms, REALFM allows for a non-linear relationship between model accuracy and utility, which improves the utility gained by the server and participating devices. On real-world data, REALFM improves device and server utility, as well as data contribution, *by over three and four magnitudes* respectively compared to baselines.

Finally, Chapter 7 proposes a rigorous framework for AI regulation. In an era of "moving fast and breaking things", regulators have moved slowly to pick up the safety, bias, and legal debris left in the wake of broken AI deployment. While there is much-warranted discussion about how to address the safety, bias, and legal woes of state-of-the-art AI models, rigorous and realistic mathematical frameworks to regulate AI are lacking.

This final chapter of my dissertation takes on this challenge to propose an auction-based regulatory mechanism that provably incentivizes devices (i) to deploy safe, compliant models and (ii) to participate in the regulation process. Theoretical guarantees, via derived Nash Equilibria, showcase that the best strategy for each device is to submit a model more compliant than a prescribed minimum-compliance threshold. Empirical results show that this regulatory auction boosts compliance and participation rates, by 20% and 15% respectively, to outperform simple regulatory frameworks that merely enforce minimum compliance standards.

Part I

Efficient Learning on Resource-Constrained Edge Devices

Part I Preface

Edge devices posses widely varying computational power and memory. Mobile devices provide an example of this; the new iPhone 16 has over 60% and 315% greater computing performance than the iPhone 12 and iPhone 8, respectively [44]. This comparison excludes further differences in memory and battery life. Even as edge devices are heterogeneous with respect to computational and memory constraints, the majority of edge-learning frameworks require synchronization during model training (*i.e.,* devices must wait to communicate with their neighbors until *everyone* finishes their work). This is problematic, because training time is proportionate to the speed of the slowest device in a network. Realistic edge-learning settings leverage tens of thousands, if not more, edge devices. The probability that one of these devices may be slow is high. Thus, in order to ensure edge-learning frameworks will not experience slowdown or break at scale, asynchronous frameworks must be implemented.

Along with runtime and scalability concerns, edge devices often lack the memory to fit current state-of-the-art machine learning models. As detailed in Chapter 1, even the smallest industry-grade open source LLMs can require around 32GB of memory storage. Consequently, it is impossible for many edge devices to store AI models locally for training, fine-tuning, or offline usage. As such, it is imperative to develop methods to enable efficient, personalized, and private learning on the edge. Part I of my dissertation is dedicated to tackling these difficulties issues of **learning on the edge**.

Chapter 2: SWIFT: Rapid Decentralized Federated Learning via Wait-Free Model Communication[1]

As detailed within Chapter 1.1, federated learning is an effective edge-learning method that allows edge devices to collaboratively train a powerful AI model using only their local data. Recent research [2, 30, 45] has focused on constructing decentralized federated learning algorithms to overcome speed and scalability issues found within classical centralized federated learning [32, 46]. While decentralized algorithms have eliminated a major bottleneck in the distributed setting, the central server, their scalability potential is still largely untapped. Many are plagued by high communication costs [41]. Shortening the amount of communication per round allows more devices to connect and then communicate with one another, thereby increasing scalability.

Due to the synchronous nature of current decentralized federated learning algorithms, communication time, and consequently run-time, is amplified by parallelization delays. The slowest device in the network causes parallelization delays. To circumvent these delays, asynchronous decentralized federated learning algorithms have been proposed [31, 47, 48, 49]. Unfortunately, these algorithms still suffer from high communication costs. Furthermore, their communication protocols either do not propagate models well throughout the network, due to inefficient gossip algorithms, or require partial synchronization. Finally, current asynchronous algorithms rely on a

[1] A version of Chapter 2 has been published in ICLR 2023 [35]

deterministic bounded-delay assumption, which assumes the slowest device in the network is *at most* τ iterations behind. This assumption is often unrealistic [50], and degrades the convergence rate by adding a sub-optimal reliance on τ.

To remedy these drawbacks, the **S**hared **W**a**I**t-**F**ree **T**ransmission (SWIFT) algorithm is proposed. SWIFT is an efficient, scalable, and high-performing decentralized federated learning algorithm. Unlike other decentralized federated learning algorithms, SWIFT obtains minimal communication time per round due to its wait-free structure. Furthermore, SWIFT is the first asynchronous decentralized federated learning algorithm to obtain an optimal $\mathcal{O}(1/\sqrt{T})$ convergence rate (aligning with stochastic gradient descent) *without a bounded-delay assumption*. Instead, SWIFT leverages the expected delay of each device. Experiments validate SWIFT's efficiency, showcasing a reduction in communication time by nearly an order of magnitude and run-times by upwards of 35%. All the while, SWIFT remains at state-of-the-art (SOTA) global test/train loss for image classification compared to other decentralized federated learning algorithms. In summary, the main contributions of SWIFT are as follows:

- ▷ Propose a novel wait-free decentralized federated learning algorithm (called SWIFT) and prove its theoretical convergence without a bounded-delay assumption.
- ▷ Implement a novel pre-processing algorithm ensuring non-symmetric and non-doubly stochastic communication matrices that are symmetric and doubly-stochastic under expectation.
- ▷ Provide the first theoretical device-communication error bound for non-symmetric and non-doubly stochastic communication matrices in the asynchronous setting.
- ▷ Demonstrate a significant reduction in communication time and run time for image classification in heterogeneous data settings compared to synchronous decentralized federated learning.

Table 2.1: Rate and Complexity Comparisons for Decentralized Federated Learning Algorithms.

Algorithm	Iteration Convergence Rate	Client (i) Comm-Time Complexity	Neighborhood Avg.	Asynchronous	Private Memory
D-PSGD	$\mathcal{O}(1/\sqrt{T})$	$\mathcal{O}(T \max_{j\in\mathcal{N}_i} \mathscr{C}_j)$	✓	✗	✓
PA-SGD	$\mathcal{O}(1/\sqrt{T})$	$\mathcal{O}(\lvert\mathcal{C}_s\rvert \max_{j\in\mathcal{N}_i} \mathscr{C}_j)$	✓	✗	✓
LD-SGD	$\mathcal{O}(1/\sqrt{T})$	$\mathcal{O}(\lvert\mathcal{C}_s\rvert \max_{j\in\mathcal{N}_i} \mathscr{C}_j)$	✓	✗	✓
AD-PSGD	$\mathcal{O}(\tau/\sqrt{T})$	$\mathcal{O}(T\mathscr{C}_i)$	✗	✓	✗
SWIFT	$\mathcal{O}(1/\sqrt{T})$	$\mathcal{O}(\lvert\mathcal{C}_s\rvert\mathscr{C}_i)$	✓	✓	✓

(1) Notation: total iterations T, communication set $\mathcal{C}_s$ ($|\mathcal{C}_s| < T$), client i's neighborhood $\mathcal{N}_i$, maximal bounded delay τ, and client i's communication time per round $\mathscr{C}_i$. **(2)** As compared to AD-PSGD, SWIFT does not have a τ convergence rate term due to using an expected client delay in analysis.

2.1 Preliminaries

2.1.1 Problem Formulation

Decentralized Federated Learning. The same decentralized federated learning notation presented in Chapter 1.1.2 is utilized within this chapter. The objective function to be optimized within federated learning is restated as,

$$\min_{w\in\mathbb{R}^d} f(w) := \sum_{i=1}^{n} \alpha_i f_i(w), \quad f_i(w) := \mathbb{E}_{\xi_i\sim\mathcal{D}_i}\big[\ell(w,\xi)\big], \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0. \tag{2.1}$$

The global objective function $f(w)$ is the weighted average of all local objective functions $f_i(w)$. In Equation 2.1, $\alpha_i, \forall i \in [n]$ denotes the device-influence score. This term controls the influence of device i on the global consensus model, forming the *device-influence vector* $\alpha = \{\alpha_i\}_{i=1}^n$. These scores also reflect the sampling probability of each device. Each local objective function $f_i(w)$ is the expectation of loss function ℓ with respect to potentially different local data $\xi_i = \{\xi_{i,j}\}_{j=1}^M$ from each device i's distribution $\mathcal{D}_i$, i.e., $\xi_{i,j} \sim \mathcal{D}_i$. The total number of iterations is denoted as T.

2.1.2 Inter-Device Communication in Decentralized Federated Learning

All devices balance their individual training with inter-device communications in order to achieve consensus while operating in a decentralized manner. The core idea of decentralized federated learning is that each device communicates with its neighbors (connected devices) and shares local information. Balancing individual training with inter-device communication ensures individual device models are well-tailored to personal data while remaining (i) robust to other device data, and (ii) able to converge to an optimal consensus model.

The predecessor to decentralized federated learning is gossip learning [51, 52]. Gossip learning was first introduced by the control community to assist with mean estimation of decentralized data distributions [53, 54]. Now, SGD-based gossip algorithms are used to solve large-scale machine learning tasks [31, 55, 56, 57, 58, 59]. A key feature of gossip learning is the presence of a globally shared oracle with whom devices exchange parameters at the end of training rounds [51]. While read/write-accessible shared memory is well-suited for a single-organization ecosystem (*i.e.,* all devices are controllable and trusted), this is unrealistic for more general edge-based paradigms. Below, newer decentralized SGD algorithms, such as D-PSGD [30], PA-SGD [45], and LD-SGD [2] are reviewed. These algorithms theoretically and empirically outperform their centralized counterparts, especially under heterogeneous data distributions.

Algorithms such as Periodic Averaging SGD (PA-SGD) [45] and Local Decentralized SGD (LD-SGD) [2], detailed further below, reduce communication time by performing *multiple* local updates before synchronizing. This process is accomplished through the use of a communication set $\mathcal{C}_s$, which defines the set of iterations where a device must perform synchronization,

$$\mathcal{C}_s = \{t \in \mathbb{N} \mid t \bmod (s+1) = 0,\ t \leq T\}. \tag{2.2}$$

As used later on in the Chapter, C_0 (C_1) means that communication occurs every (other) iteration.

Decentralized Parallel SGD (D-PSGD) [30]. One of the foundational decentralized federated learning algorithms is Decentralized Parallel SGD. In order to minimize Equation 2.1, D-PSGD orchestrates a local gradient step for all devices before performing synchronous neighborhood averaging. The D-PSGD process for a single device i is defined as,

$$w_i^{t+1} = \sum_{j=1}^{n} C_{ij}\left[w_j^t - g(w_j^t, \xi_j^t)\right]. \tag{2.3}$$

The term $g(w_j^t, \xi_j^t)$ denotes the stochastic gradient of w_j^t with mini-batch data ξ_j^t sampled from the local data distribution of device j. The matrix C is a communication matrix, where C_{ij} is the fraction of w_i^{t+1} which will be made up of device j's local model after one local gradient step (e.g. if $C_{ij} = 1/2$, then half of w_i^{t+1} will have been composed of device j's model after its local gradient step). The weighting matrix only has a zero value $C_{ij} = 0$ if devices i and j are not connected (they are not within the same neighborhood). The values of C_{ij} are generally selected ahead of time by a central host, with the usual weighting scheme being uniform. In D-PSGD, model communication occurs only after all local gradient updates are finished. These gradient updates are computed in parallel.

Periodic Averaging SGD (PA-SGD) [45]. The Periodic Averaging SGD algorithm is an extension of D-PSGD. In order to save communication costs when training with a large number of devices, PA-SGD performs model averaging after an additional I_1 local gradient steps. Thus,

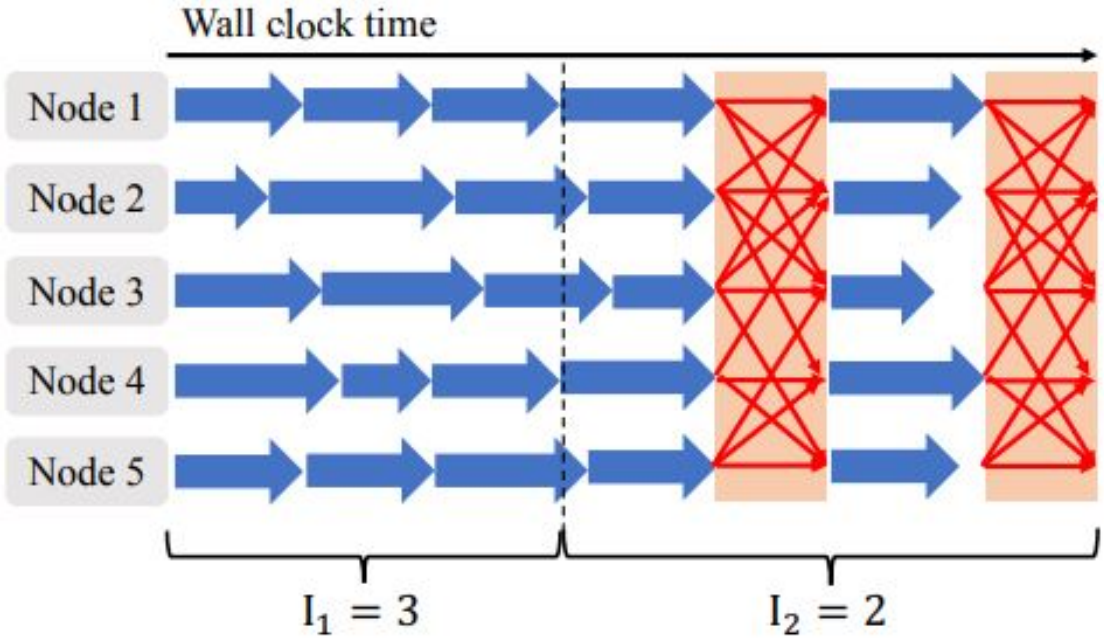


Figure 2.1: Depiction of Varying Local Updates I_1 and Averaging Steps I_2 [2].

the communication set for PA-SGD is defined as,

$$\mathcal{C}_{I_1} = \{t \in \mathbb{N} | \ t \mod (I_1 + 1) = 0\}. \tag{2.4}$$

In the special case that $I_1 = 0$, PA-SGD reduces to D-PSGD. The PA-SGD process for a single device i is defined as,

$$w_i^{t+1} = \begin{cases} \sum_{j=1}^{w} C_{ij}\left[w_j^t - g(w_j^t, \xi_j^t)\right], & t \in \mathcal{C}_{I_1} \\ w_i^t - g(w_i^t, \xi_i^t), & \text{otherwise.} \end{cases} \tag{2.5}$$

Like D-PSGD, PA-SGD suffers from having to wait for the slowest device for each update. However, PA-SGD saves communication costs by reducing the frequency of communication.

Local Decentralized SGD (LD-SGD) [2]. Continuing to generalize the foundational decentralized federated learning algorithms is Local Decentralized SGD. LD-SGD generalizes PA-

SGD by allowing multiple chunks of singular D-PSGD updates, as described in Equation 2.3, in between the increased local gradient steps seen in PA-SGD. The number of D-PSGD chunks is dictated by a new parameter I_2. The communication set for LD-SGD is defined as,

$$\mathcal{C}_{I_1,I_2} = \begin{cases} \bigcup \sum_{i=I_1}^{I_1+I_2} \{t \in \mathbb{N} |\ t \mod (i+1) = 0\} & \text{if } I_1 > 0, \\ \{t \in \mathbb{N}\} & \text{if } I_1 = 0. \end{cases} \tag{2.6}$$

For example, in the case $I_1 = 3, I_2 = 2$, LD-SGD will take three local gradient steps and then perform two D-PSGD updates (which consists of a local gradient step and then averaging) as shown in Figure 2.1. The special case of $I_2 = 1$ reduces to PA-SGD. The LD-SGD process for a single device i is defined as,

$$w_i^{t+1} = \begin{cases} \text{Perform } \sum_{j=1}^{w} C_{ij} \left[w_j^t - g(w_j^t, \xi_j^t) \right] & t \in \mathcal{C}_{I_1,I_2}, \\ w_i^t - g(w_i^t, \xi_i^t), & \text{otherwise.} \end{cases} \tag{2.7}$$

In the edge-learning literature, there also exist other asynchronous decentralized learning methods such as Cao and Başar [60], Bedi et al. [61, 62], but they are limited to convex objectives and hence not applicable to modern AI models that are non-convex.

2.1.3 Related Works

Asynchronous Learning. HOGWILD! [58], AsySG-Con [55], and AD-PSGD [30] are seminal examples of asynchronous algorithms that allow devices to proceed at their own pace. However, these methods require a shared memory/oracle from which devices grab the most up-

to-date global parameters (e.g. the current graph-averaged gradient). By contrast, SWIFT relies on a message passing interface (MPI) to exchange parameters between neighbors, rather than interface with a shared memory structure. To circumvent local memory overload, common in IoT clusters [63], devices in SWIFT access neighbor models sequentially when averaging. The recent works of [64, 65] have improved asynchronous SGD convergence guarantees which no longer rely upon the largest gradient delay. Like these works, SWIFT similarly proves convergence without bounded-delay assumptions. However, SWIFT differs as it functions in the decentralized domain as well as the federated learning setting. Decentralized SGD algorithms are reviewed above in Chapter 2.1.2.

Communication Under Expectation. Few works in federated learning center on communication uncertainty. In Ye et al. [66], a lightweight, yet unreliable, transmission protocol is constructed in lieu of slow heavyweight protocols. A synchronous algorithm is developed to converge under expectation of an unreliable communication matrix (probabilistic link reliability). SWIFT also convergences under expectation of a communication matrix, yet in a different and asynchronous setting. SWIFT is lightweight, reliable, and does not consider link reliability.

Communication Efficiency. Minimizing each device i's communication time per round $\mathscr{C}_i$ is a challenge in federated learning, as the radius of information exchange can be large [67]. MATCHA [41] decomposes the base network into m disjoint matchings. Every epoch, a random sub-graph is generated from a combination of matchings, each having an activation probability p_k. Devices then exchange parameters along this sub-graph. This requires a total communication-time complexity of $\mathcal{O}(T\sum_{k=1}^{m}\alpha_k \max_{j\in\mathcal{N}_i}\mathscr{C}_j)$, where $\mathcal{N}_i$ are device i's neighbors. LD-SGD [2] and PA-SGD [45] explore how reducing the number of neighborhood parameter exchanges affects convergence. Both algorithms create a communication set $\mathcal{C}_s$, defined in Chapter 2.1.2,

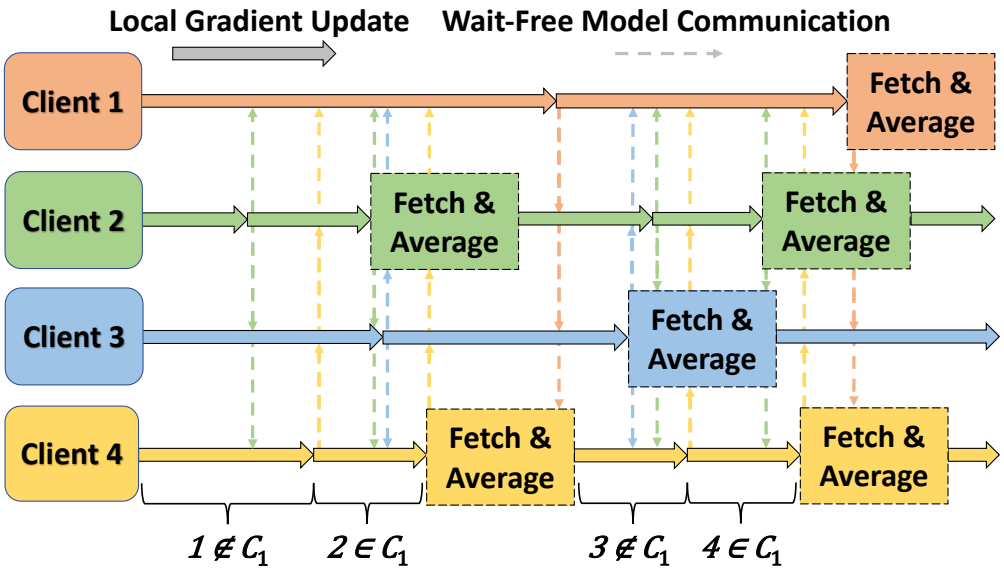


Figure 2.2: SWIFT Schematic with $\mathcal{C}_s = \mathcal{C}_1$ (*i.e.,* Devices Aggregate Every Two Local Updates).

that dictate when devices communicate with one another. The communication-time complexities are listed in Table 2.1. These methods, however, are synchronous and their communication-time complexities depend upon the slowest neighbor $\max_{j \in \mathcal{N}_i} \mathscr{C}_j$. SWIFT improves upon this, achieving a complexity depending on a device's own communication-time per round. Unlike AD-PSGD [31], which achieves a similar communication-time complexity, SWIFT allows for periodic communication, uses only local memory, and does not require a bounded-delay assumption.

2.2 Shared Wait-Free Transmission (SWIFT) Federated Learning

Below, the **S**hared **W**a**I**t-**F**ree **T**ransmission (SWIFT) algorithm is presented. SWIFT is an asynchronous algorithm that allows devices to work at their own speed, thereby removing dependency on the slowest device (a major drawback of synchronous methods). Moreover, unlike other asynchronous algorithms, SWIFT does not require a bound on the speed of the slowest device in the network and allows for neighborhood averaging and periodic communication.

A SWIFT Overview. Each device i runs SWIFT in parallel, receiving an initial model w_i, communication set $\mathcal{C}_s$, and counter $v_i \leftarrow 1$. SWIFT is summarized in the following steps:

1) Determine device-communication weights c_i via Algorithm 4.

2) Broadcast the local model to all neighboring devices.

3) Sample a random local data batch of size M.

4) Compute the gradient update of the loss function ℓ with the sampled local data.

5) Fetch and store neighboring local models, averaging them with one's local model if $v_i \in \mathcal{C}_s$.

6) Update the local model with the computed gradient update, as well as the counter $v_i \leftarrow v_i + 1$.

7) Repeat steps **(2)**-**(6)** until convergence.

A diagram and pseudocode of SWIFT are depicted in Figure 2.2 and Algorithm 3 respectively.

Active Devices, Asynchronous Iterations, and the Local-Model Matrix. Each time that a device finishes a pass through steps **(2)**-**(6)**, one global iteration is performed. Thus, the global iteration t is increased after the completion of *any* device's averaging and local gradient update. The device that performs the t-th iteration is called the active device, and is designated as i_t (Line 5 of Algorithm 3). *There is only one active device per global iteration*. All other device models remain unchanged during the t-th iteration (Line 15 of Algorithm 3). In synchronous algorithms, the global iteration t increases *only after all devices finish an update*. SWIFT, which is asynchronous, increases the global iteration t after *any* device finishes an update. In the theoretical analysis, the local-model matrix $W^t \in \mathbb{R}^{d\times n}$ is defined as the concatenation of all local device models at iteration t for ease of notation,

$$W^t := [w_1^t, \ldots, w_n^t] \in \mathbb{R}^{d\times n}. \tag{2.8}$$

Algorithm 3 **S**hared **Wa**i**t**-**F**ree **T**ransmission (SWIFT).

Input: Vertex set $\mathcal{V}$, Total steps T, Step-size γ, Device Influence Vector α, Distributions of device data $\mathcal{D}_i$, Communication set $\mathcal{C}_s$, Batch size M, Loss function ℓ, and Initial model w^0.

1: Initialize each device's local update counter $v_i \leftarrow 1,\ \forall i \in \mathcal{V}$
2: Obtain each device's new communication vector c_i^t using Algorithm 4
3: **for** $t = 1, \ldots, T$ **do**
4: **if** network topology changes **then** update communication vector c_i^t via Algorithm 4
5: Randomly **select an active device** i_t according to Device Influence Probability Vector α
6: **Broadcast active device's model** $w_{i_t}^t$ to all its neighbors $\{k \mid c_{i_t,k}^t \neq 0, k \neq i_t\}$
7: Sample a batch of active device i_t's local data $\xi_{i_t}^t := \{\xi_{i_t,m}^t\}_{m=1}^M$ from distribution $\mathcal{D}_{i_t}$
8: **Compute the gradient update**: $g(w_{i_t}^t, \xi_{i_t}^t) \leftarrow \frac{1}{M}\sum_{m=1}^M \nabla \ell(w_{i_t}^t, \xi_{i_t,m}^t)$
9: **if** current step falls in the predefined communication set, i.e., $v_{i_t} \in \mathcal{C}_s$ **then**
10: **Fetch and store the latest models** $\{w_k^t\}$ from i_t's neighbors $\{k \mid c_{i_t,k}^t \neq 0, k \neq i_t\}$
11: **Model average for the active device**: $w_{i_t}^{t+1/2} \leftarrow \sum_k c_{i_t,k}^t w_k^t + c_{i_t,i_t}^t w_{i_t}^t$
12: **else**
13: Active device model remains the same: $w_{i_t}^{t+1/2} \leftarrow w_{i_t}^t$
14: **Model update for the active device**: $w_{i_t}^{t+1} \leftarrow w_{i_t}^{t+1/2} - \gamma g(w_{i_t}^t; \xi_{i_t}^t)$
15: Update other devices: $w_j^{t+1} \leftarrow w_j^t, \quad \forall\ j \neq i_t$
16: Update the active device's counter $v_{i_t} \leftarrow v_{i_t} + 1$
17: **Output:** Consensus model $\frac{1}{n}\sum_{i=1}^n w_i^T$

Inspired by PA-SGD [45], SWIFT handles multiple local gradient steps before averaging models amongst neighboring devices (Line 9 of Algorithm 3). Periodic averaging for SWIFT, governed by a dynamic device-communication matrix, is detailed below.

Wait Free. The backbone of SWIFT is its wait-free structure. Unlike other decentralized federated learning algorithms, SWIFT does not require simultaneous averaging between two devices or a neighborhood of devices. Instead, each device fetches the latest models that its neighbors have sent, and performs averaging with those available (Lines 10-11 of Algorithm 3). There is no pause in local training waiting for a neighboring device to finish computations or average with, making SWIFT wait-free.

Update Rule. SWIFT runs in parallel, with devices performing local gradient updates and model communication simultaneously. Collectively, the update rule is written in matrix form as,

$$W^{t+1} = W^t C_{i_t}^t - \gamma G(w_{i_t}^t, \xi_{i_t}^t), \tag{2.9}$$

where γ denotes the step size parameter and the matrix $G(w_{i_t}^t, \xi_{i_t}^t) \in \mathbb{R}^{d \times n}$ is the zero-padded gradient of the active model $w_{i_t}^t$. Entries of $G(w_{i_t}^t, \xi_{i_t}^t)$ are zero except for the i_t-th column, which contains the active gradient $g(w_{i_t}^t, \xi_{i_t}^t)$. The device-communication matrix $C_{i_t}^t$ is detailed below.

Device-Communication Matrix. The backbone of decentralized federated learning algorithms is the device-communication matrix C (also known as the communication matrix). To remove all forms of synchronization and to become wait-free, SWIFT relies upon a novel device-communication matrix $C_{i_t}^t$ that is neither symmetric nor doubly-stochastic, unlike other algorithms in federated learning [2, 31, 45, 68]. The result of a non-symmetric and non-doubly stochastic device-communication matrix, is that averaging occurs for a single active device i_t and not over a pair or neighborhood of devices. This curbs superfluous communication time.

Within SWIFT, a dynamic device-communication matrix is implemented to allow for periodic averaging. The active device-communication matrix is denoted $C_{i_t}^t$ in SWIFT, where i_t is the active device which performs the t-th global iteration. $C_{i_t}^t$ can be one of two forms: (1) an identity matrix $C_{i_t}^t = I_n$ if $v_{i_t} \notin \mathcal{C}_s$ or (2) a communication matrix if $v_{i_t} \in \mathcal{C}_s$ with structure,

$$C_{i_t}^t := I_n + (c_{i_t}^t - \boldsymbol{e}_{i_t})\boldsymbol{e}_{i_t}^\intercal,\ c_{i_t}^t := [c_{1,i_t}^t, \ldots, c_{n,i_t}^t]^\intercal \in \mathbb{R}^n,\ \sum_{j=1}^{n} c_{j,i} = 1,\ c_{i,i} \geq 1/n\ \forall i. \tag{2.10}$$

The vector $c_{i_t}^t \in \mathbb{R}^n$ denotes *the active device-communication vector* at iteration t, which contains the communication coefficients between device i_t and all devices (including itself). The device-communication coefficients induce a weighted average of local neighboring models. Often, $c_{i_t}^t$ is

sparse since devices are connected to few other devices in most decentralized settings.

Novel Device-Communication Weight Selection. While utilizing a non-symmetric and non-doubly-stochastic device-communication matrix decreases communication time, there are technical difficulties when it comes to guaranteeing convergence. One contribution of this chapter is the careful design of a device-communication matrix $C_{i_t}^t$ such that it (i) is symmetric and doubly-stochastic *under expectation of all potential active devices* i_t and (ii) contains diagonal values greater than or equal to $1/n$.

$$\mathbb{E}_{i_t}\left[C_{i_t}^t\right] = \sum_{i=1}^{n} \alpha_i \left[I_n + (c_i^t - \boldsymbol{e}_i)\boldsymbol{e}_i^\intercal\right] = I_n + \sum_{i=1}^{n} \alpha_i (c_i^t - \boldsymbol{e}_i)\boldsymbol{e}_i^\intercal =: \bar{C}^t, \tag{2.11}$$

Let $\bar{C}^t$ denote the *expected device-communication matrix* with the following form,

$$[\bar{C}^t]_{i,i} = 1 + \alpha_i(c_{i,i}^t - 1), \text{ and } [\bar{C}^t]_{i,j} = \alpha_j c_{i,j}^t, \text{for } i \neq j. \tag{2.12}$$

Note that $\bar{C}^t$ is column stochastic, as all columns sum to one. If $\bar{C}^t$ can become symmetric, then it will become doubly-stochastic. By Equation 2.12, $\bar{C}^t$ becomes symmetric if,

$$\alpha_j c_{i,j}^t = \alpha_i c_{j,i}^t \; \forall i, j \in \mathcal{V}. \tag{2.13}$$

To achieve the symmetry of Equation 2.13, SWIFT deploys a novel pre-processing algorithm, the Communication Coefficient Selection (CCS) Algorithm. Given any device-influence vector α_i, CCS determines all device-communication coefficients such that Equations 2.10 and 2.13 hold for all global iterations t. Unlike other algorithms, CCS focuses on the *expected* device-communication matrix, ensuring its symmetry. For a fixed network topology, CCS only needs to

Algorithm 4 Communication Coefficient Selection (CCS).

Input: Device Influence Score (DIS) $\alpha_i \in \mathbb{R}$, Device Degree $d_i \in \mathbb{R}$, Device Neighbor Set $J_i = \{\forall j : \text{client } j \text{ is a one-hop neighbor of client } i\}, \forall i$.

for $i = 1 : n$ in parallel **do**
  if DIS are non-uniform **then**
    Initialize Device-Communication Vector $c_i = [c_{1,i}, c_{2,i}, \cdots, c_{n,i}] \leftarrow (1/n)\boldsymbol{e}_i$
  else
    Initialize Device-Communication Vector $c_i = [c_{1,i}, c_{2,i}, \cdots, c_{n,i}] \leftarrow \mathbf{0}$
  Exchange DIS and degree with all neighbors
  Store *Neighbor DIS Vector* $A^J \leftarrow [\{\alpha_j\}_{j \in J_i}] \in \mathbb{R}^{d_i}$
  Construct Neighbor Subsets J^{L}, J^{SE}, $J^{\mathsf{E}} \subset J_i$ as subsets of i's neighbors with degree larger than, no larger than and equal to d_i respectively
  for $\forall j \in J^{\mathsf{L}}$ **do** **Wait to fetch** $c_{j,i}$ from neighbor client j with a degree larger than d_i
    Determine the sum of the *total coefficients assigned (TCA)* $s_c^i \leftarrow \sum_{m=1}^n c_{m,i}$
    if $|J^{\mathsf{SE}}| > 0$ **then**
      Determine the sum of all remaining neighbors' DIS $s_\alpha^i \leftarrow \sum_{j \in J^{\mathsf{SE}}} A_j^J$
      if $|J^{\mathsf{E}}| > 0$ **then**
        Exchange s_c^i and s_α^i with all neighbors $j \in J^{\mathsf{E}}$, storing all exchanged s_c^j, s_α^j
        Store $s_c^* \leftarrow \max\{s_c^i, s_c^j\}$ and $s_\alpha^* \leftarrow \max\{s_\alpha^i, s_\alpha^j\} \ \ \forall j \in J^{\mathsf{E}}$
        Set $c_{j,i} \leftarrow \frac{(1-s_c^*)A_j^J}{s_\alpha^*} \ \forall j \in J^{\mathsf{E}}$
        Recompute $s_c^i = \sum_{m=1}^n c_{m,i}$ and $s_\alpha^i = \sum_{j \in \{J^{\mathsf{SE}} \cup i\} \setminus J^{\mathsf{E}}} A_j^J$
      Set $c_{j,i} \leftarrow c_{j,i} + \frac{(1-s_c^i)A_j^J}{s_\alpha^i} \ \forall j \in \{J^{\mathsf{SE}} \cup i\} \setminus J^{\mathsf{E}}$ for all remaining neighbors
      Send $c_{i,j} = \frac{(1-s_c^i)A_i^J}{s_\alpha^i}$ to all waiting neighbors $j \in J^{\mathsf{SE}} \setminus J^{\mathsf{E}}$
    else
      $c_{i,i} = 1 - s_c^i$
Output: Device-Communication Vector $c_i \in \mathbb{R}^n, \forall i \in [n]$

be run once before running SWIFT. In the event that the underlying topology changes, CCS can be run again during the middle of training.

The CCS Algorithm, shown in Algorithm 4, is a waterfall method: devices receive coefficients from their larger-degree neighbors. Devices run CCS concurrently in the following steps:

1) Receive coefficients from larger-degree neighbors. If the largest, or tied, skip to **(2)**.

2) Calculate the total coefficients already assigned s_c as well as the sum of the device influence scores for the unassigned devices s_α.

3) Assign the leftover coefficients $1 - s_c$ to the remaining unassigned neighbors (and self) pro-

portional to each unassigned device i's percentage of the leftover influence scores α_i/s_α.

4) If tied with neighbors in degree size, ensure coefficients do not sum to larger than one.

5) Send coefficients to smaller-degree neighbors.

Device-communication vector initializations differ in uniform and non-uniform settings. This ensures that the self-weight for each device i, $c_{i,i}$, is at least as large as $1/n$. This naturally occurs when the influence scores are uniform, but does not when they are non-uniform. In Algorithm 4, the terms s_c and s_α play a pivotal role in satisfying Equations 2.10 and 2.13 respectively. Equation 2.13 is satisfied by assigning weights amongst device i and its neighbors j as follows,

$$\alpha_j \frac{(1-s_c)\alpha_i}{s_\alpha} = \alpha_i \frac{(1-s_c)\alpha_j}{s_\alpha}. \tag{2.14}$$

Assigning weights proportionally with respect to neighboring device influence scores and total neighbors ensures higher influence devices receive higher weighting during averaging.

2.3 SWIFT Theoretical Analysis

Major Message from Theoretical Analysis. As summarized in Table 2.1, the efficiency and effectiveness of decentralized federated learning algorithms depend on both the iteration convergence rate and communication-time complexity; their product roughly approximates the total time for convergence. In this chapter, it is proven that SWIFT improves the SOTA convergence time of decentralized federated learning as it obtains SOTA iteration convergence rate (Theorem 1) and outperforms SOTA communication-time complexity. Before presenting the theoretical results, standard assumptions [67] required for the analysis are first detailed.

Assumption 1 (L-smooth objective functions)**.** $\|\nabla f(w) - \nabla f(y)\| \leq L \|w - y\|$.

Assumption 2 (Unbiased stochastic gradient)**.** $\mathbb{E}_{\xi \sim \mathcal{D}_i}\left[\nabla \ell(w;\xi)\right] = \nabla f_i(w)$ *for each* $i \in \mathcal{V}$.

Assumption 3 (Bounded inter-device gradient variance)**.** *The variance of the stochastic gradient is bounded for any* w *with device* i *sampled from probability vector* α *and local device data* ξ *sampled from* $\mathcal{D}_i$*. This implies there exist constants* $\sigma, \zeta \geq 0$ *(where* $\zeta = 0$ *in IID settings) such that:* $\mathbb{E}_i \|\nabla f(w) - \nabla f_i(w)\|^2 \leq \zeta^2 \; \forall i, \forall x$*,* $\mathbb{E}_{\xi \sim \mathcal{D}_i} \|\nabla f_i(w) - \nabla \ell(w;\xi)\|^2 \leq \sigma^2, \forall w$.

As mentioned in Chapter 2.2, the use of a non-symmetric, non-doubly-stochastic matrix $C_{i_t}^t$ causes issues in analysis. In Appendix A.2.1, properties of stochastic matrices are discussed, including symmetric and doubly-stochastic matrices, and ρ_ν, a network-connectivity-related constant is defined. Utilizing the symmetric and doubly-stochastic expected device-communication matrix (via Algorithm 4), Equation 2.9 can be reformulated by adding and subtracting $\bar{C}^t$,

$$W^{t+1} = W^t \bar{C}^t + W^t (C_{i_t}^t - \bar{C}^t) - \gamma G(w_{i_t}^t, \xi_{i_t}^t). \tag{2.15}$$

The first theoretical result, Lemma 1, establishes a device-communication error bound.

Lemma 1 (Device-Communication Error Bound)**.** *Following Algorithm 4, the product of the difference between the expected and actual device communication matrices is bounded by,*

$$\mathbb{E} \sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (C_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2 = \mathcal{O}\Big(\frac{\sigma^2}{M} + \mathbb{E} \sum_{j=0}^{t} \|\nabla f_{i_j}(w_{i_j}^j)\|^2\Big). \tag{2.16}$$

Remark. One novelty of SWIFT is that it is the first to bound the device-communication error in the asynchronous decentralized federated learning setting. The upper bound in Lemma 1 is

unique to the analysis, as other decentralized works are not wait-free [2, 30, 31, 45]. Now, the main theorem can be presented, establishing the convergence rate of SWIFT:

Theorem 1 (Convergence Rate of SWIFT)**.** *Under assumptions 1, 2 and 3 (with Algorithm 4), let* $\Delta_f := f(\bar{w}^0) - f(\bar{w}^*)$*, step-size* γ*, total iteration* T*, and average model* $\bar{w}^t$ *be defined as,*

$$\gamma := \frac{\sqrt{Mn^2\Delta_f}}{\sqrt{TL} + \sqrt{M}} \leq \sqrt{\frac{Mn^2\Delta_f}{TL}}, \quad T \geq 193^2 LM\Delta_f \rho_\nu^2 n^4 \alpha_{max}^2, \quad \bar{w}^t := \frac{1}{n}\sum_{i=1}^{n} w_i^t.$$

Then, for the output of Algorithm 3, it holds that,

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2\sqrt{\Delta_f}}{T} + \frac{2\sqrt{L\Delta_f}\left(1 + \frac{1921}{20}\rho_\nu\left(\sigma^2 + 6\zeta^2\sqrt{M}\right)\right)}{\sqrt{TM}}. \tag{2.17}$$

Iteration Convergence Rate Remarks. (1) SWIFT obtains a $\mathcal{O}(1/\sqrt{T})$ iteration convergence rate, matching the optimal rate for SGD [30, 31, 69, 70]. **(2)** Unlike existing asynchronous decentralized SGD algorithms, SWIFT's iteration convergence rate does not depend on the maximal bounded delay. Instead, any delays are bounded by taking the *expectation* over the active device. The probability of each device i being the active device is simply its sampling probability α_i. Therefore, each device i is *expected* to perform updates at its prescribed sampling probability rate α_i. Devices which are often delayed in practice can be dealt with by lowering their inputted sampling probability. **(3)** SWIFT converges in fewer total iterations T with respect to n total devices compared to other asynchronous methods [31] ($T = \Omega(n^4\alpha_{max}^2)$ in SWIFT versus $T = \Omega(n^4)$ in AD-PSGD). Similar to AD-PSGD, SWIFT achieves a linear speed-up in computational complexity as the number of devices increase.

Communication-Time Complexity Remarks. (1) Due to its asynchronous nature, SWIFT

achieves a communication-time complexity that relies only on each device's own communication time per round $\mathscr{C}_i$. This improves upon synchronous decentralized SGD algorithms, which rely upon the communication time per round of the slowest neighboring device $\max_{j \in \mathcal{N}_i} \mathscr{C}_j$. **(2)** Unlike AD-PSGD [31], which also achieves a communication-time complexity reliant on $\mathscr{C}_i$, SWIFT incorporates periodic averaging which further reduces the communication complexity from T rounds of communication to $|\mathcal{C}_s|$. Furthermore, SWIFT allows for entire neighborhood averaging, and not just one-to-one gossip averaging. This increases neighborhood information sharing, improving model robustness and reducing model divergence.

Corollary 1 (Convergence under Uniform Device Influence)**.** *In the common scenario where device influences are uniform,* $\alpha_i = 1/n \ \forall i \implies \alpha_{max} = 1/n$*, Swift obtains convergence improvements: total iterations* T *with respect to the number of total devices* n *improves to* $T = \Omega(n^2)$ *as compared to* $T = \Omega(n^4)$ *for AD-PSGD under the same conditions.*

2.4 Experiments

Image classification experiments are performed, using CIFAR-10 [71], for a range of decentralized federated learning (DFL) algorithms. SWIFT is compared to the following baselines:

▷ The most common synchronous DFL algorithm: D-PSGD [30] (also notated as D-SGD).

▷ Synchronous DFL communication reduction algorithms: PA-SGD [45] and LD-SGD [2].

▷ The most prominent asynchronous DFL algorithm: AD-PSGD [31].

Finer details of the experimental setup are in Appendix A.1. Throughout the experiments, two network topologies are tested: standard ring and ring of cliques (ROC). ROC-xC signifies a ring of cliques with x clusters. The ROC topology is a realistic network, as networks usually have

Table 2.2: Comparison of Average Epoch and Communication Times.

Decentralized FL Algorithms	16 Device Ring			
	Epoch (s)	% Change	Comm. (s)	% Change
SWIFT ($\mathcal{C}_0$)	**1.019**	-34.60	**0.086**	-86.28
D-SGD ($\mathcal{C}_0$)	1.558	—	0.627	—-
AD-PSGD* ($\mathcal{C}_0$)	—	-15.86	—	—
SWIFT ($\mathcal{C}_1$)	**1.016**	-34.79	**0.064**	-89.79
LD-SGD ($\mathcal{C}_1$)	1.320	-15.28	0.428	-31.74
PA-SGD ($\mathcal{C}_1$)	1.281	-17.78	0.358	-42.90

* AD-PSGD results come from Table 4 in [31].

pockets of connected devices. These topologies are visualized in Appendix A.1 (Figure A.1).

2.4.1 Baseline Comparison

To compare the performance of SWIFT to all other algorithms listed above, including AD-PSGD, the experiments within Lian et al. [31] are reproduced. With no working code for AD-PSGD to run on except for a super-computing cluster (Section 5.1.2 of [31]), reproducing this experiment allows a relative comparison between SWIFT to AD-PSGD.

Table 2.2 showcases that SWIFT reduces the average epoch time, relative to D-SGD, by 35% ($\mathcal{C}_0$ and $\mathcal{C}_1$). This far outpaces AD-PSGD (as well as the other synchronous algorithms), with AD-PSGD only reducing the average epoch time by 16% relative to D-SGD. Finally, Figure 2.3 displays how much faster SWIFT achieves optimal train and test loss values compared to other decentralized baseline algorithms. SWIFT outperforms all other baseline algorithms even without any slow-down (which is examined in Chapter 2.4.2), where wait-free algorithms like SWIFT especially shine.

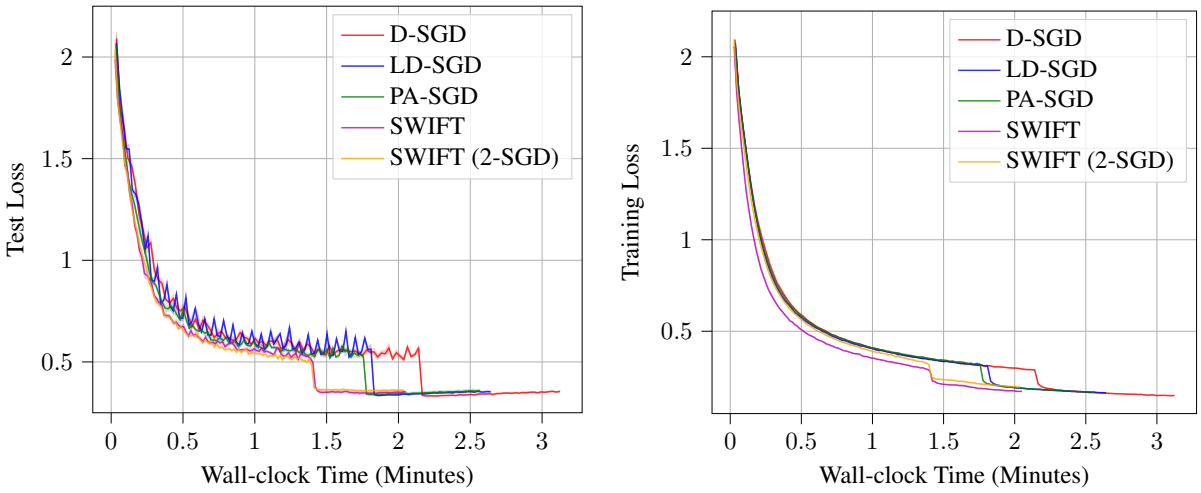


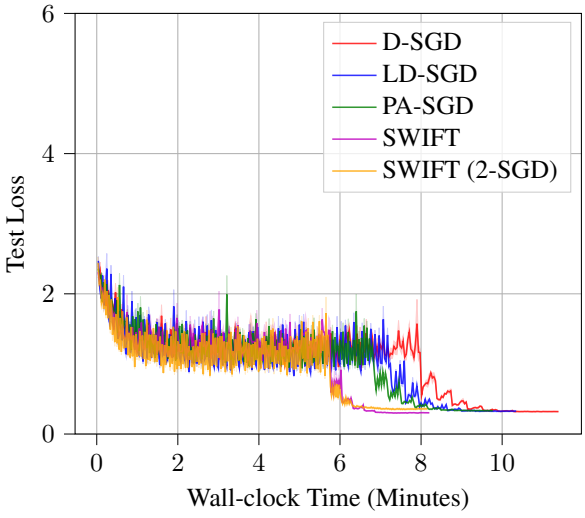
Figure 2.3: Baseline Performance Comparison on CIFAR-10 for 16 Devices (Ring).

Table 2.3: Average CIFAR-10 Epoch & Communication Times for 16 Device Ring (Slowdown).

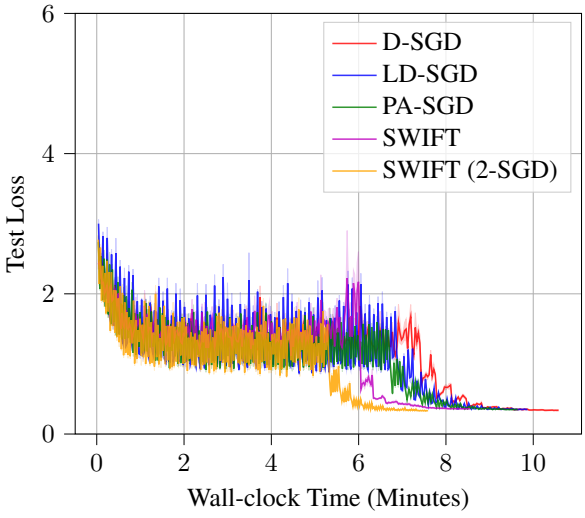
Decentralized FL Algorithms	No Slowdown			2x Slowdown			4x Slowdown		
	Total (s)	Epoch (s)	Comm. (s)	Total (s)	Epoch (s)	Comm. (s)	Total (s)	Epoch (s)	Comm. (s)
SWIFT ($\mathcal{C}_0$)	**2.69**	**1.033**	**0.087**	**2.451**	**0.964**	**0.064**	**3.054**	**1.117**	**0.091**
D-SGD ($\mathcal{C}_0$)	3.110	1.45	0.564	3.972	1.571	0.616	6.137	1.666	0.651
SWIFT ($\mathcal{C}_1$)	**2.44**	**0.996**	**0.061**	**2.152**	**0.928**	**0.040**	**2.847**	**1.074**	**0.065**
LD-SGD ($\mathcal{C}_1$)	3.323	1.353	0.413	3.762	1.389	0.428	5.917	1.412	0.448
PA-SGD ($\mathcal{C}_1$)	3.183	1.270	0.331	3.557	1.262	0.309	5.743	1.270	0.318

2.4.2 Varying Heterogeneities

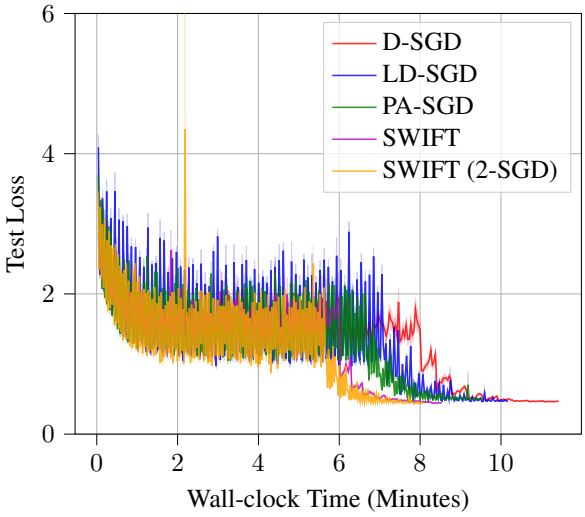
Varying Degrees of Non-IIDness. The second experiment evaluates SWIFT's efficacy at converging to a well-performing optimum under varying degrees of non-IIDness in a 3-cluster ROC network topology. The degree (percentage) of each device's data coming from one label is varied. The remaining percentage of data is randomly sampled (IID) data over all labels. Ten devices are used for ease of label distribution: CIFAR-10 has 10 labels. As expected, when data becomes more non-IID, the test loss increases. However, SWIFT converges *much* faster, and to a *lower* average loss, than all other synchronous baselines, even in non-IID settings (Figure 2.4). Thus, SWIFT outperforms synchronous algorithms across all settings.



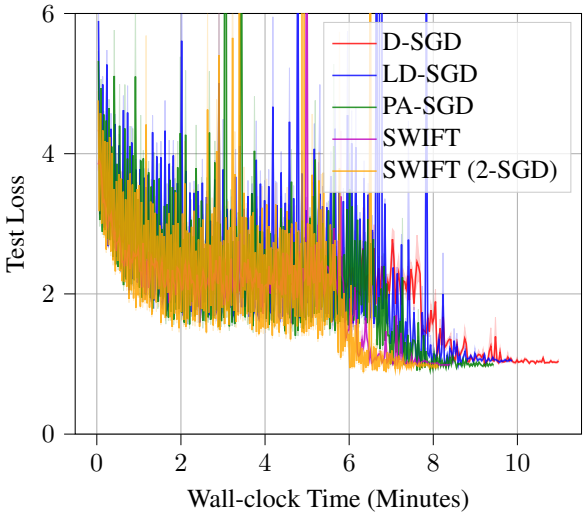
(a) 1/4 degree non-IID data.



(b) 1/2 degree non-IID data.

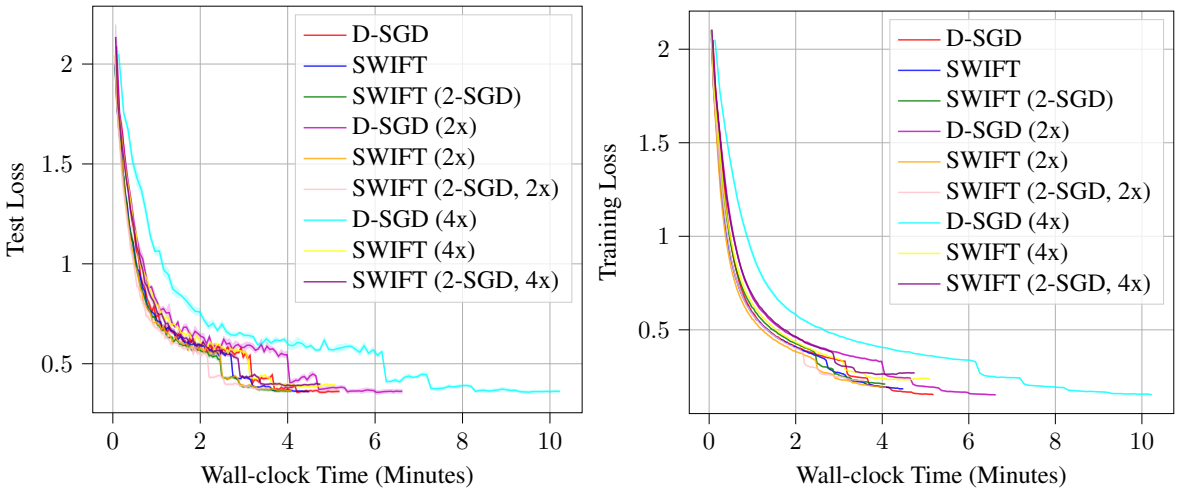


(c) 7/10 degree non-IID data.



(d) 9/10 degree non-IID data.

Figure 2.4: Average CIFAR-10 Test Loss for Varying non-IIDness on 10 Devices (ROC-3C).



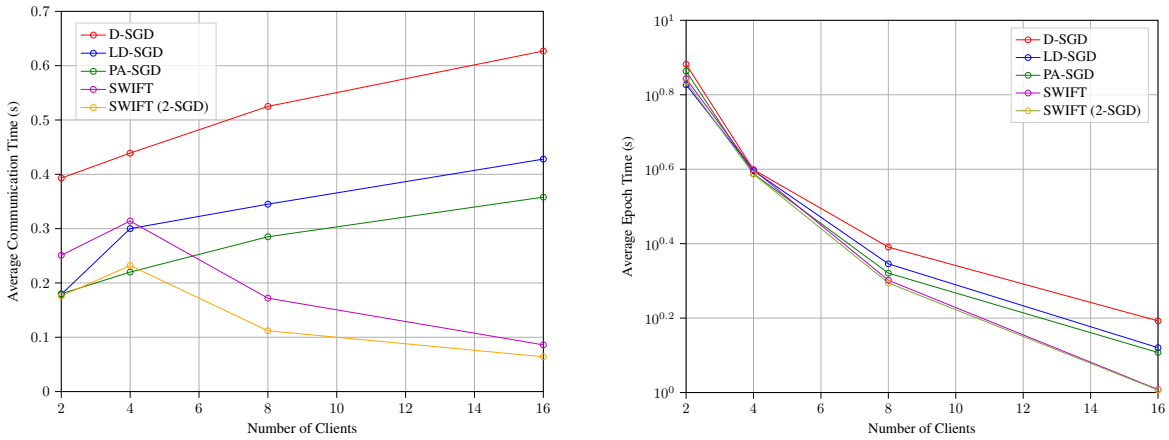
(a) Test loss slowdown.

(b) Train loss slowdown.

Figure 2.5: SWIFT vs. D-SGD for CIFAR-10 in 16 Device (Ring) with Varying Slowdown.

Varying Heterogeneity of Devices. In this experiment, the performance of SWIFT under varying heterogeneity, or speed, of devices (causing different delays) is investigated. This is done with 16 devices in a ring topology. An artificial slowdown is added, suspending execution of one of the devices such that it takes a certain amount of time longer (slowdown) to perform the computational portions of the training process. Tests are performed in the case where a device is two times (2x) and four times (4x) as slow as usual. Then, the performance of decentralized algorithms under these circumstances is compared to the normal setting with no added slowdown.

In Table 2.3, the average epoch, communication, and total time is displayed. Average total time includes computations, communication, and any added slowdown (wait time). SWIFT avoids large average total times as the slowdown grows larger. The wait-free structure of SWIFT allows all non-slowed devices to finish their work at their own speed. All other algorithms require devices to wait for the slowest device to finish a mini-batch before proceeding. At large slowdowns (4x), the average total time for SWIFT is nearly *half* of those for synchronous algorithms. Thus, SWIFT is very effective at reducing the run-time when devices are slow within the network.



(a) Communication time.

(b) Epoch time.

Figure 2.6: Average Communication and Epoch Times for Increasing Numbers of Devices.

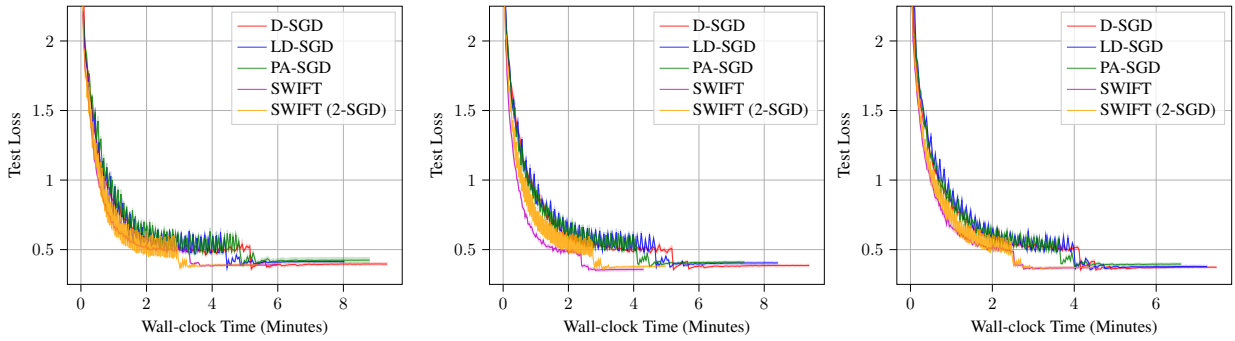
Figure 2.5 shows how SWIFT is able to converge faster to an equivalent, or smaller, test loss than D-SGD for all slowdowns. In the case of large slowdowns (4x), SWIFT significantly outperforms D-SGD, finishing in better than half the run-time. Other baseline algorithms are not included in order to avoid overcrowding of the plotting space. However, SWIFT also performs much better than PA-SGD and LD-SGD as shown in Table 2.3. These results show that the wait-free structure of SWIFT allows it to be efficient under device slowdown.

2.4.3 Varying Numbers of Devices & Network Topologies

Varying Numbers of Devices. In the fourth experiment, the performance of SWIFT performs versus other baseline algorithms is analyzed as the number of devices is varied. SWIFT reduces the time per epoch in a near-linear manner as the number of devices grow in size. For all baselines, there is parallel overhead when the number of devices is small. However, this overhead reduces as the number of devices grow. In comparison to the synchronous algorithms, SWIFT actually *decreases* its communication time as the number of devices increases. This allows the parallel performance to be quite efficient, as shown in Figure 2.6.

Table 2.4: Average CIFAR-10 Epoch & Communication Times with Varying Devices (Ring).

Decentralized FL Algorithms	16 Client Ring		8 Client Ring		4 Client Ring		2 Client Ring	
	Epoch (s)	Comm. (s)	Epoch (s)	Comm. (s)	Epoch (s)	Comm. (s)	Epoch (s)	Comm. (s)
SWIFT ($\mathcal{C}_0$)	**1.019**	**0.086**	**2.003**	**0.172**	**3.964**	**0.314**	**6.971**	**0.251**
D-SGD ($\mathcal{C}_0$)	1.558	0.627	2.459	0.525	3.970	0.439	7.62	0.393
SWIFT ($\mathcal{C}_1$)	**1.016**	**0.064**	**1.970**	**0.112**	**3.862**	0.232	6.83	**0.176**
LD-SGD ($\mathcal{C}_1$)	1.320	0.428	2.217	0.345	3.946	0.300	**6.712**	0.179
PA-SGD ($\mathcal{C}_1$)	1.281	0.358	2.093	0.285	3.871	**0.220**	7.303	0.180



(a) ROC-2C ($n = 16$). (b) ROC-4C ($n = 16$). (c) Ring ($n = 16$).

Figure 2.7: Average Test Loss for Varying Network Topologies on CIFAR-10.

Varying Topologies. The fifth, and final, experiment analyzes SWIFT's effectiveness versus other baseline decentralized algorithms under different, and more realistic, network topologies. In this experiment setting, 16 devices are trained on varying network topologies (Table 2.5 and Figure 2.7). SWIFT is robust to network topology. Regardless of the layout or sparsity of the network, SWIFT outperforms its baselines in terms of test loss and communication time.

Table 2.5: Average CIFAR-10 Epoch & Communication Times for Varying Topologies.

Decentralized FL Algorithms	16 Client ROC-2C		16 Client ROC-4C		16 Client Ring	
	Epoch (s)	Comm. (s)	Epoch (s)	Comm. (s)	Epoch (s)	Comm. (s)
SWIFT ($\mathcal{C}_0$)	**1.793**	**0.416**	**1.291**	**0.124**	**1.367**	**0.121**
D-SGD ($\mathcal{C}_0$)	2.799	1.479	2.813	1.464	2.241	0.962
SWIFT ($\mathcal{C}_1$)	**1.611**	**0.295**	**1.494**	**0.174**	**1.348**	**0.085**
LD-SGD ($\mathcal{C}_1$)	2.408	0.987	2.525	1.105	2.172	0.517
PA-SGD ($\mathcal{C}_1$)	2.639	0.765	2.216	0.708	1.982	0.500

Chapter 3: Large-Scale Distributed Learning via Private On-Device Locality-Sensitive Hashing[1]

This chapter focuses on alleviating the memory constraints that many edge devices face when training large-scale AI models. The methodology utilized to accomplish this goal, locality-sensitive hashing (LSH), has already been proven to be a remarkably effective tool for memory- and computationally-efficient data clustering and nearest neighbor search [73, 74, 75]. Locality-sensitive hashing algorithms, such as SimHash [73], are used to search for vectors in a big collection $W \subset \mathbb{R}^{d \times n}$ that form a large inner product with a reference vector $x \in \mathbb{R}^d$. This procedure, known as maximum inner product search (MIPS) [76], has been applied to neural network training. In neural network training, the weights of a dense layer that are estimated to produce a large inner product with the input (*i.e.,* a large activation) are used while the remaining weights are pruned, or removed.

While LSH-based pruning greatly reduces training costs associated with large-scale models, popular frameworks such as SLIDE [77] and Mongoose [78] cannot be deployed in distributed settings on memory-constrained devices such as GPUs or mobile phones for the following reasons: **(a)** required maintenance of a large target layer in memory and **(b)** access to the input is necessary to conduct locality-sensitive hashing.

[1] A version of Chapter 3 has been published in NeurIPS 2023 [72]

With many modern neural-network architectures reaching billions of parameters in size, requiring resource-constrained devices to conduct LSH analysis over even part of such a large model is infeasible. Doing as such requires many linear projections of massive weights. The hope of offloading this memory- and computationally-intensive task to a central host in the distributed setting is equally fruitless: LSH-based pruning cannot be conducted by a central host because it requires access to either local device data or hashed mappings of such data. Both of these violate the fundamental host-device privacy contract, especially in a federated setting [67]. Therefore, in order to maintain privacy, devices are forced to conduct LSH themselves, returning back to the original drawback in a vicious circle. Thus, the following question is raised:

> *Can a resource-constrained device conduct LSH-like pruning of a large dense layer without ever needing to see the entirety of its underlying weight?*

This chapter makes the following contributions to positively resolve this question:

- ▷ Introduce a novel family of hash functions, PGHash, for the detection of high cosine similarity amongst vectors. PGHash improves upon the efficiency of SimHash by comparing binarized random projections of *folded* vectors.
- ▷ Present an algorithmic locality-sensitive hashing framework, leveraging our hash functions, which allows for private, personalized, and memory-efficient distributed training of large-scale recommender networks via dynamic pruning.
- ▷ Showcase that PGHash can efficiently train large-scale recommender networks. This approach is competitive against distributed SLIDE [77] using full-scale LSH. Furthermore, where entry-magnitude similarity is desired over angular similarity, the DWTA [79] variant of PGHash, PGHash-D, empirically matches the performance of using full-scale DWTA.

3.1 Preliminaries

3.1.1 Problem Formulation

Let $W = \{w_1, w_2, \ldots, w_n\} \subset \mathbb{R}^d$ be the weights of a dense hidden layer and $x \in \mathbb{R}^d$ be the input. For brevity, $W \in \mathbb{R}^{d \times n}$ is referred to as the *weight* of the layer. In particular, $w_i \in \mathbb{R}^d$ corresponds to the i^{th} neuron of the layer. It is assumed that the layer contains dn parameters. Within PGHash, MIPS is performed, as the weights that produce large inner products with x are selected. Let $p = \max w_i^\top x$ for $1 \leq i \leq n$. For $0 < \epsilon < 1$ the goal is to select $\mathcal{S} \subset W$, such that for $\forall w_i \in S$, $w_i^\top x > \epsilon p$. The weights of $\mathcal{S}$ will pass through activation while the rest are dropped out, reducing the computational complexity of the forward and backward pass through this layer. As detailed in Chapter 3.2 and illustrated in Figure 3.1, $\mathcal{S}$ is determined by estimating angles with a projection $BW = \{Bw_i\}_{i=1}^n$, with $B \in \mathbb{R}^{c \times d}$ such that $c << d$.

3.1.2 Locality-Sensitive Hashing

Locality-sensitive hashing [75] is an efficient framework for solving the ϵ-approximate nearest-neighbor search (NNS) problem:

Definition 1 (ϵ-NNS). *Given a set of points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space $\mathcal{M}$ and similarity funcion $Sim : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ over this space, find a point $p \in P$, such that for a query point $q \in X$ and all $p' \in P$, $Sim(p, q) \leq (1 + \epsilon)Sim(p', q)$.*

It is important to note that the similarity function $Sim(\cdot)$ need not be a distance metric, but rather any general comparison mapping. Popular choices include Euclidean distance and cosine similarity, the latter of which is the primary focus of this chapter. Cosine similarity for $x, y \in \mathbb{R}^d$

is defined as $\cos(x, y) \triangleq x^\top y / \big(||x||_2 ||y||_2\big)$. The MIPS problem described previously can be framed as an ϵ-NNS problem if the weights $w_i \in W$ are assumed to be unit length. Thus, the aim is to search for ϵ-nearest neighbors in W of the query x according to cosine similarity.

Consider a family $\mathcal{H}$ containing hash functions of the form $h : \mathcal{M} \rightarrow \mathcal{S}$, where $\mathcal{S}$ is a co-domain with significantly lower feature dimensionality than $\mathcal{M}$. $\mathcal{H}$ is locality-sensitive if the hashes of a pair of points x, y in $\mathcal{M}$, computed by an $h \in \mathcal{H}$ (selected uniformly at random), have a higher collision (matching) probability in S the more similar x and y are according to Sim. Now, this notion is formally defined following the work of Indyk and Motwani [74]:

Definition 2 (Locality-sensitive Hashing)**.** *A family $\mathcal{H}$ is called $(S_0, \epsilon S_0, p_1, p_2)$-**sensitive** if for any two points $x, y \in \mathbb{R}^d$ and h chosen uniformly at random from $\mathcal{H}$ satisfies the following,*

1. *if $Sim(x, y) \geq S_0 \Rightarrow Pr(h(x) = h(y)) \geq p_1$,*
2. *if $Sim(x, y) \leq \epsilon S_0 \Rightarrow Pr(h(x) = h(y)) \leq p_2$.*

For effective locality-sensitive hashing, $p_1 < p_2$ and $\epsilon < 1$ is required. An LSH family provides an avenue to conduct a similarity search over a collection of vectors through a comparison of their hashed mappings. Of course, locality loss is inevitable if Sim is a dimension-lowering projection. Through a mixture of increased precision (raising the output dimension of $\mathcal{H}$) and repeated trials (running several trials over independently chosen h), the correspondence between Sim and matches over $\mathcal{H}$ can be tightened following the Johnson-Lindenstrauss Lemma [80].

Definition 3 (Hash table)**.** *Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and $V = \{v_1^\top, \ldots, v_k^\top\} \subset \mathcal{N}(0, I_d)$, where k is the **hash length**. Define $h_V : \mathbb{R}^d \rightarrow \mathbb{R}^k$ by $[h_V(x)]_i = h_{v_i}(x)$, where $h_{v_i} \in \mathcal{H}^{Sim}$, $1 \leq i \leq k$. The **hash table** $h_V(X) \in \mathbb{R}^{k \times n}$ is a binary matrix with columns $h_V(x_j)$, $1 \leq j \leq n$.*

3.1.2.1 SimHash

A popular LSH algorithm for estimating cosine similarity is SimHash, which uses signed random projections [73] as its hash functions. Specifically, for a collection of vectors $W \subset R^d$, the SimHash family $\mathcal{H}^{Sim}$ consists of hash functions h_v, each indexed by a random Gaussian vector $v \sim \mathcal{N}(0, I_n)$, i.e., an n-dimensional vector with iid entries drawn from $\mathcal{N}(0, 1)$. For $x \in R^n$, the hash mapping is defined as $h_v(x) := \text{sgn}(v^\top x)$. Here, sgn$(x)$ is modified to return 1 if $x > 0$, else it returns 0. For Gaussian v chosen uniformly at random and fixed $x, y \in \mathbb{R}^d$, the following holds:

$$Pr(h_v(x) = h_v(y)) = 1 - \frac{\arccos(\frac{x^\top y}{||x|| \cdot ||y||})}{\pi}. \tag{3.1}$$

This hashing scheme was popularized in [81] as part of a randomized approximation algorithm for solving MAX-CUT. Notice that the probability of a hashed pair matching is monotonically increasing with respect to the cosine similarity of x and y, satisfying Definition 2. More precisely, if $S_0 = \cos(x, y)$ then $\mathcal{H}^{Sim}$ is $\left(S_0, \epsilon S_0, \left(1 - \arccos(\frac{S_0}{\pi})\right), \left(1 - \arccos(\frac{\epsilon S_0}{\pi})\right)\right)$-sensitive [73, 82]. The above discussion considers the sign of a single random projection, but in practice, multiple projections are performed.

Following the notation above, the similarity between an input $q \in \mathbb{R}$ and a collection of vectors $X \subset \mathbb{R}^d$ can be estimated by measuring the Hamming distances (or exact sign matches) between $h_V(q)$ and columns of $h_V(X)$. SimHash is now more discriminatory, as $\mathcal{H}^{Sim}$ can separate R^d into 2^k buckets corresponding to all possible length k binary vectors (which are referred to as **hash codes**). Finally, counting the frequency of exact matches or computing the average Hamming distance over several independently generated hash tables further improves

our estimation of closeness. Implementations of the well-known SLIDE framework [77, 83], which utilize SimHash for LSH-based weight pruning, require upwards of 50 tables.

3.1.2.2 Densified Winner-Take-All (DWTA)

Another popular similarity metric is to measure how often high-magnitude entries between two vectors occur at the exact same positions. The densified winner-take-all (DWTA) LSH family [84] estimates this similarity by uniformly drawing k random coordinates over W and recording the position of the highest-magnitude entry. Similar to SimHash, this process is repeated several times, and vectors with the highest frequency of matches are expected to have similar magnitude ordering. This type of comparative reasoning is useful for computer vision applications [85].

3.1.3 Related Works

LSH Families. Locality-sensitive hashing families are often used to efficiently solve the approximate nearest neighbors problem [74, 75, 86]. SimHash [73], based on randomized hyperplane projections, is used to estimate cosine similarity. Each SimHash function requires a significant number of random bits if the dimensionality of each target point is large. However, bit reduction using Nisan's pseudo-random generator [87] is often suggested [88, 89]. MinHash [90], a competitor to SimHash [82], measures Jaccard similarity between binary vectors and has been used for document classification. The Winner Take All (WTA) hash [91] compares the ordering of entries by magnitude (corresponding to Kendall-Tau similarity); such comparative reasoning has proven popular in vision applications [92]. However, it was observed that WTA was ineffective at differentiating highly sparse vectors leading to the development of Densified

WTA (DWTA) [79]. Since MinHash, WTA, and DWTA are better suited for binary vector comparison, and comparison over real-valued vectors is required, PGHash is founded on SimHash.

Hash-based Pruning. One of the earlier proposals of pruning based on input-neuron angular similarity via LSH tables is in [93], where a scheme for asynchronous gradient updates amongst multiple threads training over a batch along with hashed backpropagation are also outlined. These principles are executed to great effect in both the SLIDE [77] and Mongoose [78] frameworks for training extreme-scale recommender networks. Mongoose improves SLIDE by using an adaptive scheduler to determine when to re-run LSH over a layer weight, and by utilizing learnable hash functions. Both works demonstrated that a CPU using LSH-based dynamic dropout can achieve competitive training complexity against a GPU conducting fully-dense training. Reformer [94] uses LSH to reduce the memory complexity of self-attention layers.

Distributed Recommender Networks. Several works which prune according to input-neuron angular similarity estimations via locality-sensitive hashing utilize multiple workers on a single machine [77, 78, 83, 93]. Federated training of recommender networks is an emerging topic of interest, with particular interest in personalized training [95, 96] malicious devices [97, 98], and wireless unreliability [99]. D-SLIDE [100], which is the federated version of SLIDE, eases local on-device memory and computational requirements by sharding the network across devices. However, in the presence of low device numbers, the proportion of the model owned per device can still be taxing, whereas our compression is independent of the number of federated agents. In Xu et al. [101], devices query the server for weights based on the results of locality-sensitive hashing conducted using server-provided hash functions. This complete server control over the hashing family, and thus access to hash-encoding of local device data, is non-private and potentially open to honest-but-curious attacks.

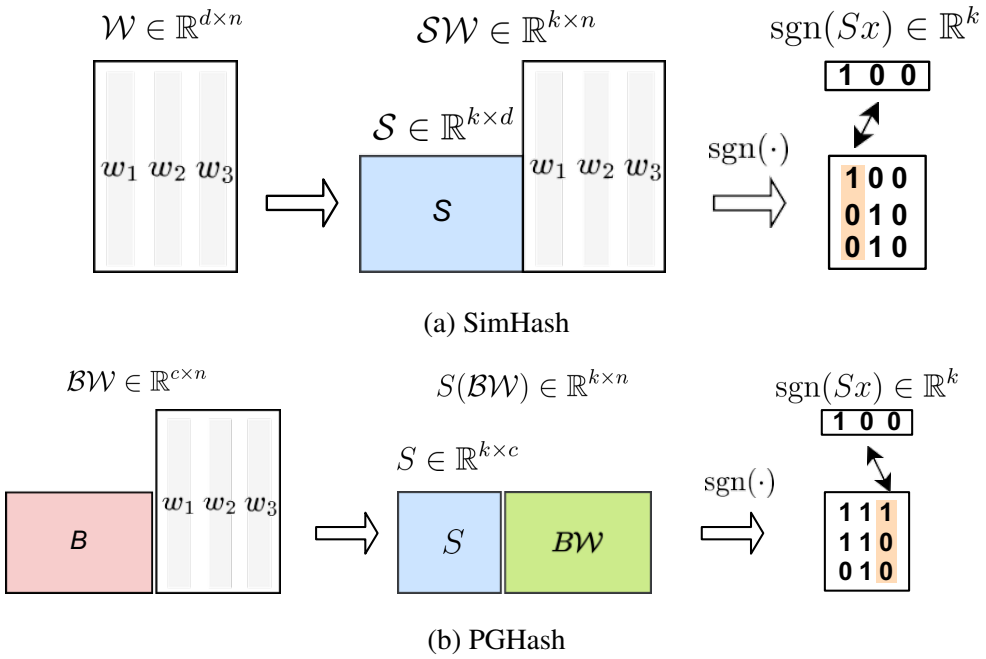


Figure 3.1: **Hash-based Dropout.** Two procedures for conducting cosine similarity estimation between full weight matrix $W \in \mathbb{R}^{d \times n}$ (column w_i denotes the weight of neurons i) and an input $x \in \mathbb{R}^d$. **(a)** SimHash generates a hash table via left-multiplication by a randomized rectangular Gaussian matrix $S \sim \mathcal{N}(0, I_d)$ onto the *fully-sized* W. **(b)** PGHash generates a hash table via left-multiplication by a randomized square Gaussian matrix $S \sim \mathcal{N}(0, I_c)$ onto a base projection $BW \in \mathbb{R}^{c \times n}$ of W. In both procedures, weight w_i is selected for activation if its signed hash matches the signed hash of x.

3.2 PGHash

Below, a family of hash functions $\mathcal{H}^{PG}$ allowing memory-efficient serial generation of hash tables using a dimensionally-reduced sketch of W is proposed. This is in contrast to traditional LSH frameworks, which produce hash tables over the *entirety* of W. First, an overview of the Periodic Gaussian Hash (PGHash) is presented, including its algorithm for distributed settings. Then, several statistical properties regarding the local sensitivity of $\mathcal{H}^{PG}$ are explored.

PGHash Motivation. As detailed in Chapter 3.1, the goal is to efficiently estimate cosine similarity between an input to a layer, $x \in \mathbb{R}^d$, and the columns of a large weight matrix $W \in \mathbb{R}^{d \times n}$. SimHash performs hash table generation by first multiplying a matrix of uniformly drawn Gaussian hyperplanes $T_V \in \mathbb{R}^{c \times d}$ with W. The full hash table is computed as $h_V(W) = \text{sgn}(T_V W)$. Then, the j^{th} neuron is activated if $\text{sgn}(T_V x) = h_V(w_j)$ for layer input x.

Generation of a new hash table $h_{\tilde{V}}(W)$ requires both **(i)** computation of $T_{\tilde{V}} W$, which requires access to the fully-sized weights, and **(ii)** the storage of $T_{\tilde{V}}$ to compute $\text{sgn}(T_{\tilde{V}} x')$ for further inputs x'. *This is problematic for a memory-constrained device, as it would need to maintain both W and $T_{\tilde{V}}$ to generate further tables and perform dynamic pruning.* To solve this issue, a family of hash functions generated from a single projection BW of W is introduced.

Definition 4 (Periodic Gaussian Hash)**.** *Let **sketch dimension** $c << d$ divide d for simplicity. Let $B = [I_c | I_c | \cdots | I_c] \in \mathbb{R}^{c \times d}$. I_c is the $c \times c$ identity matrix and $|$ denotes $\frac{d}{c}$ concatenations. Let $S \in \mathbb{R}^{k \times c}$ be a random Gaussian matrix with iid entries drawn from $\mathcal{N}(0, 1)$. A **Periodic Gaussian Hash (PGHash)** function is defined as $h_S^{PG} : \mathbb{R}^d \to \mathbb{R}^k$ by $[h_S^{PG}(x)]_i = [sgn(SBx)]_i$ for $1 \leq i \leq k$. Let the family of all such hash functions be $\mathcal{H}^{PG}(c, d)$.*

3.2.1 Theoretical Analysis

The term "periodic" is used to describe the hash functions described in Definition 4. Unlike SimHash, which projects a point via a fully random Gaussian vector as in SimHash, the projection is accomplished using a repeating, or periodic, concatenation of a length c Gaussian vector. Furthermore, for $h_S^{PG} \subset H^{PG}(c, d)$, the matrix representation SB is a tiling of a single Gaussian matrix. Notice that one may easily extend the notion of a PGHash of one vector to an entire hash table over multiple vectors following Definition 3. In this manner, a sequence of hash tables $\{h_{S_i}^{PG}(W)\}_{i=1}^{\tau}$ is generated over a weight matrix W simply by drawing random Gaussian matrices $S_i \in \mathbb{R}^{k \times c}$ for $1 \leq i \leq \tau$ (where k is the hash length) and computing $\text{sgn}(S_i BW)$.

Extension to DWTA (PGHash-D) Remark. When DWTA (described in Chapter 3.1) is preferred over SimHash for similarity estimation, one may modify Definition 4 as follows: $B = D_{\mathbb{1}} P$ where P is a $d \times d$ random permutation matrix, and $D_{\mathbb{1}}$ is a $c \times d$ rectangular diagonal matrix with $D_{ii} = 1$ for $1 \leq i \leq c$. These new hash functions are denoted as $h_S^{PG-D} : \mathbb{R}^d \to \mathbb{R}^k$ by $h_S^{PG-D}(x) = \max_i [SBx]_i$ for $1 \leq i \leq k$, with $k \leq c$, where S is now a rectangular permutation matrix which selects k rows of Bx at random. This scheme is referred to as PGHash-D, whereas PGHash refers to Definition 4.

Local Memory Complexity Remark. When generating a new table using PGHash, a device maintains S and only needs access BW, which costs $\mathcal{O}(kc)$ and $\mathcal{O}(cn)$ space complexity respectively. This is smaller than the $\mathcal{O}(kd)$ and $\mathcal{O}(dn)$ local memory requirements of SimHash, thereby allowing resource-constrained devices the ability to run locality-sensitive hashing locally.

3.2.1.1 Sensitivity Analysis

In this chapter, the sensitivity of PGHash functions $\mathcal{H}^{PG}(c,d)$ is explored.

Definition 5. *Let $x \in \mathbb{R}^d$ and $c \in \mathbb{R}$ such that $c|d$. Define the (d,c)-**folding** $x_c \in \mathbb{R}^c$ of x as $[x_c]_i = \sum_{j=1}^{\frac{d}{c}} [x]_{i+j\cdot\frac{d}{c}}$. Equivalently, $x_c = Bx$, with B as specified in Definition 4.*

Theorem 2. *Let $x, y \in R^d$. Define the following similarity function $Sim_c^d(x,y) \triangleq \cos(x_c, y_c)$, where x_c, y_c are (d,c)-foldings of x, y. $\mathcal{H}^{PG}(c,d)$ is an LSH family with respect to Sim_c^d.*

Proof. Let $h_v \in \mathcal{H}^{PG}$. Thus, for a randomly chosen $v' \sim \mathcal{N}(0, I_{\frac{d}{c}})$, v is a c-times concatenation of v'. One can see that $\text{sgn}(v^\top x) = \text{sgn}\left(\frac{v^\top x}{||v||\cdot||x||}\right) = \text{sgn}\left(\sqrt{\frac{c}{d}}\frac{||x_c||}{||x||}\frac{v'^\top x_c}{||v'||||x_c||}\right)$. Since sgn is unaffected by the positive multiplicative factors, it is concluded that $\text{sgn}(v^\top x) = \text{sgn}(v'^\top x_c)$. Through symmetric argument, $\text{sgn}(v^\top y) = \text{sgn}(v'^\top y_c)$. Since $v' \sim \mathcal{N}(0, I_c)$, comparing the sign of $v^\top x$ to $v^\top y$ is equivalent to a standard SimHash over x_c and y_c, i.e., estimation of $\cos(x_c, y_c)$. □

Corollary 2. *Let $x, y \in \mathbb{R}^d$, then $\mathcal{H}^{PG}(c,d)$ is $\left(S_c, \epsilon S_c, \left(1-\arccos(\frac{S_c}{\pi})\right), \left(1-\arccos(\frac{\epsilon S_c}{\pi})\right)\right)$-sensitive where $S_c = \cos(x_c, y_c)$,.*

Proof. This follows directly from the well-known sensitivity of SimHash [73]. □

The main result of Corollary 2 is that $\mathcal{H}^{PG}$ is LSH with respect to the angle between (d,c)-foldings of vectors. The use of periodic Gaussian vectors restricts the degrees of freedom (from d to d/c) of the projections. However, the usage of pseudo-random or non-IID hash tables has been observed to perform well in certain regimes [102, 103]. *Although $\mathcal{H}^{PG}(c,d)$ is LSH, is $\cos(x_c, y_c)$ necessarily an acceptable proxy for $\cos(x,y)$, in particular, for high angular similarity?* Heuristically, yes, for highly-cosine similar vectors: assuming x and y are both unit (since scaling does

Algorithm 5 Periodic Gaussian Hash (PGHash).
Input: batched input $X \in \mathbb{R}^{d \times M}$, projected weight $BW \in \mathbb{R}^{c \times n}$, compression rate CR, τ hash tables, hash length k, sketch dim c.

Set $\Theta = [\ ]$
for $i = 1, 2, \ldots, \tau$ **do**
    Draw random Gaussian $S \in \mathbb{R}^{k \times c}$
    for each sample x in X **do**
        for $j = 1, 2, \ldots, n$ **do**
            if $\text{sgn}(SBx) = [\text{sgn}(SBW)]_{:,j}$ **then**
                Θ.add(j)
    if $|\Theta| > \lfloor CR \cdot n \rfloor$ **then**
        Randomly remove selected neurons from table-i until $|\Theta| = \lfloor CR \cdot n \rfloor$
        Output: Θ
Output: Θ

not affect angle), then $||x - y||^2 = 2 - 2\cos(x, y)$. If ℓ_2 similarity between x and y is already high, then the ℓ_2 similarity of their (normalized) (d, c)-foldings will also be high, and thus their cosine similarity as well. Now, a characterization of the angle distortion of a (d, c)-folding is provided.

Theorem 3. *Let $x, y \in \mathbb{S}^{d-1}$. Assume that neither x nor y vanish under multiplication by B and define the set $S_{x,y} := \{v \in \text{span}(x, y) \ : ||v|| = 1\}$. Let $\lambda = d/c$, $\theta := \frac{1}{2}\arccos(\cos(x, y))$, $\alpha = \inf\{c > 0 \mid |||Bv|| \geq c, \forall v \in S_{x,y}\}$, and $\beta = \alpha/\lambda$. Assume $\alpha > 0$. Then $\cos(x_c, y_c)$ lives between $\frac{1-\beta^2\tan^2\theta}{1+\beta^2\tan^2\theta}$ and $-\frac{\tan^2\theta^2-\beta^2}{\tan^2\theta^2+\beta^2}$.*

Proof sketch. Without loss of generality, assume x and y are unit. Consider the unit circle $S_{x,y}$ contained in $\text{span}(x, y)$. The linear distortion $BS_{x,y}$ is an ellipse containing $x_c = Bx$ and $y_c = By$. The length of the axes of this ellipse is determined by the eigenvalues of $B^\top B$. The bounds follow from further trigonometric arguments, by considering when the axes of $BS_{x,y}$ are maximally stretched and shrunk respectively. These distortions are strongly related to λ and β.

One can see that as $\cos(x, y) \to 1$, then $\cos(x_c, y_c) \to 1$. It is natural to consider the

Algorithm 6 Distributed Periodic Gaussian Hash (PGHash).
Input: weights $\{W_0^\ell\}_{\ell=1}^L$, objective $\mathcal{F}$, N devices, target layer P, T iterations, folding matrix B, τ hash tables, hash length k, sketch dim c, comp. ratio CR.

Server Executes:
for $t = 1, 2, \ldots, T$ **do**
    Compile pre-target weights $W_A = \{W_t^\ell\}_{\ell=1}^{P-1}$
    Compile post-target weights $W_B = \{W_t^\ell\}_{\ell=P+1}^{L}$
    for each device i **in parallel do**
        $\Theta_i \leftarrow$ **DeviceLSH**(i, W_A, BW_t^P, CR, τ, k, c)
        $W_{t+1,i} \leftarrow$ **DeviceUpdate**(i, $W_t^P(\Theta_i) \bigcup W_B$)
    $W_{t+1} \leftarrow$ Average the active weights $W_{t+1,i}$
across all devices
Output: $\{W_T^\ell\}_{\ell=1}^L$

DeviceLSH($i, W_A, BW, CR, \tau, k, c$):
Sample input ξ from local data distribution $\mathcal{D}_i$
$\xi^{P-1} = \mathcal{F}(W_A, \xi)$ W_A is stored locally
$\Theta_i \leftarrow \texttt{PGHash}(\xi^{P-1}, BW, CR, \tau, k, c)$
Output: Θ_i

DeviceUpdate(i, W_B):
$W \leftarrow W_A \bigcup W_B$
Output: $W - \eta \nabla_W \mathcal{F}(W; \xi)$

distribution of α in Theorem 3 as how extreme shrinking by B (the folding matrix) can greatly distort the angle. This statistical distribution can be characterized exactly.

Proposition 1. *Let $u \in S^{d-1}$ be drawn uniformly at random. $||Bu||^2 \sim \text{Beta}(\frac{c}{2}, \frac{d-c}{2}, 0, \frac{d}{c})$, the four parameter Beta distribution pdf $f(x) = \frac{(2x/c)^{c/2-1}(1-2x/c)^{(d-c)/2-1}}{(d/c)\beta(c/2,(d-c)/2}$, and $\mathbb{E}||Bu||^2 = 1$.*

The proof of Proposition 1 is deferred to the Appendix B.4.3. Since the folded magnitude $||Bu||^2$ is unit in expectation, the distortion term α^2/λ^2 in Theorem 3 will often be close to 1, greatly tightening the angle distortion bounds.

3.2.2 PGHash Algorithm

Below, the protocol for deploying PGHash in a centralized distributed setting is detailed and presented algorithmically in Algorithm 6. Over a network of N devices, the central host identifies a target layer P whose weight $W_t^P \in \mathbb{R}^{d\times n}$ (at iteration t) is too expensive for memory-constrained devices to fully train or host in local memory. Neurons (columns of W_t^P) are pruned by devices according to their estimated cosine similarity to the output x^{P-1} of the previous layer.

The central host begins each round of distributed training t by sending each device **(1)** all weights $\{W_t^\ell\}_{\ell=1}^{P-1}$ required to generate the input x^{P-1} and **(2)** the compressed target layer BW_t^P. Using these weights, each device conducts PGHash analysis (via Algorithm 5) using its current batch of local data to determine its activated neurons. The central host sends each device i their set of activated neurons $W_t^P(\Theta_i)$, and each device performs a standard gradient update on their new model $\{W_t^\ell\}_{\ell=1}^{P-1} \bigcup W_t^P(\Theta_i) \bigcup \{W_t^\ell\}_{\ell=P+1}^{L}$. Finally, the central host receives updated models from each device and averages only the weights that are activated during training.

The on-device PGHash analysis (Algorithm 5) consists of first performing a forward pass

Table 3.1: Extreme Multi-Label Dataset Overview.

Dataset	Feature Dimensionality	Label Dimensionality	Number of Train Points	Number of Test Points
Delicious-200K	782,585	205,443	196,606	100,095
Amazon-670K	135,909	670,091	490,449	153,025
WikiLSHTC-325K	1,617,899	325,056	1,778,351	587,084

up to the target layer to generate the input x^{P-1}. Devices generate personal hash tables by performing left-multiplication of BW_t^P and x^{P-1} by a random Gaussian matrix $S \sim \mathcal{N}(0, I_c)$, as described in Chapter 3.1. The number of tables τ is specified by each device. A neuron j is marked as active if the input hash code sgn(Sx^{P-1}) is identical to j-th weight's hash code sgn$(SB[W_t^P]_{:,j})$. In Chapter 3.3.2.1, adapting PGHash to use Hamming distance for neuron selection is discussed.

Computational Complexity Remark. Through the use of dynamic pruning, PGHash significantly reduces both the forward and backward training computational complexities. PGHash activates *at most* $CR \cdot n$ neurons per sample as opposed to n for full training. In practice, PGHash activates only a fraction of the $CR \cdot n$ neurons (as shown in Figure 3.6a). Therefore, the number of floating point operations within forward and backward training is dramatically reduced.

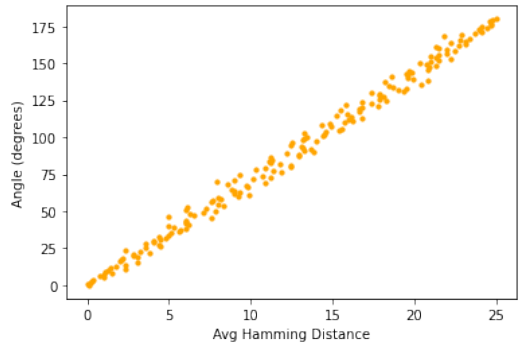
Communication Complexity Remark. By reducing the size of the model needed in local memory and subsequently requesting a pruned version of the architecture, communication efficiency is improved. For a fixed number of rounds T and target weight size dn, the total communication complexity, with respect to this data structure, is $\mathcal{O}(T \cdot CR \cdot dn)$. This is significantly fewer bits than the vanilla $\mathcal{O}(Tdn)$ communication cost of vanilla federated training. In Chapter 3.3, it is shown that PGHash achieves near state-of-the-art results with only $CR = 0.1$ (*i.e.,* 10% of a massive weight matrix).

3.3 Experiments

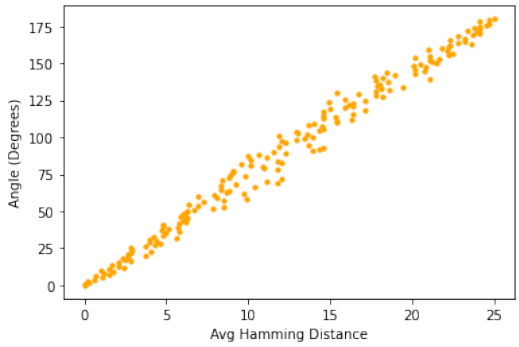
The goals of this chapter are to **(1)** gauge the sensitivity of PGHash and **(2)** analyze the performance of PGHash and its DWTA variant (PGHash-D) in training large-scale recommender networks. PGHash and PGHash-D require only 6.25% ($c = 8$) of the final layer sent by the server to perform on-device LSH in the experiments. In PGHash, devices receive the compressed matrix $BW \in \mathbb{R}^{c \times n}$ via the procedure outlined in Chapter 3.2. In PGHash-D, devices receive c out of d randomly selected coordinates for all n neurons in the final layer weight. Using k of the c coordinates, ensuring privacy since the server is unaware of the coordinates used for LSH, PGHash-D selects neurons that, within the k coordinates, share the same index of highest-magnitude entry between the input and weight. PGHash is tested on Delicious-200K [104] while PGHash-D is tested on Amazon-670K [105] and WikiLSHTC-325K [106].

3.3.1 Sensitivity Analysis

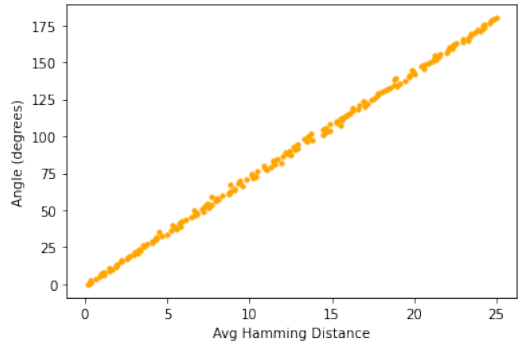
The first experiment measures the ability of $\mathcal{H}^{PG}(c, d)$ to estimate cosine similarity. A fixed unit vector $x \in \mathbb{R}^{100}$ and random set of 180 vectors $\{v_i\}_{i=1}^{180}$ of the same dimension are generated. Both the Gaussian vector x and collection of vectors V are fed through varying numbers of SimHash and PGHash tables. A resulting scatter plot measuring the correlation between the angle and average Hamming distance is plotted in Figure 3.2. Figure 3.2 provides the following insight: **(1)** PGHash is an effective estimator of cosine similarity. Like SimHash, PGHash successfully produces low average Hamming distances for vectors that are indeed close in angle. **(2)** Selecting neurons with exact hash code matches (vanilla sampling) effectively finds neurons close in angle to the input vector. **(3)** Increasing the number of hash tables indeed reduces variance.



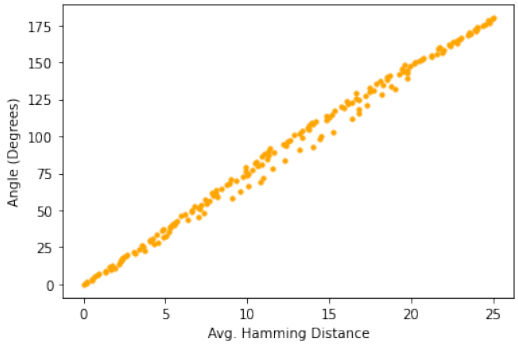
(a) SimHash (10 tables).



(b) PGHash (10 tables).



(c) SimHash (100 tables).



(d) PGHash (100 tables).

Figure 3.2: **Correlation Between Angle and Hamming Distance.** The average Hamming distance (x-axis) between a PG/SimHash hashed fixed vector x and collection of vectors $\mathcal{V}$ versus their true angles (y-axis) is plotted. Vectors are unit, length 100, and hashed to dimension $k = 25$ binary vectors via $\tau = 10$ or 100 hash tables. PGHash's sketch dimension is $c = 25$. PGHash and SimHash show strong correlation between Hamming distance and angular similarity.

3.3.2 Large-Scale Recommender Network Training

The second experiment tests how well PGHash(-D) can train large-scale recommender networks. These networks are trained efficiently by utilizing dynamic neuronal dropout as done in [77]. Three extreme multi-label datasets are used for training recommender networks: Delicious-200K, Amazon-670K, and WikiLSHTC-325K. These datasets come from the Extreme Classification Repository [107]. Both the feature and label dimensionalities of these datasets are large. Table 3.1 details the size of the extreme multi-label datasets tested in the experiments.

3.3.2.1 Experimental Setup

The feature and label sets of these datasets are extremely sparse. Akin to [77, 78, 108], a recommender network is trained using a fully-connected neural network with a single hidden layer of size 128. Therefore, for Amazon-670K, the two dense layers have weight matrices of size $(135,909 \times 128)$ and $(128 \times 670,091)$. The final layer weights output logits for label prediction, and PGHash(-D) is used to prune its size to improve computational efficiency during training.

Unlike [77, 78, 108], PGHash(-D) can be deployed in a federated setting. Within these experiments, the efficacy of PGHash is demonstrated for both single- and multi-device settings. Training in the federated setting (following the protocols of Algorithm 6) allows each device to rapidly train portions of the entire neural network in tandem. Data is partitioned evenly (in an IID manner) amongst devices. Finally, the neural network is trained using TensorFlow. An Adam [109] optimizer with an initial learning rate of 1e-4 is used. A detailed list of the hyperparameters used within the experiments can be found in Appendix B.1. Test accuracy referenced in the figures refers to the $P@1$ metric, which measures whether the predicted label with the

Table 3.2: **Average LSH time for PGHash over a range of tables.** The average μ time (and standard deviation σ) it takes for PGHash to perform *vanilla sampling* (exact matches) between the hash codes of sample x and each weight w_i in the final dense layer. Times are sampled for PGHash on Delicious-200k for batch size $M = 128$, $k = 9$, and $c = 8$ for one device.

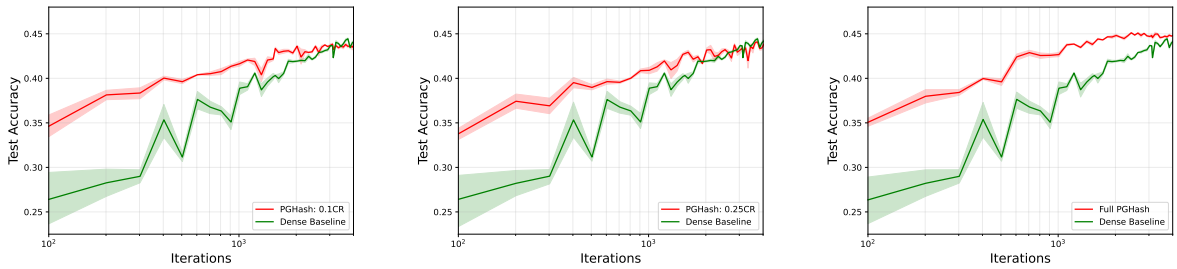
Method	**1 table (seconds)**	**50 tables (seconds)**	**100 tables (seconds)**
PGHash	$\mu = 0.0807, \sigma = 0.0076$	$\mu = 3.1113, \sigma = 0.0555$	$\mu = 6.2091, \sigma = 0.1642$
SLIDE	$\mu = 0.0825, \sigma = 0.0099$	$\mu = 3.2443, \sigma = 0.1671$	$\mu = 6.2944, \sigma = 0.0689$

highest probability is within the true list of labels. These experiments are run on a cloud cluster using Intel Xeon Silver 4216 processors with 128GB of total memory.

Sampling Strategy. One important aspect of training is how activated neurons are selected for each sample through LSH. Like [77], vanilla sampling is utilized. In the vanilla sampling protocol, a total of $CR \cdot n$ neurons are selected across the entire sampled batch of data. As detailed in Chapter 3.2 and Algorithm 5, a neuron is selected when its hash code exactly matches the hash code of the input. Neurons are retrieved until either $CR \cdot n$ are selected or all τ tables have been looked up.

Speed of Neuron Sampling. Table 3.2 displays the time it takes PGHash and SimHash to perform LSH for a set number of tables. These times were collected locally during training. The entries in Table 3.2 denote the time it takes to compute hashing of the final layer weights w_i and each sample x in batch M as well as vanilla-style matching (neuron selection) for each sample.

Table 3.2 shows that PGHash achieves near sub-linear speed with respect to the number of tables τ and slightly outperforms SLIDE. PGHash edges out SLIDE due to the smaller matrix multiplication cost, as PGHash utilizes a smaller random Gaussian matrix (size $c \times c$). The speed-up over SLIDE becomes more significant when the input layer is larger (as $d = 128$ in the experiments). Therefore, PGHash obtains superior sampling performance to SLIDE.

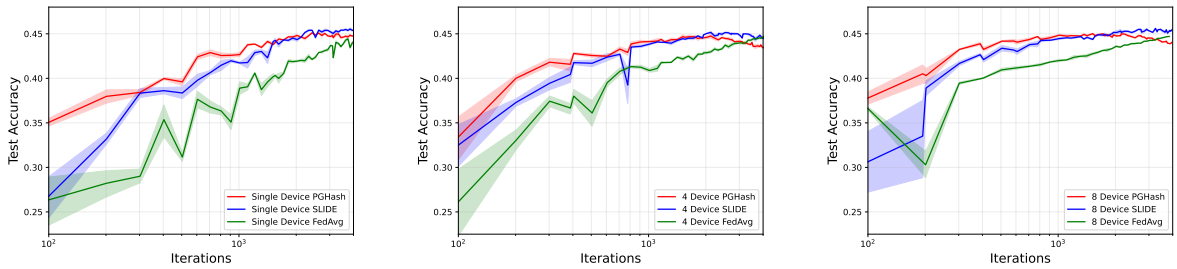


(a) 90% Compression. (b) 75% Compression. (c) No Compression.

Figure 3.3: **Compressed PGHash.** Model accuracy of a large recommendation system is recorded on an extreme classification task (Delicious-200K) using PGHash for varying compression rates (CR). Compressed PGHash, even at 90% compression, is competitive with full training (*without even including effects of sparsity-induced neuronal drop out*).

Hamming Distance Sampling. An alternative method to vanilla sampling is to instead select final layer weights (neurons) w_i which have a small Hamming distance relative to a given sample x. As a refresher, the Hamming distance simply computes the number of non-matching entries between two binary codes (strings). If two binary codes match exactly, then the Hamming distance is zero. In this sampling routine, either **(i)** the top-k weights w_i with the smallest Hamming distance to sample x are selected to be activated or **(ii)** all weights w_i with a Hamming distance of β or smaller to sample x are selected to be activated. Interestingly, the vanilla-sampling approach implemented is equivalent to using $\beta = 0$ in **(ii)**.

In either of the scenarios listed above, hash codes for w_i and x are computed as done in PGHash(-D). From there, however, the hash code for x is compared to the hash codes for all final layer weights in order to compute the Hamming distance for each w_i. The process of computing n Hamming distances for each sample x is very expensive (much harder than just finding exact matches). Therefore PGHash, as well as [77, 78], use vanilla sampling instead of other methods.



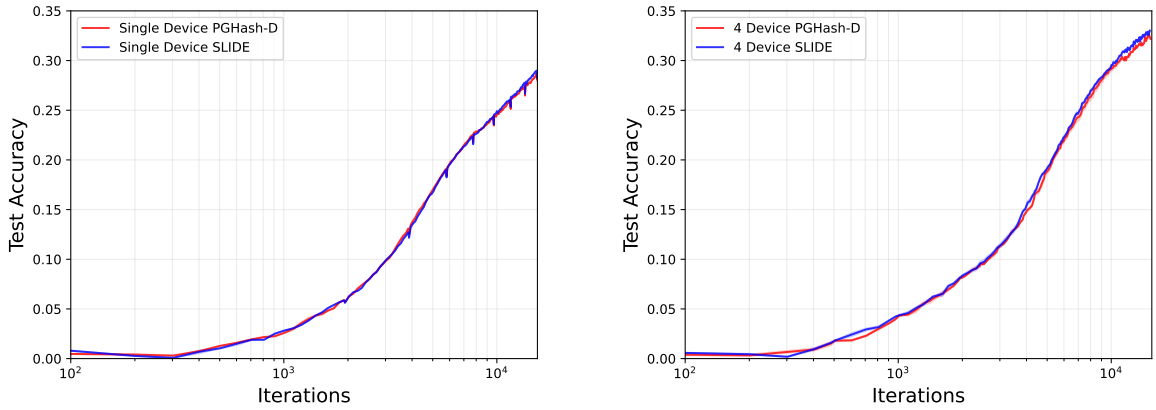
(a) 1 Device Results. (b) 4 Device Results. (c) 8 Device Results.

Figure 3.4: **Federated PGHash on Delicious-200K.** Model accuracy of a large recommendation system on an extreme classification task (Delicious-200K) trained in a federated setting is plotted. PGHash achieves competitive accuracies compared with Federated SLIDE and FedAvg. In fact, PGHash converges quicker to near-optimal accuracy.

3.3.2.2 PGHash Performance

Compression Performance. First, the performance of PGHash when varying the compression rate CR is analyzed. Figure 3.3 showcases how PGHash performs for compression rates of 75% and 90% as well as no compression. Interestingly, PGHash reaches near-optimal accuracy even when compressed. This demonstrates the effectiveness of PGHash at accurately selecting fruitful active neurons given a batch of data. The difference between the convergence of PGHash for varying compression rates lies within the volatility of training. As expected, PGHash experiences more volatile training (Figures 3.3a and 3.3b) when undergoing compression as compared to non-compressed training (Figure 3.3c).

Distributed Performance. In Figures 3.4 and 3.5, the performance of PGHash(-D) in federated settings is analyzed. PGHash(-D) is compared to a federated version of SLIDE [77] that is implemented (using a full SimHash or DWTA respectively), as well as fully-dense Federated Averaging (FedAvg) for Delicious-200K. One can immediately see in Figures 3.4 and 3.5 that PGHash(-D) performs identically to, or better than, Federated SLIDE. In fact, for Delicious-



(a) 1 Device Results.

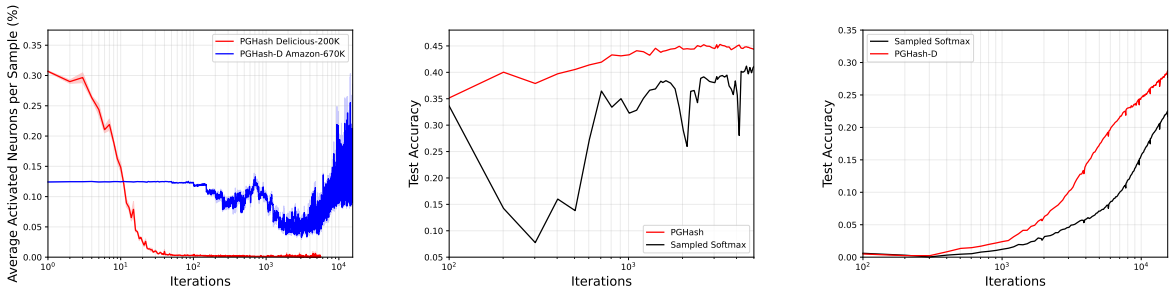
(b) 4 Device Results.

Figure 3.5: **Federated Amazon-670K PGHash-D.** Model accuracy of a large recommendation system is recorded on Amazon-670K trained in a federated setting. PGHash-D matches the convergence of Federated SLIDE without requiring LSH to be performed by the central server.

200K, PGHash and Federated SLIDE outperform the dense baseline (FedAvg). In Appendix B.2 the difficulties of PGHash-D and SLIDE in matching the dense baseline as well as the failure of SimHash to achieve performance akin to DWTA for Amazon-670K are detailed.

PGHash(-D) and Federated SLIDE smartly train portions of the network related to each batch of local device data, via LSH, in order to make up for the lack of a full output layer. However, unlike Federated SLIDE, PGHash(-D) can perform on-device LSH *using as little as 6.25% of the full weight* W ($c = 8$) for both Delicious-200K and Amazon-670K experiments. Furthermore, for Delicious-200K, PGHash generates a dense Gaussian that is only 6.25% ($c = 8$) the size of that for Federated SLIDE. In summary, PGHash(-D) attains similar performance to Federated SLIDE while storing less than a tenth of the parameters.

Induced Sparsity. PGHash(-D) induces a large amount of sparsity through its LSH process. This is especially prevalent in large-scale recommender networks, where the number of labels for each data point is a minuscule fraction of the total output layer size (*e.g.* Delicious-



(a) Average Activated Neurons. (b) Sampled Softmax (Delicious). (c) Sampled Softmax (Amazon).

Figure 3.6: **PGHash(-D) Computational Efficiency.** In Figure **3.6a**, PGHash(-D) activates only a fraction of the total final layer neurons *even without compression*. Through this induced sparsity, PGHash(-D) greatly reduces the computational complexity of forward and backward training compared to full training. In Figures **3.6b** and **3.6c**, PGHash(-D) is compared with the Sampled Softmax heuristic [3] (randomly sampling 10% of the total neurons). PGHash(-D) outperforms the Sampled Softmax baseline, selecting a better set of activated neurons, via LSH, to more efficiently train the recommender system.

200K has on average only 75.54 labels per point). PGHash(-D) performs well at identifying this small subset of neurons as training progresses. As one can see in Figure 3.6a, even when PGHash is allowed to select all possible neurons (*i.e.,* no compression $CR = 1$), it still manages to *select fewer than 1% of the total neurons after only 50 iterations of training on Delicious-200K*. For Amazon-670K, PGHash-D requires less than 30% of the total neurons for the majority of training. Therefore, PGHash(-D) greatly increases sparsity within the NN, improving the computational efficiency of the algorithm by reducing the number of floating point operations required in the forward and backward training.

Chapter 4: Shrinking the Size of Deep Extreme Multi-Label Classification[1]

Many online services use recommender systems to foster a personalized customer experience by assigning each product an individual label. These systems have grown to extreme sizes as online services have expanded the past two decades. For example, there are over 600 million products for sale on Amazon alone [111]. The output dimension of recommender systems is equivalent to the number of recommended products, so the computational and memory costs become extreme when these methods are applied to systems with millions of products. Fittingly, the name for this situation is Extreme Multi-Label Classification (XMC) [112].

Summary of Contributions. Similar to the previous chapter, the goal of this chapter is to alleviate the computational- and memory-constraints of XMC when training recommender systems that leverage modern deep-learning methods. Here, DECLARE: Deep Extreme Compressed Labeling And Recovery Estimation, a novel deep learning XMC algorithm is proposed. The main contribution of DECLARE is that it reduces the computational and memory costs of training large-scale recommender systems. Namely, DECLARE:

▷ Compresses the label dimension up to 99.975% by capturing key information within the compressed space that other compression methods fail to utilize,

▷ Trains a small neural network to learn the compression mapping from the feature space to the

[1] A version of Chapter 4 has been published in NeurIPS 2024 Compression Workshop [110]

compressed label space,

▷ Implements a recovery method to transform compressed predictions back to the uncompressed label space during inference time, without the need to train multiple networks,

▷ Closely matches uncompressed performance while reducing the label dimension by several orders of magnitude on five common XMC datasets.

4.1 Preliminaries

4.1.1 Problem Formulation

The objective of XMC is to train a classifier that maps text samples to labels within an extremely large label set. Denote $\mathcal{D} := \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ as the dataset, having n data samples. Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be the input feature vector while $\boldsymbol{y}_i \in \{0,1\}^L$ be its corresponding label vector. The sparse feature and label sets are defined as $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$ and $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]^T \in \mathbb{R}^{n \times L}$ Denote f as the objective function, defined below, $\theta \in \mathbb{R}^f$ as the parameters of a deep classifier, $h_\theta(\cdot)$ as the deep classifier parameterized by θ, and ℓ as the loss function.

$$\min_{\theta} f(\theta; \boldsymbol{x}, \boldsymbol{y}) := \frac{1}{n} \sum_{i}^{n} \ell\big(h_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i\big). \tag{4.1}$$

While Equation (4.1) can be solved through gradient descent approaches, the major issue at hand is that the label dimension, L, is large. As a result, the number of parameters, f, for the deep classifier is also large, resulting in expensive storage, inference $h_\theta(\boldsymbol{x})$, and gradient back-propagation $\nabla_\theta h_\theta(\boldsymbol{x})$.

4.1.2 Related Works

Deep XMC. Similar to the contents of this chapter, existing literature is concerned with memory constraints [72, 77, 78]. In Chen et al. [77], a locality-sensitive hashing scheme (SLIDE) is used to prune the final classification layer of an XMC network. However, the full-scale weight structure must be maintained in memory and the extent of pruning will vary batch-to-batch. Chen et al. [78] considers a staggered update of hashes to avoid continuous batch-to-batch re-hashing, but the full weight structure must still be occasionally restored, resulting in memory blowup. Rabbani et al. [72] designs a modified family of hashing functions to avoid the need for restoring or maintaining the full weight structure on a low-fidelity local device, but it requires the assistance of a centralized server to host the original weight. Yan et al. [100] proposes model splitting to alleviate the size of XMC layers across multiple nodes, though in the presence of few workers, the per-client share of the model is memory infeasible and thus inapplicable to standalone training.

Compressed Sensing. A subset of existing XMC literature focuses on compression of a large label space into a smaller one, known as compressed sensing [113, 114, 115, 116]. Zhou et al. [114] and Tai and Lin [115] utilize a random and SVD-based compression matrix to reduce the dimensionality of their label space. In Bhatia et al. [113], a low-rank embedding is optimized to preserve nearest-neighbor distances and sparsity. Unlike DECLARE, the methods above are not deep learning methods. As a result, these methods require a large number of training instances, because each input-class pair needs its own classification model. As the number of recommended products grows, so too do the number of training instances. This growth results in larger training and storage costs for large-scale recommender systems.

Algorithm 7 Deep Extreme Compressed Labeling And Recovery Estimation (DECLARE).
Input: Data (X_{trn}, Y_{trn}) & (X_{tst}, Y_{tst}), number of training samples n, original label dimension L, compressed label dimension c, q neighbors, top k, and T iterations.

Compute the truncated SVD: $U_c^Y, \Sigma_c^Y, (V_c^Y)^T \leftarrow$ TSVD(Y_{trn}, c)
Generate new compressed labels: $Z_{trn} \leftarrow U_c^Y * \Sigma_c^Y$
Train NN h_θ for T iterations (or convergence) on the compressed data (X_{trn}, Z_{trn})
for each test data sample $\boldsymbol{x}_i, \boldsymbol{y}_i \in (X_{tst}, Y_{tst})$ **do**
    Predict compressed label: $\boldsymbol{z}_i^p \leftarrow h_\theta(\boldsymbol{x}_i)$
    Compute similarity scores: $\boldsymbol{s}^{p,z} \leftarrow Z_{trn}\boldsymbol{z}_i^p$
    Find q-closest indices of Z_{trn}: $I^s \leftarrow \arg\max_{I \subseteq [n]:|I|=q} \sum_{i \in I} \boldsymbol{s}_i^{p,z}$
    Weight the true labels in I^s by their relative distances: $\boldsymbol{f}^p \leftarrow (\boldsymbol{s}_{I_s^p}^{p,z})^T Y_{I^s}$
    Select the top k largest label classes $I^y \leftarrow \arg\max_{I \subseteq [L]:|I|=k} \sum_{i \in I} \boldsymbol{f}_i^p$
    Compute and store the P@k accuracy

4.2 Compressed Labeling & Recovery Estimation

DECLARE performs information-aware compression of the original labels to produce compressed labels $\boldsymbol{z} \in \mathbb{R}^c$, where $c << L$. Then, DECLARE trains a feed-forward neural network, h_θ, to learn the mapping from $\boldsymbol{x}$ to $\boldsymbol{z}$. Finally, DECLARE uses a recovery-estimation scheme (Chapter 4.2.2) that takes a compressed label $\boldsymbol{z}$ as input and outputs the most-likely predicted label classes from its corresponding original label $\boldsymbol{y}$ (which is unknown during inference).

4.2.1 Label Compression via SVD Approximation

Because the labels, $\boldsymbol{y}$, are sparse, one can conjecture that the information stored in $\boldsymbol{y}$ can be distilled down into a much smaller, dense vector $\boldsymbol{z}$. Namely, the aim is to preserve the similarity between each label, $\boldsymbol{y}_i$, and all labels in the label set $\boldsymbol{y}_j, \forall j \in [n]$. In matrix form, this is denoted as YY^T. The following optimization problem is solved in order to determine the best low-rank approximation $Z \in \mathbb{R}^{n \times c}$ to Y,

$$\underset{\text{rank}(Z)\leq c}{\arg\min} \left\| YY^T - ZZ^T \right\|_F . \tag{4.2}$$

The best rank-c approximation of YY^T are the first c singular vectors and values (via the Eckart-Young-Mirsky Theorem [117]),

$$\underset{\text{rank}(Z)\leq c}{\arg\min} \left\| YY^T - ZZ^T \right\|_F = \underset{\text{rank}(Z)\leq c}{\arg\min} \left\| U\Sigma^2 U^T - ZZ^T \right\|_F = U_c \Sigma_c. \tag{4.3}$$

From Equation (4.3), the optimal method for label compression is to compute the truncated singular value decomposition (SVD) of Y, select the top c left singular vectors and singular values, and then perform the matrix product to obtain the compressed labels Z. In DECLARE, the original labels are compressed via Equation (4.3) to receive compressed labels Z, and train a neural network to learn the mapping between the feature and compressed label spaces $h_\theta(\boldsymbol{x}_i) = \boldsymbol{z}_i$.

4.2.2 Label Recovery

As the output of the neural network $h_\theta(\cdot)$ is within the compressed space, a label-recovery mechanism is necessary to recover the most-likely true predicted label classes in the original label space. As detailed in Chapter 4.2.1, the compressed labels Z are optimized to best approximate the similarity between each label in Y. Thus, by computing the inner product between a predicted compressed label $\boldsymbol{z}^p$ and all existing compressed labels Z, the similarity $\boldsymbol{s}$ between the original, and unknown, uncompressed prediction $\boldsymbol{y}^p$ with all labels Y is approximated,

$$\boldsymbol{s}^{p,z} = Z\boldsymbol{z}^p \approx Y\boldsymbol{y}^p = \boldsymbol{s}^{p,y}. \tag{4.4}$$

Table 4.1: DECLARE's Test Performance vs. (Un)compressed Baselines.

Dataset	DECLARE		Unstructured Pruning		Uncompressed	
	P@1	Output Dim.	P@1	Output Dim.	P@1	Output Dim.
BibTex	62.43%	100	62.73%	100	62.86%	159
EURLex-4.3K	83.24%	200	83.15%	200	86.65%	4,271
AmazonCat-13K	87.55%	200	82.52%	200	89.74%	13,330
Wiki10-31K	83.04%	100	55.01%	100	83.85%	30,938
Delicious-200K	46.17%	50	19.55%	50	46.99%	205,443

Equation (4.4) not only defines a relationship between $\boldsymbol{z}_p$ and $\boldsymbol{y}_p$, it directly relates the true and low-rank similarity vectors $\boldsymbol{s}^{p,y}, \boldsymbol{s}^{p,z} \in \mathbb{R}^n$. The indices I^s of the q largest values in $\boldsymbol{s}^{p,z}$ approximate the indices of the q most similar labels $Y_{I^s} := Y[I^s, :]$ within the dataset to the unknown true prediction $\boldsymbol{y}^p$,

$$I^s := \underset{I \subseteq [n]:|I|=q}{\arg\max} \sum_{i \in I} \boldsymbol{s}_i^{p,z}. \tag{4.5}$$

In summary, the top-q indices of $\boldsymbol{s}^{p,z}$ are computed and their corresponding labels are used within Y as the closest approximate labels Y_{I^s} to $\boldsymbol{y}^p$. Once the closest labels are found, they are scaled element-wise by their similarity scores and summed to determine the most frequent label classes,

$$\boldsymbol{f}^p := (\boldsymbol{s}^{p,z}_{I^p_s})^T Y_{I^s}, \quad I^y := \underset{I \subseteq [L]:|I|=k}{\arg\max} \sum_{i \in I} \boldsymbol{f}_i^p. \tag{4.6}$$

The indices I^y are the top-k largest label classes and are thus selected in DECLARE as the k most-likely true label classes. I^y is used to compute the Precision@k accuracy metric widely used within XMC settings. This metric gauges the percentage of k predicted label classes that are indeed found within a target label $\boldsymbol{y}_i$.

4.3 Experiments

DECLARE is compared with the original, uncompressed neural network as well as a neural network trained with unstructured pruning, one of the best deep-learning compression techniques. DECLARE is run on five common XMC datasets [112].

Datasets. Five publicly available datasets are tested: BibTex [118] (159 labels), EURLex-4.3K [119] (4,271 labels), AmazonCat-13K [105] (13,330 labels), Wiki10-31K [120] (30,938 labels), and Delicious-200K [104] (205,443 labels). Success across this variety of datasets (see Table 4.1) indicates that DECLARE will scale to real-world applications.

Unequal Baselines. DECLARE is at a disadvantage when compared to the uncompressed neural network or a neural network pruned via unstructured pruning. The uncompressed network is much larger and contains many more parameters than DECLARE. As a result, one would expect the uncompressed network to be an upper bound on how well DECLARE can perform.

However, DECLARE often matches, and is competitive with, the uncompressed network on many of the XMC datasets. Unstructured pruning, like DECLARE, trains a reduced (pruned) version of a model. Unstructured pruning serves as an upper bound for how well pruning-based compression performs. Unlike its structured counterpart, unstructured pruning sets a given number of parameters to zero (irregular sparsity) and generally performs better [121, 122]. As a result, the size of an unstructured pruning model is the same as the uncompressed network albeit with many more parameters set to zero. That type of model incurs larger memory costs than DECLARE. Furthermore, due to its irregular sparsity, unstructured pruning is more difficult to exploit for efficient computation [123]. Many modern deep learning packages, like TensorFlow and PyTorch, cannot handle irregular sparsity to speed up tensor computations. As a result, unstructured

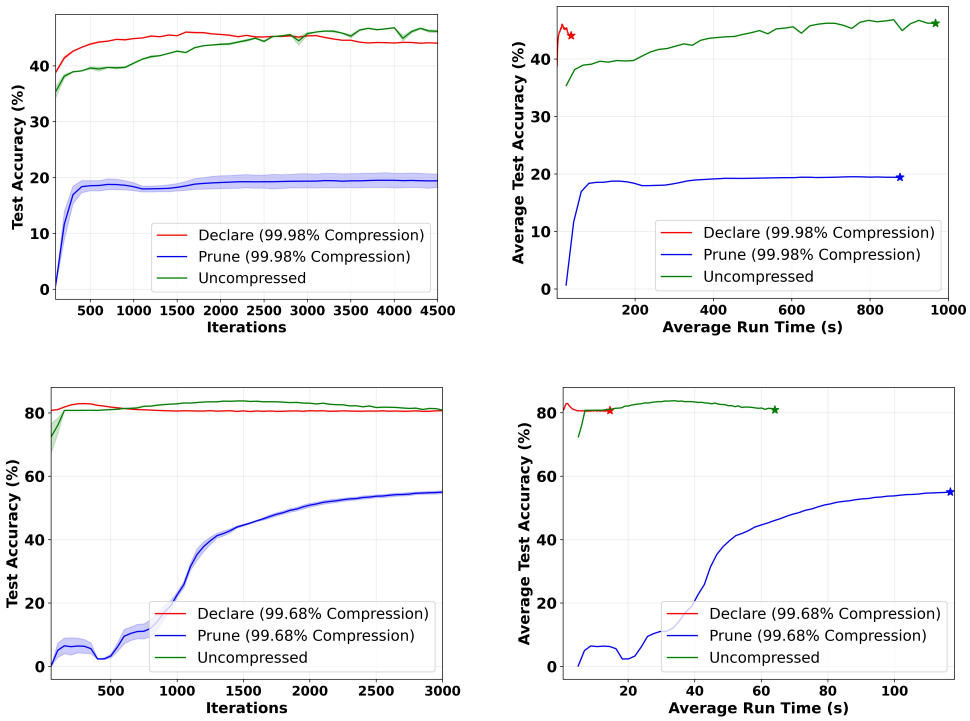


Figure 4.1: **Simultaneous Performance and Efficiency Improvement**. DECLARE outperforms its uncompressed counterpart by 5.28% and 18.36% on Delicious-200K (top) and Wiki10-31K (bottom) while reducing the final output dimension by 99.975% and 99.677% respectively. DECLARE outperforms the baselines in terms of test accuracy (left) and convergence (right).

pruning does not provide much computational speed-up, as shown in Figure 4.1.

Hyper-parameters & Evaluation. A feed-forward neural network is trained with one hidden layer having 256 neurons (following previous deep-learning XMC papers [72, 77, 78]). For BibTex, EURLex-4.3K, Wiki10-31K, and Delicious-200K a batch size of 128 is used while a batch size of 256 is used for AmazonCat-13K. The recovery scheme uses 25 nearest neighbors for all experiments, and the network is trained using ADAM with a standard learning rate of 5e-4. Each experiment is run three times, with the average results being shown in Table 4.1 as well as the dark lines in all Figures. The test accuracies listed are Precision@k (the results provided use $k = 1$), a metric widely used in XMC [72, 77, 78, 112, 113]. This metric gauges the percentage of k predicted label classes that are indeed found within a target label $\boldsymbol{y}_i$.

Computation and Memory Reduction Without Sacrificing Performance. The main takeaway from the experimental results is that DECLARE can dramatically reduce memory and computational costs while reducing performance by no more than 3% across all datasets. DECLARE can maintain performance while simultaneously reducing memory and computational costs by up to 99.975% versus an uncompressed neural network. On Delicious-200K, DECLARE achieves 46.17% P@1 accuracy compared to 46.99% for its uncompressed counterpart, while only requiring an output dimension of 50 (compared to the original dimension of 205,443). DECLARE also performs better as the output dimension grows (see Table 4.1). While unstructured pruning is competitive on smaller datasets, DECLARE outpaces it on the larger AmazonCat-13K and Delicious-200K datasets. DECLARE outperforms unstructured pruning on the larger datasets due to its distance-preserving compression scheme.

Improved Convergence Speed. Another advantage that DECLARE has over its peers is that it trains faster and converges quicker to peak performance. Figure 4.1 shows DECLARE outperforming its baselines at the beginning of training. DECLARE trains faster than the uncompressed network due its greatly reduced size (its final layer weight is reduced by upwards of four magnitudes). While unstructured pruning compresses the original network by a similar amount as DECLARE, its irregular sparsity requires a full forward pass since PyTorch does not support unstructured sparse matrix operations (as detailed above).

Part II

Mechanism Design for Edge Learning Fairness & AI Regulation

Part II Preface

Most edge-learning frameworks are susceptible to the free-rider dilemma: devices perform only a few, or no, gradient updates during training, yet still receive the global well-trained model by the end of training. Consequently, edge-learning frameworks do not ensure fairness. Free-riding devices that contribute little or nothing towards training still receive the same resultant model as those participatory devices that provide the bulk of training contributions.

Like fairness, AI regulation is another under-researched area within edge learning. Since unchecked AI models, such as large-language models, proliferate misinformation and stoke division within our society, it is imperative to employ a framework to mitigate these dangers and ensure user safety. While there is much-warranted discussion about how to address the safety, bias, and legal woes of state-of-the-art AI models, what is lacking are rigorous and realistic mathematical frameworks to regulate AI.

Thus, Part II of my dissertation tackles issues of **fairness and AI regulation on the edge** and within AI at large.

Chapter 5: FACT or Fiction: Can Truthful Mechanisms Eliminate Federated Free Riding?[1]

Truth is essential to the functioning of fair systems. In its absence, fraud and corruption lead to negative effects on those left participating. This too is true within AI. Adversaries can attack AI algorithms for insidious purposes [124, 125], and algorithms trained on poisoned or unrepresentative data can lead to dangerous and unfair results [126, 127, 128].

Issues of truthfulness have bled into multi-device collaborative learning systems. Federated learning, the preeminent edge-learning framework, is susceptible to the free-rider dilemma: devices can contribute little to nothing during training while still obtaining a well-trained final aggregated model [4, 129, 130]. Furthermore, Karimireddy et al. [4] proves that standard federated learning algorithms naturally yield catastrophic free-riding equilibria, where rational devices contribute *nothing*. It is important to note that the possibility of malicious gradient updates from adversarial devices, covered by recent works [131, 132, 133, 134], is not considered within this chapter. Instead, it is assumed that devices share a common goal of reducing loss, and act selfishly rather than maliciously.

Recent work has aimed to remedy the free-rider dilemma in federated learning through contract theory [38, 135, 136, 137], collaborative fairness [138, 139, 140], and mechanisms that

[1] A version of Chapter 5 has been published in NeurIPS 2024 [39]

incentivize contributions [4, 141]. The mechanisms presented in Karimireddy et al. [4] and Bornstein et al. [141] are notable because they do not require **(i)** carefully optimized contracts (devices will always participate because the mechanism is individually rational) or **(ii)** alterations to standard federated learning training (simple FedAvg [32] can be run with no device reputation scores). However, these mechanisms assume that devices are honest and truthful with a central server. If devices lie, then the proposed mechanisms no longer guarantee elimination of free riding.

In this chapter, the **F**ederated **A**gent **C**ost **T**ruthfulness (FACT) mechanism is proposed, which provably eliminates free riding *even when devices are untruthful*. Unlike standard federated learning approaches, where devices are incentivized to use little to none of their own data, devices participating in FACT are incentivized to use as much data for federated training as they would on their own. Furthermore, when devices use their locally optimal amount of data, FACT is individually rational (IR): devices always receive greater benefit from participating in FACT than by training alone. These results hold even if devices try to cheat the mechanism by misreporting individual training costs to the central server. FACT ensures that the best strategy for each device is to be truthful with the central server.

Summary of Contributions. In summary, the main contributions of this chapter are,

- ▷ Creating a novel penalization scheme that shifts the catastrophic free-riding optimum back to each device's local optimum (thereby eliminating free riding).
- ▷ Constructing a truthfulness mechanism (competition) that dissuades devices from lying about their individual costs with the central server.
- ▷ Showcasing the efficacy of FACT empirically in eliminating device free riding under untruthfulness, consequently improving federated training performance.

5.1 Preliminaries

5.1.1 The Challenge of Free Riding in Federated Learning

Federated Setting. This chapter, like Chapter 3, explores the setting of centralized federated learning (CFL), where n devices collectively train a model w under orchestration of a central server. Within each iteration t of centralized federated learning, the server sends down the current aggregated model w^t to each device. Devices then perform h local gradient updates (with their own local data) to generate an updated model $w_i^{t+1}\ \forall i \in [n]$. Models are aggregated by the server and this process is repeated until convergence. Each device is assumed to have access to the same data distribution $\mathcal{D}$, and thus the data is assumed to be independent and identically distributed (IID). The construction of truthful mechanisms in the non-IID setting if left for future research. This chapter aims to bring truthful mechanisms to centralized federated learning, as none have previously existed in either the IID or non-IID settings. Below, the objective that devices participating in federated learning seek to minimize is detailed.

Federated Objective. The goal of federated learning is to train a global model, leveraging the data and compute power of thousands of devices, that achieves low loss. To start, the convergence of distributed stochastic gradient descent (D-SGD) [30] to first-order stationary points in non-convex settings is presented,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(w^t)\right\|^2 \leq \frac{2\Delta_f}{\gamma T} + \frac{\gamma\sigma^2 L}{2\sum_{j=1}^{n} m_j}. \tag{5.1}$$

The parameter Δ_f is the difference between the optimal objective value f^* and starting value

$f(w^0)$. The global objective function f is the uniform sum over all device objective functions f_i. There are T total iterations. The step-size is γ and L is the Lipschitz constant, where $\gamma L < 2$. The effective gradient variance is $\sigma^2 / \sum_{j=1}^{n} m_j$. This stems from bounded variance assumptions, formally defined as $\mathbb{E} \left\| \frac{1}{M} \sum_{i=1}^{M} \nabla f(w; \xi_i) - \nabla f(w) \right\| \leq \frac{\sigma^2}{M}$ (for M batches of data ξ).

Local Device Loss. Each rational device seeks to minimize its loss. As shown in Equation 5.1, this can be accomplished by using more data, and thus a larger batch size m, as well as more rounds of gradient updates T. However, each device incurs a cost c_i for collection, sampling, and compute costs for each data sample m used. The linear cost $c_i m$ stems from the cross-device setting that is considered. Within this setting, IoT devices are prevalent and data is easy to collect (*e.g.,* IoT sensors). Thus, it makes the most sense that, on average, there are constant costs over time to collect and sample data (*e.g.,* the cost to power a sensor is usually constant on average) [142, 143, 144].

Overall, each device must balance the benefit of using more data to achieve better model performance with the costs of data collection, sampling, and computations. This trade-off is formulated for each device by defining a local loss function $\ell_{i,l}(m_i)$, that depends on the data collected m_i by each device i. Each device minimizes its loss function, which is a combination of data costs and convergence error terms. When training locally, or alone, a device loses the benefit of an increased batch size across all devices $\sum_{j=1}^{n} m_j$. Thus, after removing constants, Equation (5.1) transforms into the local device loss,

$$\ell_{i,l}(m_i) := \frac{\gamma \sigma^2 L}{2 m_i} + c_i m_i. \tag{5.2}$$

The optimal amount of data each device uses for local training is derived via calculus.

Theorem 4 (Optimal Local Data Usage)**.** *For an device i with marginal cost c_i, the optimal amount of data $m^*_{i,l}$ used for local training is $m^*_{i,l} := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$.*

Remark 1. *Devices' optimal amount of local data is inversely proportional to its cost c_i. Larger costs lead to less data used and vice versa. If costs are zero, devices use infinite data.*

The Free-Rider Dilemma: Shifted Optima in Federated Settings. When devices collectively engage in federated training, the effective batch size returns to the sum over all device data $\sum_{j=1}^{n} m_j$. By participating in federated training, devices gain the benefit of an increased batch size. As such, each device i's loss function in the federated setting becomes,

$$\ell_{i,F}(m_i) := \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i. \tag{5.3}$$

This transformation changes the optimal amount of data used by each device.

Theorem 5 (Free-Riding: Optimal Federated Data Usage)**.** *For a device i with marginal cost c_i, the optimal amount of data $m^*_{i,F}$ used for federated training is $m^*_{i,F} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}} - \sum \boldsymbol{m}_{-i}$.*

Remark 2. *By participating in federated training, each device is incentivized to use fewer data samples. This explicitly demonstrates the free-rider dilemma mathematically: it may be optimal for a device to use no data $m^*_{i,F} = 0$ (if $\sum \boldsymbol{m}_{-i} \geq \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$) and still possibly achieve lower cost overall (if $\sum \boldsymbol{m}_{-i} \geq m^*_{i,l}$).*

The proofs of Theorems 4 and 5 are found in Appendix C.2.

5.1.2 Related Works

The research area of incentives and mechanism design in federated learning is now growing. There are a few surveys [145, 146, 147] that overview federated learning mechanisms, current research directions and challenges, and future opportunities. Below, a few of these research directions are summarized, namely device contribution and truthful federated learning mechanisms, as well as fairness and device selection mechanisms.

Edge-Learning Mechanisms for Device Contribution. An important line of mechanism research in distributed learning regards mechanisms that incentivize device contributions for federated training [4, 141, 148, 149, 150]. Some of the first papers in this area were application-based, namely in mobile crowdsensing and smart sensors in vehicles.

Within Zhan et al. [149], an iterative Nash bargaining solution is proposed where mobile devices bargain with a server over the amount of data to contribute and the payments it should receive. The server dynamically alters the payment price to adjust its gap between desired supply of data and the given demand. Liu et al. [148] designs a reverse-auction approach, with subsidized or "endowment" compensation, that incentivizes non-participating vehicles to reduce roadside server congestion by offloading its data. Both works deal with data exchange and do not consider federated learning, where models are collaboratively trained together. Furthermore, it is not guaranteed in either that devices and the server will both always achieve higher payoff for participating than not (in Liu et al. [148] the server attains smaller payoff by participating).

Zhan et al. [150] is one of the first works to define device and server utility when data is shared amongst devices and the server to train a global model (including device-training costs). The server announces a total reward, and devices determine how much data to contribute in order

to maximize their reward. A deep reinforcement learning method is used by the server to learn the optimal announced reward.

An accuracy-shaping mechanism is proposed in Karimireddy et al. [4], which also follows a similar data sharing and IID setting. This mechanism does not require the use of contracts or learning of a reward, but rather guarantees that a device will receive an improved reward when participating in the mechanism than by not participating. In the accuracy-shaping mechanism, the server degrades model performance for a device if it does not contribute as much data as locally optimal. Bornstein et al. [141] propose a similar accuracy-shaping mechanism. However, unlike Zhan et al. [150] and Karimireddy et al. [4], device utility is modeled in a non-linear and more realistic manner. Additionally, the accuracy-shaping mechanism is generalized to the federated setting (no data sharing) where data can be non-IID. Unlike the majority of the works above, the mechanism proposed in this chapter seeks not just to incentivize device contribution, but to eliminate free riders. Moreover, unlike [4, 141], it is not assumed that devices are truthful. The proposed mechanism ensures that devices contribute what is locally optimal, thereby eliminating free-riding, and punishes devices for lying to the server about their costs.

Truthful Distributed Learning Mechanisms. Similar to the device contribution mechanisms above, many existing truthful distributed learning mechanisms exist within the crowdsensing setting [151, 152, 153, 154]. Xu et al. [153] deals with devices submitted untruthful bids to a server, specifically where a device can lie about sensing time windows and costs. The proposed mechanism leverages a Vickrey-Clarke-Groves auction, thereby ensuring truthfulness in bid price. Time-window truthfulness is ensured via trusted time stamping. Similarly, the remaining crowdsensing works leverage auctions to ensure device bid truthfulness. Zhang et al. [154] introduces a double auction mechanism, MobiAuc, for proximity-based mobile crowdsensing.

MobiAuc addresses issues of dynamic mobility patterns (device matching) while maintaining device truthfulness surrounding its bid (*e.g.,* service valuation or arrival time). In Wang et al. [152], a distributed truthful mechanism is proposed, where devices send bids for their services (usually data) to buyers. Gao et al. [151] designs a reverse auction mechanism that ensures vehicles report their true costs when crowdsensed. Again, the works above do not consider federated settings where models are trained collaboratively by many devices. Thus, these works do not solve the free-rider dilemma, which only exists in federated settings. Furthermore, the mechanism proposed in Chapter 5.3 is able to ensure device truthfulness without the use of an auction.

More recent works [143, 144, 155] examine truthful mechanisms within traditional federated learning. Lu et al. [143] explores truthful mechanisms in vertical federated learning, where features are split up amongst devices. Specifically, a truthful mechanism is designed (via a novel transfer-payment rule) to maximize social utility, without use of an auction, even when devices can lie about their costs. Within horizontal federated learning (the standard approach), Le et al. [155] designs a randomized auction for resource-constrained wireless devices, which achieves approximate-truthfulness in expectation. The server receives bids from devices to perform training, and selects a winning bid (device) for each uplink sub-channel. The server pays a reward designed to minimize device social cost, and does not take into account the server's utility. Wu et al. [144] builds on the auction idea in Le et al. [155], instead using a federated-auction bandit, which learns to select bidding devices that maximize the server's utility. The mechanism ensures truthfulness and individual rationality. Like the works above, the mechanism in this chapter ensures truthfulness and individual rationality of participating devices. In contrast, FACT is applied to horizontal federated learning and does not require an auction or bandit method to ensure truthfulness. As such, any willing device can participate in training (no bids are rejected), an optimal

server reward does not need to be learned, and devices do not need to have a private valuation. Finally, this mechanism requires no direct payment from the server (all reward payments come from other devices) and provably eliminates the free-riding dilemma.

Fairness Federated Learning Mechanisms. Recent work within federated learning has focused on device fairness [36, 37, 138]. These works seek to fairly allocate model performance to devices depending upon how much they contribute during federated training. Lyu et al. [138] computes a "reputation" metric for each device (measures the amount of device contribution). A better reputation leads to improved model performance. Both Blum et al. [36] and Murhekar et al. [37] seek to find device contribution equilibria that improve the welfare of all devices. Within [36], the existence and stability of equilibria which avoid free riding and device envy. In [37], a mechanism is proposed to alter device strategies in order to maximize the net utility of all participating devices. Unlike the works above, FACT is a truthful mechanism.

Mechanisms for Federated Device Selection. There exists recent literature which aims to incentivize high-quality devices to participate in federated training [38, 137, 156]. Devices are deemed as high quality if, for example, they have valuable data or are very willing to participate in training. In Liu et al. [137], a two-dimensional contract model is created to consider device willingness and data quality when designing contract fees. Kang et al. [38] introduces a scheme where device reputations are calculated (from previous interactions or other server interactions) and leveraged to select reliable devices for federated training. Zhou et al. [156] designs an auction method that incentivizes a diverse set of devices to participate in training. Truthfulness and individual rationality are ensured by properties of the constructed auction. FACT does not select devices based on a given quality or willingness criteria. All devices are able to participate if they choose. Instead, FACT solves device free riding when devices are truthful as well as untruthful.

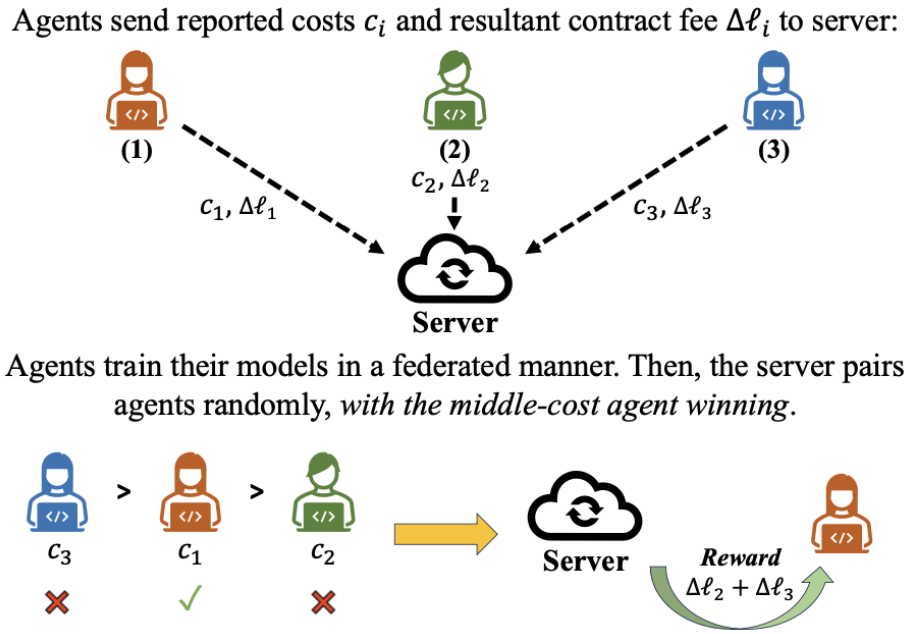


Figure 5.1: FACT Schematic.

5.2 Eliminating Free Riding via Penalization

The natural equilibrium of traditional federated learning is free riding: devices use less data than is locally optimal in proportion to how much data other devices use (Theorem 5). This necessitates the construction of a mechanism to restore the equilibrium of federated learning back to what is locally optimal for each device. In this chapter, penalization is introduced to enforce this restoration.

Penalized Federated Learning (PFL). To disincentivize devices from straying from their locally optimal data usage $m^*_{i,l}$, the central server can use contract theory to ensure each device i uses its locally optimal amount for a reported cost c_i. Namely, when agreeing to participate in federated learning, each device i agrees to pay the following free-riding penalty P_{fr} if they do not use $m^*_{i,l} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$ data samples,

$$P_{fr}(m_i) := \lambda_i\left(\frac{c_i}{2\lambda_i} - \frac{\gamma\sigma^2 L}{4\lambda_i(m^*_{i,l} + \sum \boldsymbol{m}_{-i})^2} + m^*_{i,l} - m_i\right)^2. \tag{5.4}$$

The hyperparameter $\lambda_i > 0$, chosen by the central server, denotes the harshness of the free-riding penalty. The value of λ_i is known by each device i while deciding whether or not to participate in the mechanism. Since the penalty is levied on each device by the server, it is incorporated into their federated loss function. This results in a new federated loss function $\ell_{i,PFL}(m_i)$ defined as,

$$\ell_{i,PFL}(m_i) = \underbrace{\frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i}_{\ell_{i,F}(m_i)} + P_{fr}(m_i). \tag{5.5}$$

Theorem 6 (PFL Eliminates Free Riding)**.** *For a device i with marginal cost c_i, the optimal amount of data $m^*_{i,PFL}$ used for federated training under Equation (5.5) is $m^*_{i,PFL} := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$.*

Remark 3. *By utilizing the PFL loss in Equation (5.4), devices participating in federated learning will use a locally optimal data amount ($m^*_{i,PFL} = m^*_{i,l}$). This eliminates the free-rider dilemma seen in Theorem 5.*

Devices participating in PFL (Algorithm 8) follow the amended loss shown in Equation (5.5). Careful selection of λ_i is required to ensure that PFL is Individually Rational.

Ensuring Individual Rationality (IR) in PFL. As shown in Theorem 6, it is optimal for devices participating in the penalized federated scheme (with a loss function shown in Equation (5.5)) to contribute what is locally optimal *i.e.,* $m^*_{i,PFL} = m^*_{i,l}$. To ensure individual rationality, it must be true that participating in federated training will generate at least as much reward for a device than by not participating. The selection of λ_i is crucial in accomplishing this task.

Algorithm 8 Penalized Federated Learning (PFL).
Input: data $\boldsymbol{m} = [m_1, \ldots, m_n]$, marginal costs $\boldsymbol{c} = [c_1, \ldots, c_n]$, h local steps, T iterations, E epochs, initial parameters $\boldsymbol{w}^1$, loss f, step-size γ, and constants (L, σ^2, α).

Server determines expected optimal data use for each device $m_i^* \leftarrow \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$
Server computes each device's penalty scalar $\lambda_i \leftarrow \frac{m_i^*(\sum \boldsymbol{m})}{(2-\alpha)\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}} \left(c_i - \frac{\gamma\sigma^2 L}{2(\sum \boldsymbol{m}_{-i} + m_i^*)^2}\right)^2$
Server receives free-rider penalty payment $P_{fr}(m_i)$ from devices via Equation 5.4 using λ_i
$s_i \leftarrow m_i / \sum_{j=1}^n m_j$
for $t = 1, \ldots, T$ **do**
    Server distributes $\boldsymbol{w}^t$ to all devices
    for h local steps, each device i **in parallel do**
        $\boldsymbol{w_i}^{t+1} \leftarrow$ Agent Update$(\boldsymbol{w}^t, i, m_i, f, h)$ ▷ DeviceUpdate is detailed in Algorithm 12
    $\boldsymbol{w}^{t+1} \leftarrow \sum_i s_i \boldsymbol{w_i}^{t+1}$
Output: model $\boldsymbol{w}^T$.

Lemma 2 (PFL Assurance of IR at Optimum)**.** *Let c_i be the marginal cost for a device i, $m^* := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}} = m_{i,l}^* = m_{i,PFL}^*$ be the device's locally optimal data usage, and $\alpha \in [0, 2)$ be a server-specified hyper-parameter. In order to ensure that the optimal penalized federated loss is at least lower than the optimal local loss, $\ell_{i,PFL}(m_i^*) \leq \ell_{i,l}(m_i^*)$, one must select λ_i such that,*

$$\lambda_i := \frac{m^* \sum \boldsymbol{m}}{(2-\alpha)\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}} \left(c_i - \frac{\gamma\sigma^2 L}{2(\sum \boldsymbol{m}_{-i} + m^*)^2}\right)^2. \tag{5.6}$$

Selection of such λ_i results in a loss gap between $\ell_{i,PFL}(m^*)$ and $\ell_{i,l}(m^*)$ of,

$$\Delta\ell_i(m^*) := \ell_{i,l}(m^*) - \ell_{i,PFL}(m^*) = \frac{\alpha}{4}\left(\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{m^*(\sum \boldsymbol{m}_{-i} + m^*)}\right). \tag{5.7}$$

Remark 4. *Devices provably receive improved loss $\Delta\ell$ when using their locally optimal data amount $m_{i,l}^*$ participating in PFL (Algorithm 8) versus local training, thereby ensuring IR.*

Algorithm 9 Federated Agent Cost Truthfulness (FACT).
Input: data $\boldsymbol{m} = [m_1, \ldots, m_n]$, marginal costs $\boldsymbol{c} = [c_1, \ldots, c_n]$, h local steps, T iterations, E epochs, initial parameters $\boldsymbol{w}^1$, loss f, step-size γ, and constants (L, σ^2, α).

1: **Output:** model $\boldsymbol{w}^T$ and monetary reward r_i.
2: Server receives contract fees: $\Delta\ell_i(m_i, c_i) \leftarrow \frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{2m_i(m_i + \sum \boldsymbol{m}_{-i})} - P_{fr}(m_i, c_i)$.
3: $\triangleright$ (P_{fr} cancels out in PFL, so $\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{2m_i(m_i + \sum \boldsymbol{m}_{-i})}$ is the *effective* contract fee).
4: Run Penalized Federated Learning: $\boldsymbol{w}^T \leftarrow$ PFL($\boldsymbol{m}, \boldsymbol{c}, h, T, E, \boldsymbol{w}^1, f, \gamma, L, \sigma^2, \alpha$).
5: Compute each device reward r_i via the cost truthfulness game in Equation 5.10.

Remark 5. *The parameters λ_i and α control the amount of benefit received by a device i. A larger value of α, and consequently λ_i, results in a larger gap between $\ell_{i,PFL}(m^*)$ and $\ell_{i,l}(m^*)$. Inversely, a smaller value of α and λ_i results in a smaller gap between local and PFL loss.*

The proofs of Theorem 6 and Lemma 2 are found in Appendix C.2. Now, the formal proof that PFL both (**i**) eliminates device free riding and (**ii**) is individually rational is presented.

Theorem 7 (Elimination of Federated Free-Riding With Truthful Devices)**.** *PFL (Algorithm 8) using λ_i from Lemma 2 is IR and eliminates the free-rider dilemma when devices are truthful.*

Proof. The result of Theorem 6 is that each device i's optimal strategy within the penalized federated scheme is to use their locally optimal amount of data $m_i^* = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$. Furthermore, Lemma 2 states that using m_i^* within the penalized federated scheme results in a reward, or improvement over local training, of $\Delta\ell_i = \frac{\alpha}{4}\left(\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{m_i^*(m_i^* + \sum \boldsymbol{m}_{-i})}\right)$. Thus, by combining Theorem 6 and Lemma 2, truthful devices that choose to participate in PFL attain a reshaped and lower loss at the same optimum. Individually rational devices would therefore prefer to participate in PFL (Algorithm 8) over local training due to the reshaped optimum's lower loss. This optimum achieves the same data usage as local training, thereby eliminating the free-rider dilemma.

Remark 6 (Truthfulness Concerns)**.** *Participating in PFL, rational devices contribute as much as they would locally (Theorem 6) and attain more benefit (Lemma 2). The only remaining issue is truthfulness: devices may lie about their costs in order to contribute less to training under the guise of what looks to be an "optimal amount" (thereby avoiding large P_{fr}).*

While PFL eliminates free riding when devices are truthful, this assumption is unrealistic in practice. Untruthful devices can report inflated costs to the server in order to trick it into expecting smaller data usage. In effect, devices can lie about their costs to free ride. Below, a mechanism is proposed that incentivizes devices to be truthful (*i.e.,* their best strategy is not to lie).

5.3 FACT: Eliminating Free-Riding With Untruthful Devices

When devices are truthful, PFL (Algorithm 8) provably eliminates the free-rider dilemma (Theorem 7). However, an adversarial device i can still free ride by misreporting, or lying, about their true marginal cost c_i. As previously mentioned, the situation of malicious gradient updates coming from adversarial devices, covered by recent works [131, 132, 133, 134], is not considered. Instead, it is assumed that devices share a common goal of reducing loss, and act selfishly rather than maliciously.

As seen in Theorem 5, an adversarial device can lie their way to a smaller data optimum by inflating their cost: $m_i^{lie} = \sqrt{\frac{\gamma\sigma^2 L}{2(c_i+\epsilon)}} < m_{i,l}^*$. In this case, an adversarial device will **(i)** avoid server penalty (since it will be contributing what looks like an optimum from the server's view), **(ii)** incur smaller true costs $c_i m_i^{lie} < c_i m_i$, and **(iii)** still reap the benefits of a larger batch size $\sum \boldsymbol{m}_{-i} + m_i^{lie}$.

Definition 6 (Truthful Mechanism)**.** *A truthful mechanism is one in which no device can reduce its loss by reporting a cost different from its true cost, regardless of other devices' actions. Overall, each device's dominant strategy is to report its true cost.*

To counteract untruthful devices from lying to the server about their costs, the truthful mechanism FACT: Federated Agent Cost Truthfulness (Algorithm 9) is proposed. FACT ensures that each device i's dominant strategy is to report its true cost c_i (thereby satisfying Definition 6).

Assumption 4 (No Collusion)**.** *Devices have no knowledge of other devices' costs or the distribution of device costs f_C. In the absence of information, each device i believes that its cost c_i is equally likely to be larger or smaller than any other device's cost.*

The lone assumption about device knowledge is that devices are unknowledgeable about the distribution of device costs f_C or any other device's cost: device costs are private. Assumption 4 ensures that there is no collusion amongst devices. In the absence of knowledge about other devices' costs, it is reasonable for a device to believe that its private cost is equally likely to be greater or lesser than the cost for any other device (*i.e.,* the median cost). Now that devices can report costs c which differ from their true cost c_i, each device's PFL penalty (Equation 5.4) and loss (Equation 5.5) becomes a multi-variable function,

$$P_{fr}(m_i, c) = \lambda_i\left(\frac{c}{2\lambda_i} - \frac{\gamma\sigma^2 L}{4\lambda_i(\sqrt{\frac{\gamma\sigma^2 L}{2c}} + \sum \boldsymbol{m}_{-i})^2} + \sqrt{\frac{\gamma\sigma^2 L}{2c}} - m_i\right)^2, \tag{5.8}$$

$$\ell_{i,PFL}(m_i, c) = \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i + P_{fr}(m_i, c). \tag{5.9}$$

Regardless of the reported cost c, each device i will incur a true cost $c_i m_i$ locally in Equation 5.9.

Using a Sandwich to Incentivize Truthfulness. The FACT mechanism is the first to ensure

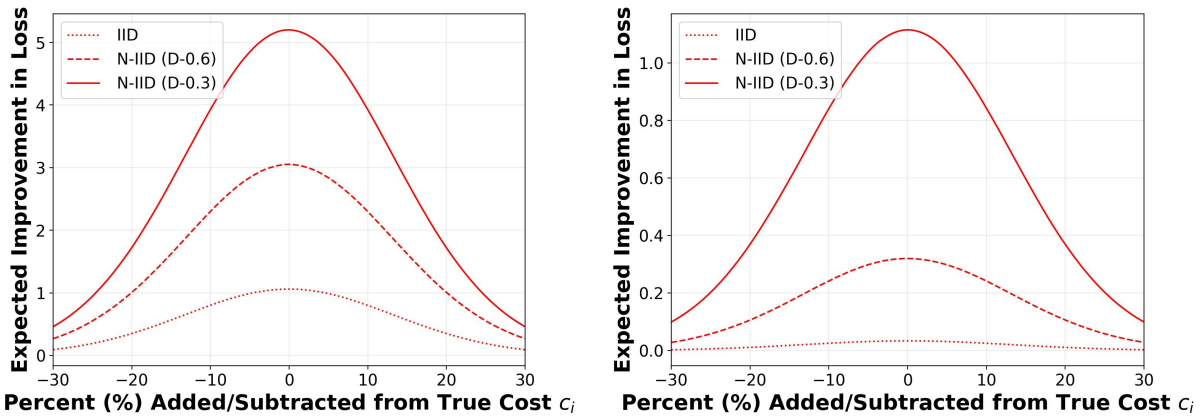


Figure 5.2: **Enforcement of Device Truthfulness.** Average net improvement in loss over local training is plotted for iid (dotted line) and two non-iid device distributions (D-0.6: dashed, D-0.3: solid). For both CIFAR10 (left) and MNIST (right), devices maximize their net improvement in loss when truthful (0% added) about their true cost. This matches the results in Theorem 8.

truthfulness while solving the free-rider dilemma in federated learning. FACT ensures device truthfulness by fostering competition amongst devices. The competition begins at the end of federated training by randomly grouping reported device costs into threes. The rules of this competition are simple: each device i "wins" the competition, and receives a reward, if its cost c_i is sandwiched between the two other device costs in its group. If there is a tie, the server randomly selects one of the tied devices as the winner. This mechanism is detailed mathematically below,

$$P_{ct}(m_i, c) := \begin{cases} \Delta\ell_i(m_i, c) - \frac{3}{n} \sum_{j \neq i \in [n]} \Delta\ell_j & \text{if } C_a < c < C_b, \\ \Delta\ell_i(m_i, c) & \text{else.} \end{cases} \quad C_a, C_b \sim f_C \text{ randomly sampled costs} \tag{5.10}$$

Lemma 3. *The probability of a device "winning" in the truthfulness mechanism in Equation (5.10), given a reported cost c, is $\upsilon(c) := 2F_C(c)(1 - F_C(c))$, where F_C is the CDF of f_C.*

Akin to Equation (5.7), multi-variate $\Delta\ell_i$ is defined as $\Delta\ell_i(m_i, c) := \ell_{i,l}(m_i, c) - \ell_{i,PFL}(m_i, c)$. Since $\Delta\ell_i$ is computed by the server, it uses c instead of c_i for the local loss $\ell_{i,l}(m_i, c)$. The crux of the truthfulness competition is that each device i *never receives its own reward* $\Delta\ell_i$. Instead, as shown in Equation (5.10), each device pays $\Delta\ell_i$ to the server before training regardless of the mechanism outcome. After training, if the device "wins" the competition, by having a cost sandwiched between C_a and C_b, then it receives triple the average of all other device rewards $\frac{3}{n}\sum_{j\neq i\in[n]}\Delta\ell_j$. The average reward is tripled, since two other devices will lose the competition and receive no reward. By making the reward decoupled from a device's own reward $\Delta\ell_i$, each device can no longer increase or decrease its reward by altering how much data m_i they use for federated training. Now, they can only affect the likelihood of winning the competition by choosing what cost c_i to report to the server.

Incorporating the truthfulness mechanism into the PFL loss function results in the FACT loss function,

$$\ell_{i,Fact}(m_i, c) = \underbrace{\frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i + P_{fr}(m_i, c)}_{\ell_{i,PFL}(m_i,c)} + P_{ct}(m_i, c). \tag{5.11}$$

Theorem 8 (Main Theorem)**.** *Each device i's best strategy participating in Fact (Equation 5.11) is to report its true cost and use its locally optimal amount of data $(m_i, c)_i^* = (\sqrt{\frac{\gamma\sigma^2 L}{2c_i}}, c_i)$.*

Corollary 3 (Individually Rational)**.** *When using their best strategy, devices participating in* FACT *will never receive worse loss than training alone, $\ell_{i,Fact}(m_i^*, c_i^*) \leq \ell_{i,l}(m_{i,l}^*)$.*

Remark 7 (Truthfulness)**.** FACT *is a truthful mechanism: each device's best strategy is to report its true cost* c_i*. Devices cannot reduce their loss by reporting a different cost (satisfying Definition 6).*

Remark 8 (Elimination of Free Riding)**.** FACT *eliminates device free riding: each device's best strategy is to use as much data as is locally optimal* $m_i^* = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$ *(Theorem 4). Thus, the improved optimum for PFL (Theorem 6) is maintained even under device untruthfulness.*

Remark 9 (Individually Rational)**.** *When using their best strategy, agents participating in* FACT *either (i) receive loss equivalent to local training if they lose the competition or (ii) reduced loss if they win (Equation 5.10). In fact, agents receive lower loss in expectation.*

5.4 Experiments

Below, the effectiveness of FACT is analyzed at **(1)** enforcing device truthfulness, **(2)** eliminating free riding, and **(3)** reducing device loss compared to local training. True federated experiments (no simulations) are performed using the CIFAR-10 [71] and MNIST [157] datasets to train an image classification model.

5.4.1 Free Rider Elimination With Untruthful Devices

Experimental Setup. Within the following experiments, 16 devices train a model individually (locally) as well as in a federated manner. Each device uses 3,125 and 3,750 data samples each for CIFAR-10 and MNIST respectively. FACT is analyzed under homogeneous and heterogeneous device data distributions. For heterogeneous device data distributions, Dirichlet

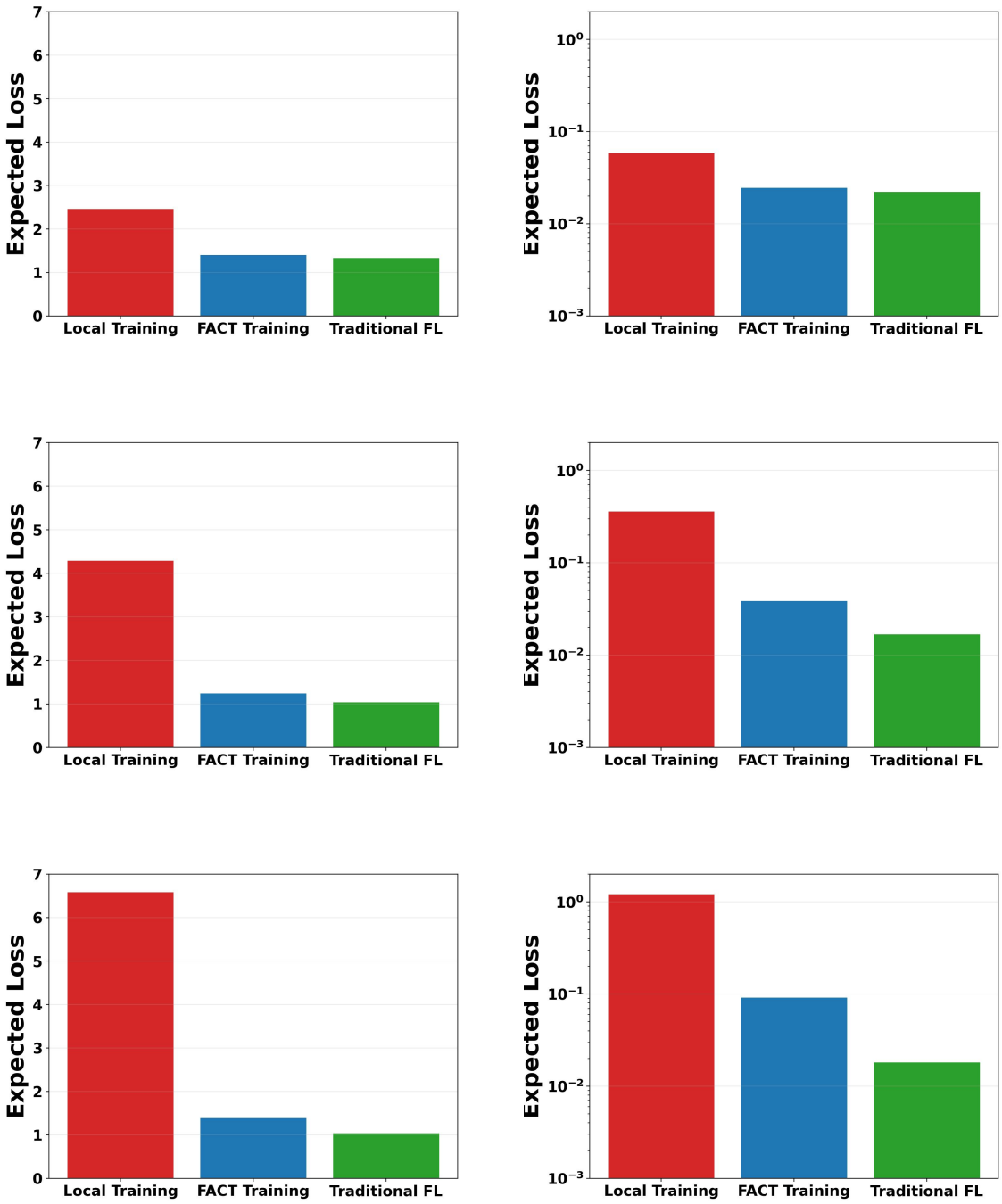


Figure 5.3: **Reduction in Device Loss.** The average device loss for baselines on CIFAR10 (top row) and MNIST (bottom row) under IID (left), and two non-IID data distributions (center: D-0.6, right: D-0.3). Traditional federated learning (FL) is an upper bound on device loss (if devices did not free ride). FACT improves device loss over local training by up to a factor of 3x for CIFAR10 and 4x for MNIST.

distributions with parameters $\alpha = 0.3, 0.6$ are used to determine the label ratio for each device [158]. The sandwich-truthfulness competition defined in Equation (5.10) is simulated by randomly sampling 2,000 synthetic device costs from a Gaussian distribution with a mean centered at each device's true cost (standard deviation of one-tenth of the true cost). One hundred thousand simulation trials are performed. Then, the mean performance for each device over all trials is computed. Further experiment details and hyperparameters are found in Appendix C.1.

Lack of Baselines. FACT is the first truthful horizontal federated learning method eliminating free riding while simultaneously enforcing truthfulness without the use of an auction or bandit (thereby all willing devices can participate). As such, the results of FACT are compared with the only available option: local training. It is showcased below that FACT reduces device loss compared to local training while simultaneously avoiding free-riding and truthfulness issues.

FACT Enforces Truthfulness. The results of Figure 5.4 back the theoretical result of Theorem 8, which states that each device maximizes its expected reward (*i.e.,* net reduction in loss versus local training) when truthfully reporting its cost. Across both homogeneous (IID) and heterogeneous (non-IID) device data distributions, devices' net improvement in loss peaks when reporting their true cost (0% added) and monotonically decreases as devices report an inflated or deflated true cost. Thus, the sandwich-style truthfulness competition, which FACT employs, disincentivizes devices from inflating or deflating their true costs. This is important, as devices are no longer incentivized to lie to the server about their true cost in order to free-ride without the free-rider penalty in Equation (5.4) being imposed. Net improvement in loss grows as data distributions become increasingly non-IID. Federated learning is robust enough to deal with the increasingly non-IID datasets while local training suffers (Figures C.2 & C.3). Therefore, the gap between federated learning and local training grows, and thus so too does the reward in FACT.

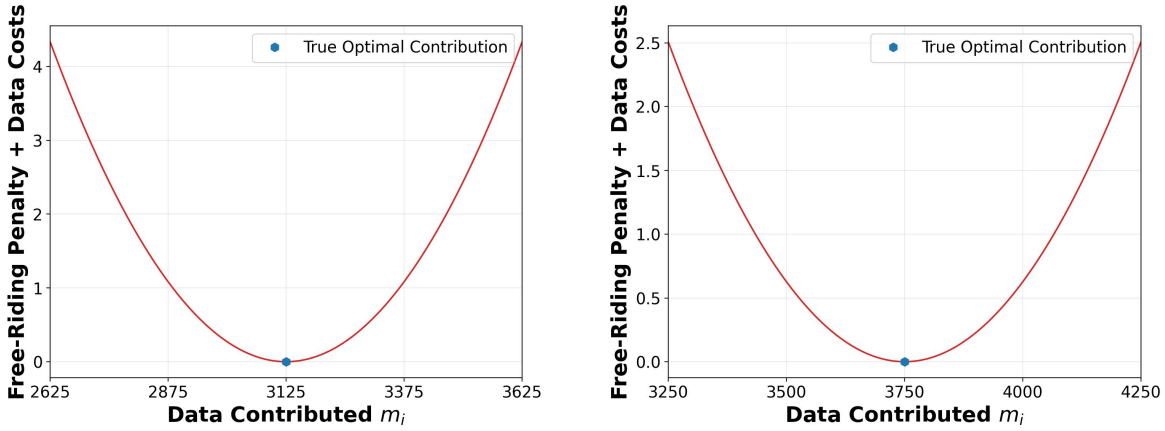


Figure 5.4: **Elimination of Free Riding via Penalty.** The penalty term $P_{fr}(m_i)$ plus data collection costs $c_i m_i$ is plotted for CIFAR10 (left) and MNIST (right) for varying data contributions m_i. These combined costs are minimized at the local optimum $m^*_{i,l}$, as predicted by Theorem 6.

FACT Reduces Device Loss. Devices participating in FACT on average achieve a reduction in total loss compared to training on their own when using the same amount of data. Consequently, participating in FACT is Individually Rational: a device at worst receives loss equivalent to local training, and on average receives improved loss (Figure 5.3). Traditional federated learning achieves a lower average device loss than FACT only when *all devices use as much data as they would locally*. However, as shown in Theorem 5, this is unrealistic. Traditional federated learning emits a catastrophic free-riding optimum. The results of traditional federated learning in Figure 5.3 is an upper bound on achievable loss, only attainable if all devices refuse to free ride.

Thus, the gap in loss between FACT and the upper bound of traditional federated learning is the cost of device untruthfulness and their inclination to free ride. The gap is the direct result of the penalty terms on free riding P_{fr} and truthfulness P_{ct} found in Equation (5.11). Average local loss increases as data distributions become increasingly non-IID, while traditional federated learning remains robust to the distribution shift (Figures C.2 & C.3). Since the devices in FACT who lose the truthfulness competition receive a loss equivalent to local training, the loss of FACT

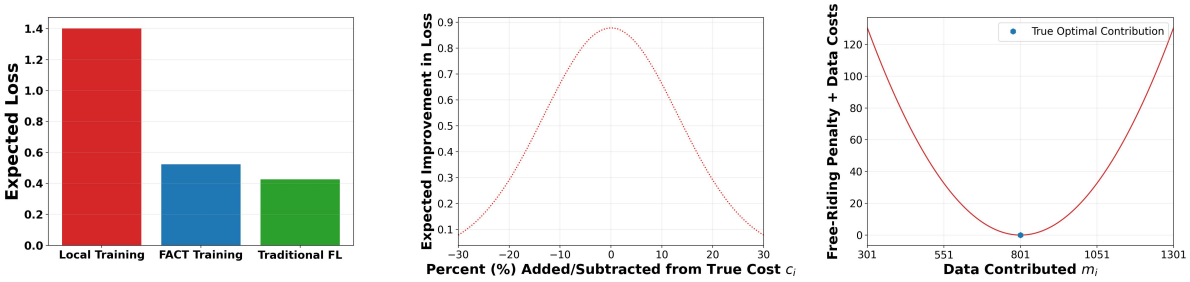


Figure 5.5: **FACT Eliminates Free Riding in Realistic Settings**. When training an image classifier for diagnosing skin cancer, devices participating in FACT achieve much lower loss (66% less) than if they did not participate (left). Devices maximize their improvement in loss over local training when they are truthful; reporting inflated or deflated costs diminishes improvement in loss (middle). Devices minimize penalties when using their locally optimal amount of data ($m^* = 801$) for training (right).

also rises in proportion to the local training loss.

FACT Eliminates Free Riding. Within the experiments for CIFAR-10 and MNIST, a marginal cost was carefully selected such that it is locally optimal for each device to use 3,125 and 3,750 data samples for training respectively. As shown in Figure 5.4, the free-rider penalty P_{fr} harshly penalizes devices for deviating from this locally-optimal amount. This result accounts for the reduction (gain) in data-collection costs for using less (more) data during training. Thus, as proven in Theorem 6, it is indeed suboptimal for a device i to use either more or less data than is what is locally optimal (for a given reported cost c_i) when participating in FACT.

5.4.2 Real-World Case Study: Inter-Hospital Skin Cancer Diagnosis

Finally, the performance of FACT is tested in a realistic situation, where a hypothetical consortium of hospitals collaboratively train a model, privately in a federated manner, that can diagnose skin cancer. One of the hospitals is smaller and resource-constrained. It is difficult, but not impossible, for this hospital to collect more data for training. Thus, in the absence of a truthful federated learning mechanism, the smaller hospital could over-report its collection costs to the

server in order to contribute little to no data towards training while still reaping the unjustified rewards of a well-trained global model.

To test how FACT deals with this scenario, an image-classification model is trained on the HAM10000 [159] dataset. HAM10000 consists of a collection of dermatoscopic images of pigmented lesions. The goal is to train a model which can perform automated diagnosis of pigmented skin lesions, including melanoma (skin cancer). This setup is similar to the experimental setup detailed in Chapter 5.4.1. One difference is that a ResNet50 model, pre-trained on ImageNet [160], is fine-tuned on HAM10000 (a realistic approach for a hospital). HAM10000 is an imbalanced dataset, and evenly partitioning 80% of the data amongst 10 devices, as their local training sets, further exacerbates the non-IID-ness of the data. This too is realistic, as data is often non-IID amongst federated devices. An Adam optimizer with a learning rate of 1e-3 and batch size of 128 is used.

As expected, FACT provably dissuades any hospital from lying about its cost in order to reduce its data-contribution level. Devices are incentivized to be truthful: reduction in loss is maximized when devices truthfully report their costs (see the middle plot in Figure 5.5). As a result, a smaller, resourced-constrained hospital would still contribute its locally optimal amount of data, or face penalties (see the right plot in Figure 5.5). Finally, all participating hospitals are still better off participating in FACT than by training alone. Participating-device loss is reduced by nearly 66% compared to local training (see the left plot in Figure 5.5). The effectiveness of FACT to dissuade free-riding in the real-world setting of training a skin cancer classifier across hospitals indicates that it will be successful in other real-world settings.

Chapter 6: Towards Realistic Mechanisms That Incentivize Federated Participation and Contribution[1]

Like Chapter 5, this chapter focuses on the issue of free riding within federated learning. Generally, it is generally assumed within federated learning literature that edge devices will (i) always participate in training and (ii) use all of its local data during training. However, it is irrational for devices to participate in, and incur the costs of, federated training without receiving proper benefits back from the server (via model performance or monetary rewards). Specifically, there are two major challenges:

(C1) Lack of Participation Incentives. Current federated learning frameworks generally lack incentives to increase device participation. This leads to training with fewer devices and data to compute local updates, which can potentially reduce model accuracy. Incentivizing devices to participate in training and produce more data, especially from a server's perspective, improves model performance (further detailed in Chapter 6.1.1), which leads to greater utility for both the devices and server.

(C2) Lack of Contribution Incentives: The Free-Rider Dilemma. In realistic settings, each device determines its own optimal amount of data usage for federated contributions. As such, many federated-learning frameworks run the risk of encountering the free-rider problem:

[1] A version of Chapter 6 has been published in NeurIPS 2023 Federated Learning Workshop [141]

devices do not contribute gradient updates, yet they reap the benefits of a well-trained collaborative model. Removing the free-rider effect, in federated learning frameworks, is critical because it improves the performance of trained models [161, 162] and reduces security risks [129, 130, 163] for devices.

To address these challenges, a handful of recent federated-learning literature [4, 146, 150, 164] consider device *utility*: the net benefit received by a device for participating in federated training. Every rational device i aims to maximize its utility u_i. Consequently, utility is the guiding factor in whether rational devices participate in federated training. Devices will only participate, fully, if the utility u_i^r gained, via its rewards (a_i^r, R_i), outstrips the maximum utility gained from local training u_i^o. In a step towards more realistic federated learning, the referenced works [4, 146, 150, 164] design mechanisms, or systems, to maximize device utility and provide greater utility than local training, when more data is contributed.

While previous federated-learning literature incorporates utility, they require *unrealistic assumptions* such as disallowing **(i)** devices having utility that depends non-linearly on their model accuracy, **(ii)** heterogeneous data distributions across devices, **(iii)** truly federated (non-data sharing) methods, and **(iv)** modeling of central server utility. In this chapter, a leap towards realistic federated systems is taken by relaxing all four assumptions above while simultaneously solving the key issues in **C1** and **C2**.

Summary of Contributions. Within this chapter, REALFM is proposed: a federated mechanism $\mathcal{M}$ (*i.e.,* system) that a server, in a federated-learning setup, implements to eliminate **(C1)** and **(C2)** when rational devices participate. REALFM is *Individually Rational* (IR): participating devices and the server provably receive greater utility than training alone $u_i^r \geq u_i^o$. The goal of REALFM is to *design a reward protocol, with model-accuracy* a^r *and monetary* R

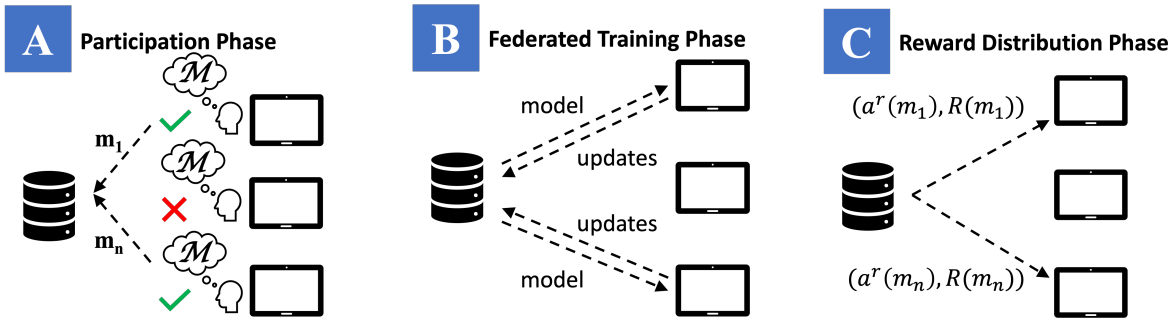


Figure 6.1: **Federated Mechanism Diagram. (A) Decision Phase for Device Participation**. Each device decides whether it wants to participate in the mechanism. If so, the quantity of data points used by each agent m_i is sent to the server (no data is shared). Note that rational devices would participate in REALFM; the utility gained by participating is *never* less than what agents attain locally. **(B) Federated Training Phase**. Devices upload their updates and receive feedback from the server in an iterative manner. **(C) Accuracy & Monetary Reward Distribution Phase.** Upon completion of federated training, the server distributes both accuracy $a^r(m_i)$ and monetary $R(m_i)$ rewards to device i. These rewards, the crux of REALFM, incentivize device participation and data contribution.

rewards, such that rational devices choose to participate and contribute more data. By increasing device participation and data contribution, a server trains a higher-performing model and subsequently attains greater utility. REALFM is a mechanism that,

▷ Provably eliminates the free-rider dilemma **(C1)** by incentivizing devices to participate and use more data during the federated training process **(C1)** than they would on their own.

▷ Allows more *realistic settings*, including: **(1)** a non-linear relationship between accuracy and utility, **(2)** data heterogeneity, **(3)** no data sharing, and **(4)** modeling of central server utility.

▷ Produces state-of-the-art results towards improving utility (for both the server and devices), data contribution, and final model accuracy on real-world datasets.

6.1 Preliminaries

6.1.1 Problem Formulation

As covered extensively, within the federated-learning setting, n devices collaboratively train the parameters $\boldsymbol{w}$ of an AI model. Devices compute local gradient updates on $\boldsymbol{w}$ using their own local data, with the server aggregating all local device updates to perform a single global update. Each device i has its own local dataset $\mathcal{D}_i$ (able to change in size and distribution) which can be heterogeneous across devices. The amount of data per device i is defined as $m_i := |\mathcal{D}_i|$ and denote $\boldsymbol{m} := \{m_1, \ldots, m_n\}$. Dataset sizes are constrained by cost: each device i has its own fixed marginal cost $c_i > 0$ per sample, which represents the cost of collecting and computing the gradient of an extra data point (*e.g.,* collecting m samples incurs a cost of $c_i m$). Linear costs are considered, as data-collection and sampling costs are generally constant over time in the cross-device setting (*e.g.,* powering an IoT sensor incurs a constant cost on average) [142, 143, 144]. Overall, each device i determines the dataset size m_i that best balances data costs $c_i m_i$ with model performance.

Mechanisms. To entice devices to participate in federated training, a central server must incentivize them. Two realistic rewards that a central server can provide are: **(i)** model accuracy and **(ii)** monetary. The interaction between the server and devices is formalized as a *mechanism* $\mathcal{M}$. When participating in the mechanism, a device i performs federated updates on a global model $\boldsymbol{w}$, using m_i local data points, in exchange for model accuracy $a_i^r \in \mathbb{R}_{\geq 0}$ and monetary $R_i \in \mathbb{R}_{\geq 0}$ rewards.

$$\mathcal{M}(m_1, \cdots, m_n) = ((a_1^r, R_1), \cdots, (a_n^r, R_n)) \,. \tag{6.1}$$

A mechanism that provides rewards in proportion to the amount of contributions m_i made by each device i is desirable. Without proportionality, federated-learning methods fall prey to the free-rider dilemma, where devices are able to reap rewards without proper contribution.

Definition 7 (Feasible Mechanism)**.** *A feasible mechanism $\mathcal{M}$ (1) returns a non-negative reward and accuracy for each device, and (2) is bounded in its provided utility.*

Definition 8 (Individual Rationality (IR))**.** *A mechanism $\mathcal{M}$ is IR if devices always receive better utility by participating than it can training by itself.*

Rational devices are willing to participate in a server's mechanism $\mathcal{M}$ if they can receive (i) realistic rewards (**feasible**) and (ii) greater utility than they can get by training alone (**IR**). Finally, it must be proven that mechanisms fulfilling such qualities can reach a stable equilibrium of device contributions.

Theorem 9 (Existence of Pure Equilibrium)**.** *Consider a feasible mechanism $\mathcal{M}$ providing utility $[\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i$ to device i. Devices receive no utility if no data is contributed, that is, $[\mathcal{M}^U(0; \boldsymbol{m}_{-i})]_i = 0$. Define the utility of a participating device i as,*

$$u_i^r(m_i; \boldsymbol{m}_{-i}) := [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i - c_i m_i. \tag{6.2}$$

If $u_i^r(m_i, \boldsymbol{m}_{-i})$, is quasi-concave for $m_i \geq m_i^u := \inf\{m_i |\ [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i > 0\}$ and continuous in $\boldsymbol{m}_{-i}$, then a pure Nash equilibrium with $\boldsymbol{m^{eq}}$ data contributions exists such that,

$$u_i^r(\boldsymbol{m^{eq}}) = [\mathcal{M}^U(\boldsymbol{m^{eq}})]_i - c_i \boldsymbol{m_i^{eq}} \geq [\mathcal{M}^U(m_i; \boldsymbol{m_{-i}^{eq}})]_i - c_i m_i \ \forall m_i \geq 0. \tag{6.3}$$

The proof of Theorem 9 is found in Appendix D.4. This proof amends and simplifies the one within Karimireddy et al. [4], as it is now shown that u_i^r must only be quasi-concave in m_i.

Remark 10. *Under only mild assumptions on the utility provided by mechanism $\mathcal{M}$ (feasibility, quasi-concavity, & continuity w.r.t data m), participating devices reach an equilibrium on local data usage for federated training $\boldsymbol{m^{eq}}$. Deviating from such equilibrium contribution $\boldsymbol{m_i^{eq}}$ results in a decrease in utility for device i (Equation 6.3).*

In order for a mechanism $\mathcal{M}$ to provide data and output utility, via model-accuracy rewards (Equation 6.1), one must define a relationship between accuracy and data.

Accuracy-Data Relationship. Data reigns supreme when it comes to model performance. Model accuracy improves as the quantity of training data increases, premised on consistent data quality [165, 166, 167]. Empirically, one finds that accuracy of a model is both concave and non-decreasing with respect to the amount of data used to train it [168]. Training of an AI model generally adheres to the law of diminishing returns: improving model performance by training on more data is increasingly fruitless once the amount of training data is already large.

Assumption 5. *Accuracy function $\hat{a}_i(m)$ is continuous, non-decreasing, and concave with respect to the data m. Bounded accuracy $a_i(m) := \max\{\hat{a}_i(m), 0\}$ has a root at 0.*

For a learning task, each device i has an optimal attainable accuracy $a_{opt}^i := a_{opt}(\mathcal{D}_i) \in [0, 1)$ given its data distribution $\mathcal{D}_i$. Assumption 5 captures the empirical relationship between AI model accuracy $\hat{a}_i(m) : \mathbb{Z}^+ \to (-\infty, a_{opt}^i)$ and data m in the wild: (**continuous & non-decreasing**) accuracy never decreases with more data and (**concavity**) accuracy experiences diminishing returns with more data. Since negative accuracy is impossible, it is defined as $a_i(m) := \max\{\hat{a}_i(m), 0\}$. Contrary to previous work [4], accuracy $a_i(m)$ differs per device.

Remark 11 (Attainability)**.** *An accuracy function $\hat{a}_i(m)$ which satisfies Assumption 5 is:*

$$\hat{a}_i(m) := a^i_{opt} - \frac{\sqrt{2k(2+\log(m/k))}+4}{\sqrt{m}}. \tag{6.4}$$

The theory allows general $\hat{a}_i(m)$ as long as Assumption 5 is satisfied. However, like [4], $\hat{a}_i(m)$ (as defined in Equation 6.4) is used for experiments. $k > 0$ denotes the hypothesis class complexity. Equation 6.4 is rooted in a generalization bound found in Appendix D.3.

Remark 12 (Server Accuracy)**.** *The central server C has its own accuracy function $\hat{a}_C(m)$ with optimal attainable accuracy $\bar{a}_{opt} := \mathbb{E}_{i\in[n]}\left[a^i_{opt}\right] = \sum_{i=1}^{n} \frac{m_i}{\sum_j m_j} a^i_{opt}$,*

$$\hat{a}_C(m) := \bar{a}_{opt} - \frac{\sqrt{2k(2+\log(m/k))}+4}{\sqrt{m}}. \tag{6.5}$$

Remark 13 (Heterogeneous Distributions)**.** *Instead of assuming that each device's local data is independently chosen from a common distribution (known as the IID setting), here it is generalized to the heterogeneous and non-IID setting. This is a major novelty, as previous mechanisms, like Karimireddy et al. [4], focus on identical device data distributions.*

6.1.2 Related Works

Federated Mechanisms. Previous literature [146, 150, 164, 169] has proposed mechanisms to solve **(C1)** and incentivize devices to participate in federated training. However, these works fail to address **(C2)**, the free-rider problem, and have unrealistic device utilities. In Chen et al. [169], data sharing is allowed, which is prohibited in the federated learning setting due

to privacy concerns. In Zhan et al. [146, 150, 164], device utility incorporates a predetermined reward for participation in federated training without specifying how this amount is set by the server. This could be unrealistic, since rewards should be dynamic and depend upon the success (resulting model accuracy) of the federated training. Setting too low of a reward impedes device participation, while too high of a reward reduces the utility gained by the central server (and risks negative utility if performance lags total reward paid out). Overall, predicting an optimal reward *prior* to training is unrealistic. In contrast, the proposed REALFM introduces a principled mechanism to set the rewards.

The recent work by Karimireddy et al. [4] is the first to simultaneously solve **(C1)** and **(C2)**. They propose a mechanism that incentivize devices to (i) participate in training and (ii) produce more data than on their own (data maximization). By incentivizing devices to maximize production of local data, the free-rider effect is eliminated. While this proposed mechanism is a great step forward for realistic mechanisms, impediments remain. First, Karimireddy et al. [4] requires data sharing between devices and the central server. This is acceptable if portions of local data are shareable (*i.e.* no privacy risks exist for certain subsets of local data), yet it violates the key tenet of federated learning: privacy. Second, device utility is designed in Karimireddy et al. [4] such that the utility improves linearly with increasing accuracy. This is unrealistic, as devices likely find greater utility for an increase in accuracy from 98% to 99% rather than from 48% to 49%. Finally, Karimireddy et al. [4] assumes all local data comes from the same distribution, which is unrealistic. The proposed REALFM mechanism addresses all issues above.

The works Tu et al. [145], Zhan et al. [146], Zeng et al. [170], Ali et al. [171] survey the different methods of incentives present in federated learning literature. The goal of the presented methods are solely to increase device participation within federated learning frameworks. The

issues of free riding and increased data or gradient contribution are ignored. Sim et al. [139], Xu et al. [172] design model rewards to meet fairness or accuracy objectives. In these works, as detailed below, each device receives models proportionate to the amount of data each contributes. However, each device is not incentivized to contribute more data. This chapter seeks to incentivize each device to contribute more during training.

Contract Theory and Federated Free Riding. Contract theory in federated learning aims to optimally determine the balance between device rewards and registration fees (cost of participation). In contract mechanisms, devices may sign a contract from the server specifying a task, reward, and registration fee. If agreed upon, the device signs and pays the registration fee. Each device receives the reward if it completes the task and receives nothing if it fails. Contract mechanisms have the ability to punish free riding in federated learning by creating negative incentive if a device does not perform a prescribed task (*i.e.,* it will lose its registration fee). The works Kang et al. [38], Cong et al. [135], Lim et al. [136], Liu et al. [137], Lim et al. [173], Wang et al. [174] propose such contract-based federated learning frameworks. While effective at improving model generalization accuracy and utility, Lim et al. [136], Liu et al. [137] focus on optimal-reward design. The REALFM mechanism does not require registration fees, thus boosting device participation, and utilizes an accuracy-shaping method to provide rewards at the end of training in an optimal and more realistic approach. Furthermore, like Karimireddy et al. [4], this mechanism incentivizes increased contributions to federated training, which is novel compared with the existing contract-theory literature and mechanisms.

Collaborative Fairness and Federated Shapley Value. Collaborative fairness in federated learning [138, 139, 140, 175] is closely related to this chapter. The works Lyu et al. [138], Sim et al. [139], Xu and Lyu [140], Lyu et al. [175] seek to fairly allocate models with

varying performance depending upon how much devices contribute to training in federated learning settings. This is accomplished by determining a "reputation" (a measure of device contributions) for each device, using a hyperbolic sine function, to enforce devices converge to different models relative to their amount of contributions during federated learning training. Thus, devices who contribute more receive a higher-performing model than those who do not contribute much.

There are key differences between this line of work and REALFM, namely: **(1)** REALFM incentivizes devices to both participate in training and increase their amount of contributions. There are no such incentives in collaborative fairness. **(2)** Device and server utility is modeled. Unlike collaborative fairness, it is not assumed that devices will always participate in federated learning training. **(3)** The mechanism design in this chapter *provably* eliminates the free-rider phenomena unlike collaborative fairness, since devices who try to free-ride receive the same model performance as they would on their own (Equations 6.9 and 6.10). **(4)** No monetary rewards are received by devices in current collaborative fairness methods.

Federated Shapley Value (FSV), first proposed in Wang et al. [176], allows for estimation of the Shapley Value in a federated-learning setting. This is crucial to appraise the data coming from each device and possibly pave the way for rewarding devices with important data during the training process. While this chapter allows devices to have heterogeneous data distributions, data valuation is not performed to further fine-tune rewards for each device (*i.e.,* provide more accurate models or more monetary rewards to devices who provide more valuable data). The overarching goal of this chapter is to incentivize and increase device participation, through mechanism design, within federated learning. However, further reward fine-tuning via FSV remains the subject of future research.

6.2 Modeling Realistic Utility

Utility powers the performance of a mechanism. If utility is modeled incorrectly, *e.g.,* unrealistically, a mechanism will guide participants towards unrealistic and suboptimal results.

A More Generalized Accuracy Payoff. Unlike previous federated mechanisms, such as [4, 146, 164], a non-linear accuracy payoff compositional function is introduced $\phi_i(a(m)) : [0, 1) \to \mathbb{R}_{\geq 0}$, which allows for a flexible definition of the utility device i receives from having a model with accuracy $a(m)$. In Karimireddy et al. [4], it is assumed that the outer function ϕ_i is linear, $\phi_i(a(m)) = a(m)$, for all devices, which is restrictive. For example, accuracy improvement from 48% to 49% should be rewarded differently than from 98% to 99%. Therefore, here the outer function ϕ_i is generalized to be a convex and increasing function (which includes the linear case). These requirements (summarized in Assumption 6) ensure that increasing accuracy leads to enhanced utility for rational devices.

Assumption 6. *$\phi_i(a_i(m)) : [0, a^i_{opt}) \to \mathbb{R}_{\geq 0}$ is continuous and non-decreasing for each device i. The outer function $\phi_i(a)$ is convex and strictly increasing w.r.t a ($\phi_i(0) = 0$). The compositional function $\phi_i(a_i(m))$ remains concave and strictly increasing w.r.t m (for all m such that $\hat{a}_i(m) \geq 0$).*

Many **realistic choices** for ϕ_i which satisfy Assumption 6 exist. For $a_i(m)$ as in Equation 6.4, one reasonable choice is $\phi_i(a) = \frac{1}{(1-a)^2} - 1$. This choice captures how utility increasingly grows as accuracy approaches 100%, especially compared to the linear relationship $\phi_i(a) = a$. After defining the accuracy-utility relationship, the server and device utilities can be defined.

Defining Server Utility. *The overarching goal for a central server C is to attain a high-performing AI model from federated training.* As discussed at the beginning of Chapter 6.1.1,

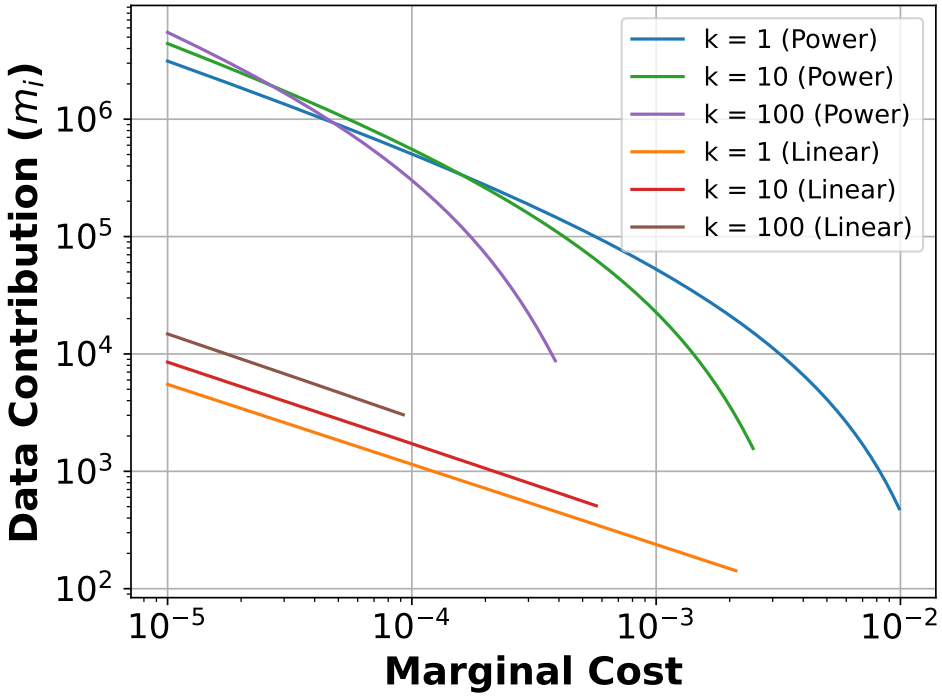
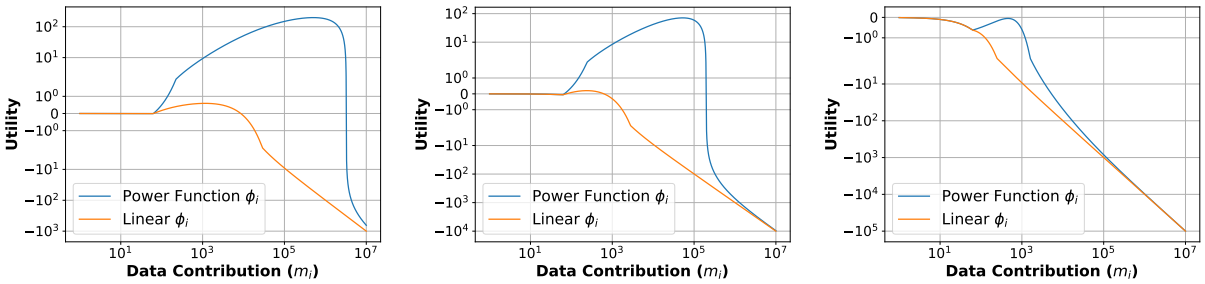


Figure 6.2: **Local Optimal Data Contribution for Varying Payoff Functions.** The optimal data contribution across different payoff functions are compared. Realistic power payoff functions, $\phi_i(a) = \frac{1}{(1-a)^2} - 1$, result in greater optimal contribution compared to linear payoff functions, $\phi_i(a) = a$. The accuracy function $\hat{a}_i(m)$ used is defined in Equation 6.4, with $a^i_{opt} = 0.95$ and multiple k values.



(a) Marginal Cost $c_i = 1\text{e}{-4}$. (b) Marginal Cost $c_i = 1\text{e}{-3}$. (c) Marginal Cost $c_i = 1\text{e}{-2}$.

Figure 6.3: **Utility Functions for Varying Cost and Payoff Functions.** Using both linear, $\phi_i(a) = a$, and power, $\phi_i(a) = \frac{1}{(1-a)^2} - 1$, payoff functions, the change of device utilities with rising costs is analyzed. Once marginal costs c_i become too high, the utility is always negative and devices will not collect data for training. The accuracy function $\hat{a}_i(m)$ used is defined in Equation 6.4, with $a^i_{opt} = 0.95$ and $k = 1$.

AI model accuracy generally improves as the total amount of data contributions $\boldsymbol{m}$ increase. Therefore, server utility u_C is a function of model accuracy a_C (Equation 6.5) which in turn is a function of data contributions,

$$u_C(\boldsymbol{m}) := p_m \cdot \phi_C\left(a_C\left(\sum \boldsymbol{m}\right)\right). \tag{6.6}$$

The fixed parameter $p_m \in (0, 1]$ denotes the central server's profit margin (percentage of utility kept by the central server). Since p_m is fixed, server utility in Equation 6.6 is maximized when $a_C \to 100\%$. However, server accuracy is upper bounded by the optimal attainable accuracy $\bar{a}_{opt} \in [0, 1)$. Thus, to maximize Equation 6.6, the server wants to push $a_C \to \bar{a}_{opt} \approx 100\%$. Accomplishing this requires closing the gap between both (**i**) a_C and $\bar{a}_{opt}$ as well as (**ii**) $\bar{a}_{opt}$ and 100%. The gap in (**i**) can be reduced by increasing the total amount of contributions $\sum \boldsymbol{m} \to \infty$ (via Equation 6.5). The gap in (**ii**) can shrink by receiving more contributions from devices i with high optimal attainable accuracies a^i_{opt} (Remark 12). Thus, an optimal mechanism $\mathcal{M}$ from the server's viewpoint *would be one which incentivizes more data used for federated contributions,*

with a larger proportion coming from devices i with greater a^i_{opt}. Finally, it is important to note that only p_m of the total collected utility is kept by the server in Equation 6.6. The value of p_m is posted by the central server, and thus known by all devices, during the participation phase (Figure 6.1). In Chapter 6.3, it is detailed how the mechanism distributes the remaining $(1 - p_m)$ utility as a reward in proportion to how much data each participating device contributes.

Defining Local Device Utility. The utility u_i for each device i is a function of data contribution: devices determine how many data points m to collect in order to balance the benefit of model accuracy $\phi_i(a(m))$ versus the costs of data collection $-c_i m$. Thus, device utility is mathematically defined as,

$$u_i(m) = \phi_i(a_i(m)) - c_i m. \tag{6.7}$$

Theorem 10 (Optimal Local Data Collection)**.** *Consider device i with marginal cost c_i, accuracy function $\hat{a}_i(m)$ satisfying Assumption 5, and accuracy payoff ϕ_i conforming to Assumption 6. Then the optimal amount of data m^o_i device i should collect is*

$$m^o_i = \begin{cases} 0 \quad \textit{if } \max_{m_i \geq 0} u_i(m_i) \leq 0 \quad \textit{else,} \\ \\ m^*, \textit{ s.t. } \phi_i'(\hat{a}_i(m^*)) \cdot \hat{a}_i'(m^*) = c_i. \end{cases} \tag{6.8}$$

Theorem 10 details how much data a device collects when *training on its own and thereby not participating in a federated training scheme*. Utility curves are plotted for both non-linear and linear accuracy payoff functions, with varying marginal costs c_i, in Figure 6.3. Figure 6.3c shows how utility peaks at a negative value when c_i becomes too large. In this case, as shown in Theorem 10, devices do not contribute data. The proof of Theorem 10 is found within Appendix D.4.

6.3 REALFM: A Step Towards Realistic Federated Mechanisms

Again, the goal of the proposed REALFM mechanism $\mathcal{M}_R$ (Algorithm 10) is to design a reward protocol, with model-accuracy a^r and monetary R rewards, such that rational devices choose to participate and contribute more data than is locally optimal (Theorem 4) in exchange for improved device utility. Let the *utility* REALFM provides to device i be $\mathcal{M}^U_{R,i} := [\mathcal{M}^U_R(\boldsymbol{m})]_i$.

$$\mathcal{M}_R(m_i; \boldsymbol{m}_{-i}) := \begin{cases} \big(a_i(m_i),\ 0\big) & \text{if } m_i \leq m_i^o, \\ \big(a_i\big(m_i^o\big) + \gamma_i(m_i),\ r(\boldsymbol{m})(m_i - m_i^o)\big) & \text{if } m_i \in [m_i^o, m_i^*], \\ \big(a_C(\sum \boldsymbol{m}),\ r(\boldsymbol{m})(m_i - m_i^o)\big) & \text{if } m_i \geq m_i^*. \end{cases} \tag{6.9}$$

$$\mathcal{M}^U_{R,i} := \begin{cases} \phi_i\big(a_i(m_i)\big) & \text{if } m_i \leq m_i^o, \\ \phi_i\big(a_i(m_i^o) + \gamma_i(m_i)\big) + r(\boldsymbol{m})(m_i - m_i^o) & \text{if } m_i \in [m_i^o, m_i^*], \\ \phi_i\big(a_C(\sum \boldsymbol{m})\big) + r(\boldsymbol{m})(m_i - m_i^o) & \text{if } m_i \geq m_i^*. \end{cases} \tag{6.10}$$

As described mathematically above, REALFM eliminates free-riding by returning a model with accuracy equivalent to local training $a_i(m_i)$ for a device i that does not contribute more than locally optimal ($m_i \leq m_i^o$). This is accomplished in practice by degrading the final model with noisy perturbations, or continued training on skewed data. Importantly, REALFM ensures Individually Rationality, as devices receive at least as good performance as if they had trained locally (**incentivizing participation**). Devices which contribute more than what is locally optimal receive boosted accuracy $\gamma_i(m_i)$ and monetary rewards $r(\boldsymbol{m})(m_i - m_i^o)$ up to a new federated equilibrium m_i^* (detailed in Theorem 11). This process ensures that devices are incentivized to

Algorithm 10 Realistic Federated Mechanism (REALFM).
Input: Data $\boldsymbol{m}$, marginal costs $\boldsymbol{c}$, profit margin p_m, payoff/shaping/accuracy functions $\boldsymbol{\phi}$/$\boldsymbol{\gamma}$/$\boldsymbol{a}$, h local steps, T total iterations, total epochs E, initial parameters $\boldsymbol{w^1}$, loss ℓ, and step-size η.

$s_i \leftarrow m_i / \sum_{j=1}^{n} m_j$
for $t = 1, \ldots, T$ **do**
  Server distributes $\boldsymbol{w^t}$ to all devices
  for h local steps, each device i **in parallel do**
    $\boldsymbol{w_i^{t+1}} \leftarrow$ DeviceUpdate$(i, \boldsymbol{w^t})$
  $\boldsymbol{w^{t+1}} \leftarrow \sum_j s_j \boldsymbol{w_i^{t+1}}$
$r(\boldsymbol{m}) \leftarrow (1 - p_m) \frac{\phi_C\left(a_C\left(\sum_j m_j\right)\right)}{\sum_j m_j}$
for $i = 1$ to n **do**
  Compute m_i^o and m_i^* using a_i, c_i, ϕ_i, and $\gamma_i(m_i)$
  Output: (a_i^r, R_i) to device i using Equation 6.9

DeviceUpdate$(i, \boldsymbol{w})$:
$\mathcal{B} \leftarrow$ batch m_i data points
for each epoch $e = 1, \ldots, E$ **do**
  for batch $b \in \mathcal{B}$ **do**
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla \ell(\boldsymbol{w}; b)$

contribute more than what is locally optimal (**incentivizing contributions**). Finally, if devices contribute more than the new equilibrium, the server cannot provide accuracy greater than that of the trained global model $a_C(\sum \boldsymbol{m})$ as well as monetary rewards $r(\boldsymbol{m})(m_i - m_i^o)$ (**feasibility**). Below, it is detailed how the accuracy γ_i and monetary $r(\boldsymbol{m})$ rewards are constructed to ensure devices receive greater utility even when they contribute more than what is locally optimal.

Accuracy Rewards: Accuracy Shaping. Accuracy shaping is responsible for incentivizing devices to collect more data than locally optimal m_i^o. Accuracy shaping incentivizes each device i to use more data $m_i^* \geq m_i^o$ for federated training by providing improved model accuracy whose utility outstrips the marginal cost of collecting more data $c_i \cdot (m_i^* - m_i^o)$. Unlike the linear approach in Karimireddy et al. [4], REALFM performs accuracy shaping with non-linear accuracy payoffs ϕ. To overcome issues with non-linear ϕ's, an accuracy-shaping function $\gamma_i(m)$ is constructed for each device i. Devices participating in $\mathcal{M}_R$ share c_i, ϕ_i with the server.

Assumption 7. *For a given set of device contributions $\boldsymbol{m}$, the maximum accuracy attained by the server must be greater than that of any single device, $a_C(\sum \boldsymbol{m}) \geq a_i(m_i^o)\ \forall i \in [n]$.*

Assumption 7 states that the globally-trained model outperforms any locally-trained model

from the participating devices. This is valid if the direction of descent taken by the server benefits all participating devices (*i.e.,* the inner product between local gradients and the aggregated gradient is positive). This assumption may be violated in certain non-IID settings, where a participating device's local data distribution greatly differs from the majority of devices. In Chapter 6.4, various non-IID data distributions are tested, none of which violate Assumption 7.

Theorem 11 (Accuracy Shaping Guarantees)**.** *Consider a device i with marginal cost c_i and accuracy payoff function ϕ_i satisfying Assumptions 6 and 7. Denote device i's optimal local data contribution as m_i^o and its subsequent accuracy $\bar{a}_i := a_i(m_i^o)$. Define the derivative of $\phi_i(a)$ with respect to a as $\phi_i'(a)$. For any $\epsilon \to 0^+$ and marginal server reward $r(\boldsymbol{m}) \geq 0$, device i has the following accuracy-shaping function $\gamma_i(m)$ for $m \geq m_i^o$,*

$$\gamma_i := \begin{cases} \frac{-\phi_i'(\bar{a}_i)+\sqrt{\phi_i'(\bar{a})^2+2\phi_i''(\bar{a}_i)(c_i-r(\boldsymbol{m})+\epsilon)(m-m_i^o)}}{\phi_i''(\bar{a}_i)}, \\ \frac{(c_i-r(\boldsymbol{m})+\epsilon)(m-m_i^o)}{\phi_i'(\bar{a}_i)} \quad \textit{if } \phi_i''(\bar{a}_i) = 0 \end{cases} \tag{6.11}$$

Given the defined $\gamma_i(m)$, the following inequality is satisfied for $m \in [m_i^o, m_i^*]$,

$$\phi_i(\bar{a}_i + \gamma_i(m)) - \phi_i(\bar{a}_i) > (c_i - r(\boldsymbol{m}))(m - m_i^o). \tag{6.12}$$

Now, $m_i^* := \{m \geq m_i^o \mid a_C(m + \sum_{j \neq i} m_j) = \bar{a}_i + \gamma_i(m)\}$ is the optimal contribution for each device i. Device i's data contribution increases $m_i^* \geq m_i^o$ for any contribution $\boldsymbol{m}_{-i}$.

Theorem 11 defines an accuracy-shaping function γ_i (Equation 6.11) that ensures devices receive more gain in utility than loss by contributing more than is locally optimal (Equation 6.12) up to a feasible equilibrium m_i^* (the server cannot provide an accuracy beyond $a_C(\sum \boldsymbol{m})$).

Remark 14. *For a linear accuracy payoff, $\phi_i(a) = wa$ for $w > 0$, Equation 6.11 relays $\gamma_i = \frac{(c_i - r(\boldsymbol{m}) + \epsilon)(m - m_i^o)}{w}$. The accuracy-shaping function in Karimireddy et al. [4], Equation (13), is recovered with their no-reward $r(\boldsymbol{m}) = 0$ and $w = 1$ linear setting. Thus, this accuracy-shaping function generalizes the one in Karimireddy et al. [4].*

Remark 15 (Proportional Shaping)**.** *At low local accuracy $\bar{a}_i$, the values of $\phi_i'(\bar{a}_i), \phi_i''(\bar{a})$ are smaller (Assumption 6) and thus $\gamma_i(m)$ grows faster w.r.t m (Equation 6.11). Thus, for a low-accuracy device i, γ_i reaches its accuracy limit at a smaller optimal contribution m_i^*. The result is **proportional shaping**: contributions are incentivized in proportion to device performance.*

Theorem 12 (Existence of Improved Equilibrium)**.** *RealFM, $\mathcal{M}_R$ (Equation 6.9), performs accuracy-shaping with γ_i defined in Theorem 11 for each device $i \in [n]$ and some $\epsilon \to 0^+$. As such, $\mathcal{M}_R$ is Individually Rational (IR) and has a unique Nash equilibrium at which device i will contribute $m_i^* \geq m_i^o$ updates, thereby eliminating free riding. Since $\mathcal{M}_R$ is IR, devices are incentivized to participate, as they never receive more utility by not participating.*

The proofs of Theorems 11 and 12 are found within Appendix D.4.

Monetary Rewards. As detailed in Chapter 6.2, the server keeps a fraction p_m of its utility gained at the end of training (Equation 6.6). REALFM disburses the remaining $1 - p_m$ utility as a marginal monetary reward $r(\boldsymbol{m})$ for each data point contributed more than locally optimal,

$$r(\boldsymbol{m}) := (1 - p_m) \cdot \phi_C \left(a_C \left(\sum \boldsymbol{m} \right) \right) / \sum \boldsymbol{m} \;\longrightarrow\; R_i := r(\boldsymbol{m})(m_i - m_i^o). \tag{6.13}$$

The marginal monetary reward $r(\boldsymbol{m})$ is dynamic and depends upon the total data used by devices

during federated training. Therefore, $r(\boldsymbol{m})$ is unknown to devices when $\mathcal{M}_R$ is issued. However, the server computes and provides the monetary rewards once training is complete.

Since REALFM (**i**) returns model accuracy equivalent to local training if devices do not contribute more than what is locally optimal, and (**ii**) ensures that devices are provided improved utility when contributing more than locally optimal (Theorem 11), then REALFM is Individually Rational. Furthermore, since $\mathcal{M}_R$ is feasible and the utility provided is continuous and quasi-concave (Equation 6.10), there exists a Nash Equilibrium (Theorem 9). The use of accuracy shaping (Theorem 11) also ensures that devices contribute more data than is locally optimal, eliminating the free-rider effect. Experiments in Chapter 6.4 provide backing that REALFM indeed incentives devices to contribute more data and improves both device and server utility.

6.4 Experiments

To test the efficacy of REALFM, its performance is analyzed at **(i)** improving utility for the central server and devices, and **(ii)** increasing the amount of data contributions to federated training on image classification experiments. Experiments for REALFM are performed on CIFAR-10 [71] and MNIST [157].

Experimental Baselines. Few federated learning mechanisms eliminate the free-rider dilemma, and with none doing so without sharing data. Therefore, the mechanism proposed by Karimireddy et al. [4] is adapted as the baseline to compare against (it is denoted as LINEAR REALFM). REALFM is also compared to a local training baseline where the average device utility attained by devices if they did not participate in the mechanism is measured. Server utility is inferred in this instance by using the average accuracy of locally trained models.

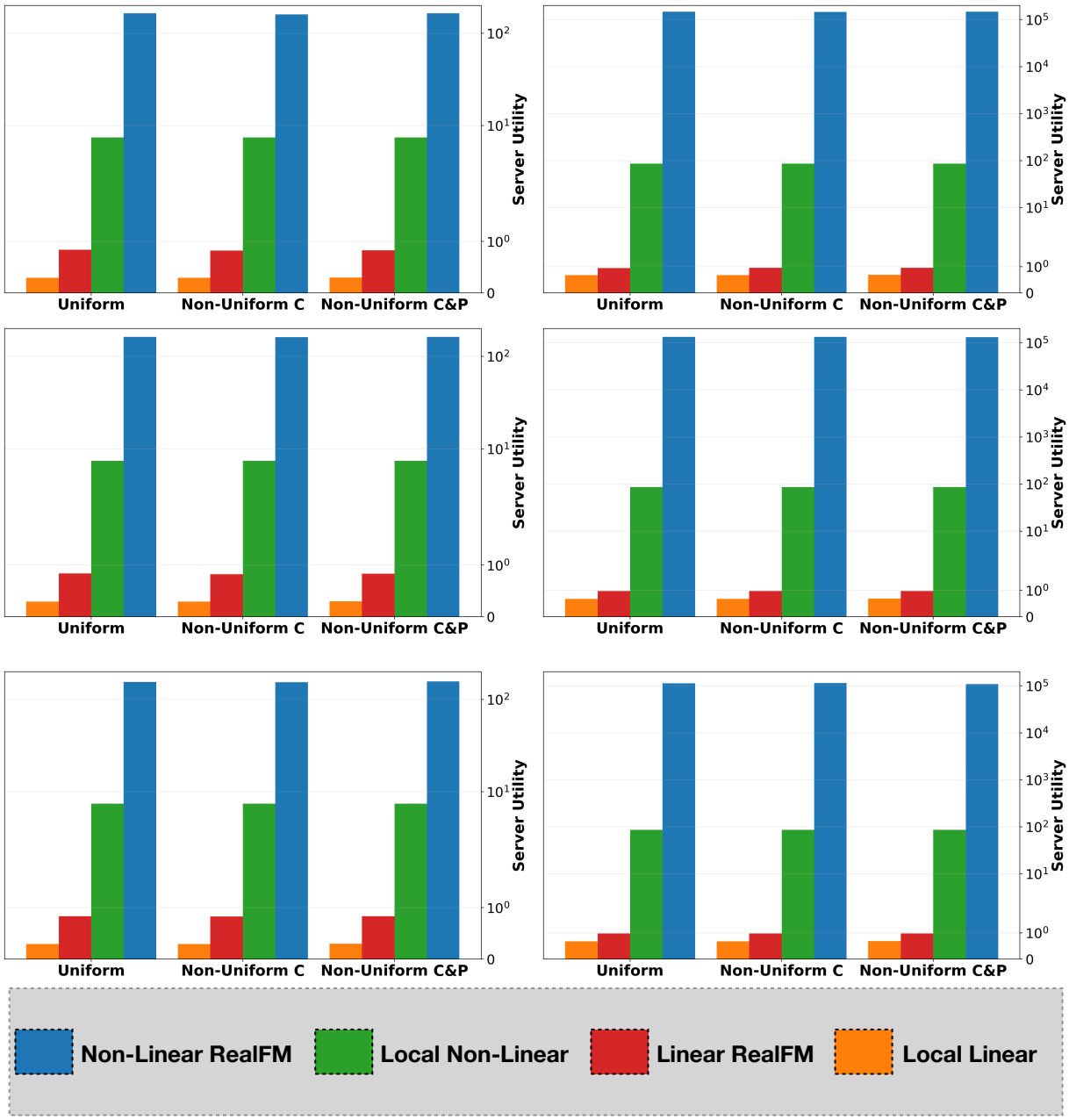


Figure 6.4: **Improved Server Utility on CIFAR-10 & MNIST.** REALFM increases server utility on CIFAR-10 (top row) and MNIST (bottom row) for 16 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a FL version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

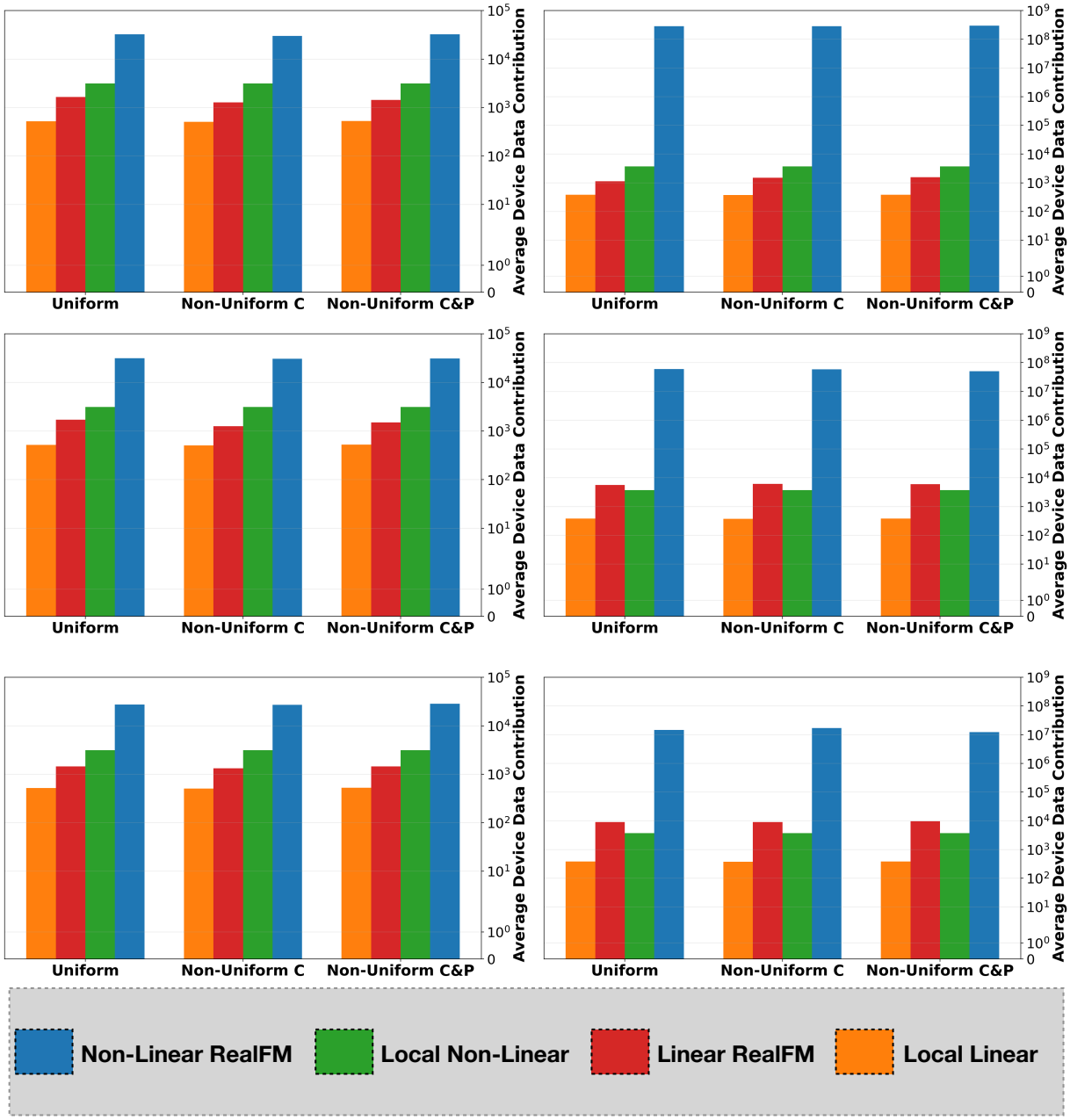


Figure 6.5: **Increased Federated Contribution on CIFAR-10 & MNIST.** REALFM incentivizes devices to use more local data during federated training on CIFAR-10 (left column) and MNIST (right column) for 16 devices compared to relevant baselines. REALFM achieves upwards of 4 magnitudes more federated contributions than LINEAR REALFM across both uniform and various heterogeneous Dirichlet data distributions (top: uniform, middle: D-0.6, bottom: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

Testing Scenarios. REALFM and its baselines are tested under homogeneous and heterogeneous device data distributions. In the heterogeneous case, two different Dirichlet distributions (parameters 0.6 and 0.3) are used to determine label proportions for each device [158]. These settings are denoted as $D-0.6$ and $D-0.3$ respectively. REALFM is also tested under non-uniform marginal costs (C) and accuracy payoffs functions (P). Accuracy plots as well 8 device results are detailed within Appendix D.2.

Increased Contributions to Federated Training. The power of REALFM, via its accuracy shaping, to incentivize devices to contribute more data points is showcased in Figure 6.5. Through its construction, REALFM's accuracy-shaping function incentivizes devices to use more data during federated training than locally optimal in exchange for greater utility. This is important for two reasons. First, incentivizing devices to contribute more than local training proves that the free-rider effect is not taking place. Second, higher data contribution lead to better-performing models and higher accuracies. This improves the utility for all participants. Overall, REALFM is superior at incentivizing contributions compared to state-of-the-art federated mechanisms.

Improved Server and Device Utility. REALFM leverages an improved reward mechanism (Equation 6.9) to boost the amount of contributions to federated training. The influx of data subsequently increases model performance (detailed in Chapter 6.1.1). This is backed up empirically: Figure 6.4 showcases upwards of *five magnitudes greater server utility* compared to state-of-the-art federated learning mechanisms. Devices participating in REALFM also boost utility *by over five magnitudes* (Figure 6.6). The improvement stems from (i) effective accuracy-shaping by REALFM and (ii) the use of non-linear accuracy payoff functions ϕ, which more precisely map the benefit derived from an increase in model accuracy.

Performance Under Non-Uniformities. Non-uniform costs and accuracy payoff func-

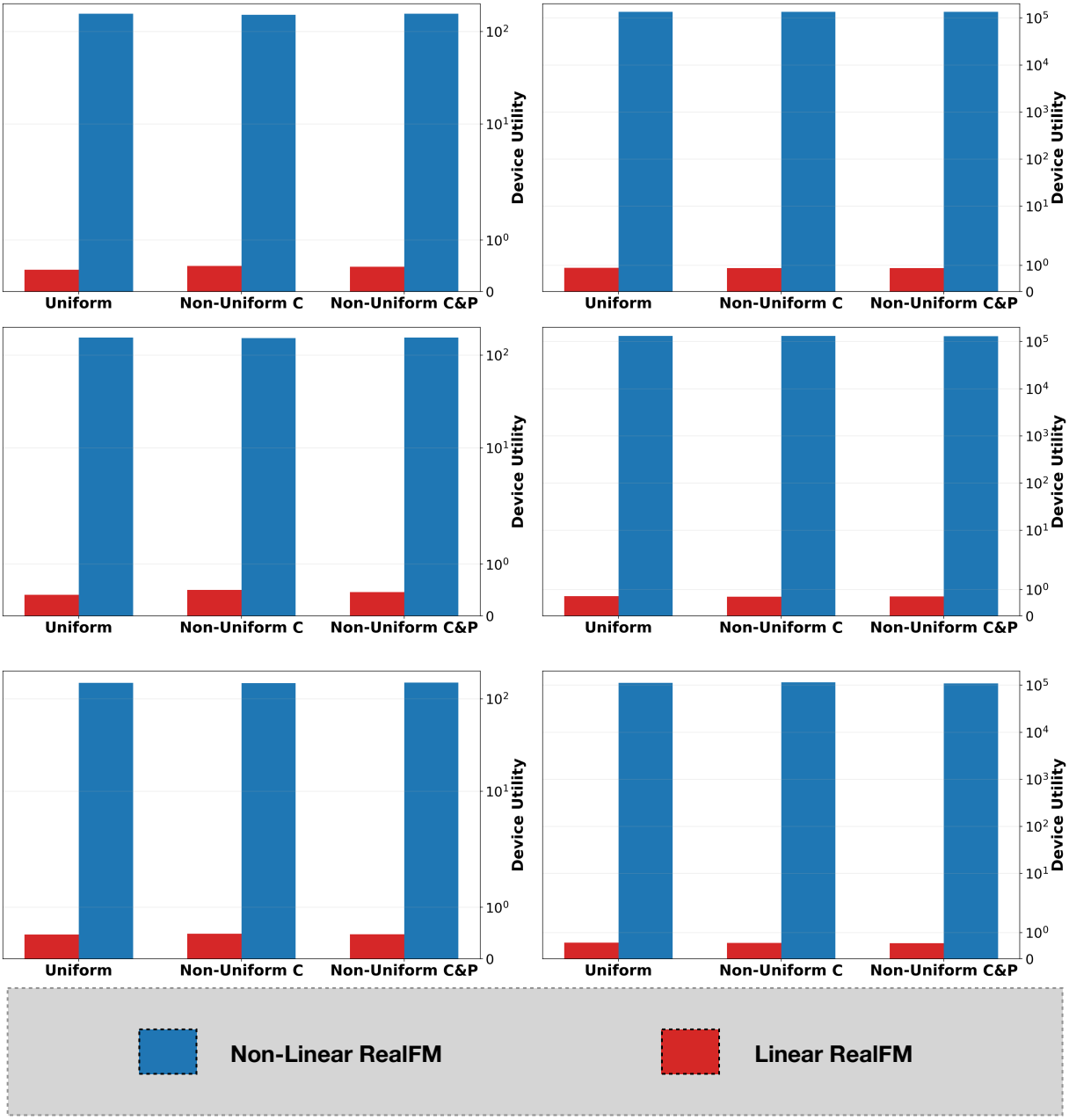


Figure 6.6: **Improved Device Utility on CIFAR-10 & MNIST.** REALFM increases device utility on CIFAR-10 (top row) and MNIST (bottom row) for 16 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a federated learning version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

tions do not affect REALFM performance. However, as expected, REALFM performance slightly degrades as device datasets become more heterogeneous (Figures 6.4 & 6.5). In this setting, both local and federated accuracies are reduced due to the difficulties of out-of-distribution generalization and model drift. Nevertheless, even under heterogeneous distributions, REALFM greatly outperforms all other baselines under both non-uniform costs and accuracy payoff functions as well as non-uniform (heterogeneous) device data distributions.

Chapter 7: Auction-Based Regulation for Artificial Intelligence [1]

Current Artificial Intelligence (AI) models have revolutionized a wide swath of industries. The recent large-scale deployment of Large Language Models (LLMs) has simultaneously boosted human productivity while sparking concern over safety (*e.g.,* hallucinations, bias, and privacy). Many industry leaders, such as Google, Meta, and OpenAI, remain embroiled in controversy surrounding bias and misinformation [178, 179, 180], safety [181, 182, 183], as well as legality and ethics [184, 185, 186] in their development and deployment of LLMs. Furthermore, irresponsible deployment of LLMs risks allowing adversaries the ability to spread misinformation or propaganda [187, 188, 189]. To date, a consistent and industry-wide solution to oversee responsible AI deployment remains elusive.

Naturally, one solution to mitigate these dangers is to increase governmental regulation over AI deployment. In the United States, there have been some strides, on federal [190] and state levels [191], to regulate the safety and security of large-scale AI systems (including LLMs). While these recent executive orders and bills highlight the necessity to develop safety standards and enact safety and security protocols, few details are offered. This follows a consistent trend of well-deserved scrutiny towards the lack of AI regulation *without the development of rigorous and realistic mathematical frameworks to regulate*.

[1] A version of Chapter 7 is under submission [177]

This chapter sets out to mathematically design a novel regulatory framework to not only enforce strict compliance, *e.g.,* safety or ethical compliance, of deployed AI models, but simultaneously to *incentivize the production of more compliant AI models.* Specifically, the AI regulatory process is formulated as an *all-pay auction*, where agents (enterprises) submit their models to a regulator. The regulator's job is twofold: **(a)** to prohibit deployment of models that fail to meet prescribed compliance thresholds, and **(b)** to incentivize compliant model production and deployment by providing additional rewards to agents that submit more compliant models than their peers. Below, an auction-based regulatory mechanism is designed, with a novel reward-payment protocol, that emits Nash Equilibria at which agents deploy models that are more compliant than a prescribed threshold.

Summary of Contributions. The main contributions of this chapter are as follows.

- ▷ Propose a Compliance-Incentivized Regulatory-Centered Auction (CIRCA), offering a simple yet practical approach to AI regulation.
- ▷ Establish, through derived Nash Equilibria, that agents are incentivized to submit models that surpass the required compliance threshold.
- ▷ Empirical results demonstrate that CIRA increases model compliance by over 20% and boosts participation rates by 15% compared to baseline regulatory mechanisms.

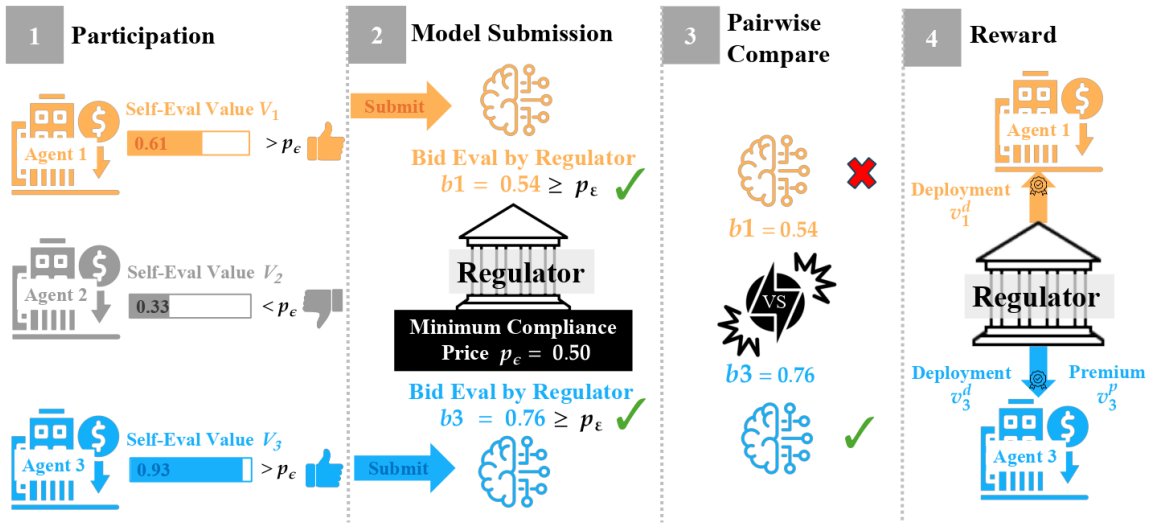


Figure 7.1: **CIRCA Schematic.** (Step 0) The regulator sets a compliance threshold, ϵ, having corresponding price, p_ϵ, required to achieve ϵ. (Step 1) Agents evaluate their total value, V_i, from model deployment value (v_i^d) and potential regulator compensation (v_i^p). Agents only participate if their total value exceeds p_ϵ. (Step 2) Participating agents submit their models to the regulator, accompanied by their bid b_i, which reflects the amount spent to improve their model's compliance level. Models with bids below p_ϵ are automatically rejected. (Step 3) The submitted models are randomly paired, and the more compliant model (*i.e.,* the higher bid) in each pair wins. In this example, agent 3 wins since $b_3 > b_1$. (Step 4) Winning models receive both a premium and deployment value (*i.e.,* agent 3 wins premium v_3^p and deployment v_3^d values), while losing models receive only the deployment value (*i.e.,* agent 1 only wins deployment value v_1^d).

7.1 Preliminaries

7.1.1 Problem Formulation

There exists a regulator R with the compliance power to set and to enforce laws and regulations (*e.g.,* U.S. government regulation on lead exposure). The regulator wants to regulate AI model deployment, by ensuring that all models meet a given compliance threshold $\epsilon \in (0, 1)$, *e.g.,* the National Institute for Occupational Safety and Health (NIOSH) regulates that N95 respirators filter out at least 95% of airborne particles. If a model does not reach the compliance threshold ϵ, then the model is deemed unsafe and the regulator bars deployment. On the other side, there are n rational model-building agents. Agents seek to maximize their own benefit, or utility.

Bidding & Evaluation. By law, each agent i must submit, or bid in auction terminology, its model $w_i \in \mathbb{R}^d$ for evaluation to the regulator before it can be approved for deployment. Let $S(w; x) : \mathbb{R}^d \to \mathbb{R}_+$ output a compliance level (the larger the better) for model w given data x. In effect, each agent, given its own data x_i, bids a compliance level $s_i^A := S(w_i; x_i)$ to the regulator. Subsequently, the regulator, using its own data x_R, independently evaluates the agent's compliance level bid as $s_i^R := S(w_i; x_R)$. Agent and regulator evaluation data is assumed to be independent and identically distributed (IID) $x_i, x_R \sim \mathcal{D}$.

Assumption 8. *Agent & regulator evaluation data is from the same distribution* $x_i, x_R \sim \mathcal{D}$.

This assumption is realistic, because agents and regulators typically rely on standardized data collection processes [192] or widely accepted datasets [193, 194] for evaluation. This ensures a fair and unbiased assessment of compliance. For example, FDA guidelines detail that data

collection should assess efficacy and safety across various subgroups, *e.g.,* demographics, while also not changing "baseline data collection determined by the clinical trial objectives" [192]. In areas such as Natural Language Processing, common datasets, or benchmarks, are employed to consistently evaluate model comprehension [193, 194], trustworthiness [195], and security [196] across various models. Therefore, it is reasonable to define agent i's compliance level bid as $s_i := \mathbb{E}_{x\sim\mathcal{D}}[S(w_i; x)]$. The scenario where evaluation data may be non-IID is addressed within Appendix E.4.

In regulatory settings, like the NIOSH example above, a scalar compliance metric is often used. If multiple compliance metrics must be monitored, S can be defined to aggregate and weigh the various metrics. This too is realistic in AI. For example, LLM safety alignment literature uses a scalar-valued reward to ensure a model is safety aligned [197, 198, 199].

Price of Compliance. A strictly increasing function $M : (0, 1) \to (0, 1)$ is assumed, determining the "price of compliance" (*i.e.,* maps compliance into cost). Simply put, higher-compliant models cost more to attain. Thus, the price of ϵ-compliance is defined as $p_\epsilon := M(\epsilon)$.

Assumption 9. *A strictly increasing function M that maps compliance to cost exists.*

One prominent existing example of this relationship is the cap-and-trade system that the Environmental Protection Agency exercises to combat pollution [200, 201]. Companies that pollute above a set emission threshold can reach compliance by purchasing allowances, or pollution deficits, from other compliant companies. Thus, pollution compliance is attained with greater cost. For an example within AI, models can achieve higher safety compliance through Machine Unlearning [202] or AI Alignment [203]. However, such methods incur greater computational and data collection costs in exchange for improved compliance. Furthermore, it has been found

empirically that larger models and longer inference attain higher levels of compliance in adversarial training, robustness transfer, and defense [204, 205]. However, larger models and longer inference increase training and inference costs. The compliance-cost relationship is validated empirically in Chapter 7.4.1.

Agent Costs. Realistically for agents, training a compliant model comes with added cost. Consequently, each agent i must decide how much money to *bid*, or spend, b_i to make its model compliant. By Assumption 9, the compliance level of an agent's model will be $s_i = M^{-1}(b_i)$.

Agent Values. *(1) Model deployment value* v_i^d. While it costs more for agents to produce compliant models, they gain value from having their models deployed. Intuitively, this can be viewed as the expected value v_i^d of agent i's model. The valuation for model deployment varies across agents (*e.g.,* Google may value having its model deployed more than Apple). *(2) Premium reward value* v_i^p. Beyond value for model deployment, the regulator can also offer additional, or premium, compensation valued as v_i^p by agents (*e.g.,* tax credits for electric vehicle producers or Fast Track and Priority Review of important drugs by the U.S. Food & Drug Administration). The regulator provides additional compensation to agents whose models exceed the prescribed compliance threshold. However, the value of this compensation varies across agents due to differing internal valuations. It is unrealistic for the regulator to compensate all agents meeting the compliance threshold due to budget constraints. Therefore, additional rewards are limited to a top-performing half of agents surpassing the threshold. This ensures targeted compensation for agents enhancing compliance while maintaining feasibility for the regulator.

Value Distribution. The total value for each agent i is defined as $V_i := v_i^d + v_i^p$, which represents the sum of the deployment value and premium compensation. Although these values may vary widely in practice, $\{V_i\}_{i=1}^n$ is normalized for all n agents to be between 0 and 1 for

analytical tractability, allowing a standardized range. Consequently, the price to achieve the compliance threshold ϵ is also normalized to fall within the $(0, 1)$ interval, *i.e.,* $p_\epsilon \in (0, 1)$. The scaling factor $\lambda_i \sim \mathcal{D}_\lambda(0, 1/2)$ dictates the proportion of total value allocated to deployment versus compensation. Therefore, (i) the deployment value is $v_i^d := (1 - \lambda_i)V_i$, and (ii) the premium compensation value is $v_i^p := \lambda_i V_i$. Both V_i and λ_i are private to each agent, though the distributions $\mathcal{D}_V$ and $\mathcal{D}_\lambda$ are known by participants. The maximum allowable factor is set at $\lambda_i = 1/2$, reflecting the realistic constraint that compensation should not exceed deployment value. Although the results in Chapter 7.3 primarily consider $\lambda_i \leq 1/2$, theoretical extensions are permissible for scenarios where $\lambda_i > 1/2$.

All-Pay Auction Formulation. Agents face a trade-off: producing higher-compliant models garners value, via the regulator, but incurs greater costs. Furthermore, in order to attain the rewards detailed above, agents must submit a model with a compliance level at least as large as ϵ. This problem is formulated as an *asymmetric all-pay auction* with *incomplete information* [206, 207, 208]. An all-pay auction is used since agents incur an unrecoverable cost, training costs, when submitting their model to regulators. The auction is formulated as asymmetric with incomplete information since valuations V_i are private and differ for each agent.

Agent Objective. The objective, for each model-building agent i, is to maximize its own utility u_i. Namely, each agent seeks to determine an optimal compliance level to bid to the regulator b_i. However, given the all-pay auction formulation, agents may need to take into account all other agents' bids $\boldsymbol{b}_{-i}$ in order to determine their optimal bid b_i^*,

$$b_i^* := \arg\max_{b_i} u_i(b_i; \boldsymbol{b}_{-i}). \tag{7.1}$$

A major portion of this chapter is constructing an auction-based mechanism, thereby designing the utility of each agent, such that each participating agent maximizes its utility when each bids more than "the price to obtain the minimum compliance threshold", *i.e.,* $b_i^* > p_\epsilon$. To begin, a simple mechanism is provided, already utilized by regulators, that does not accomplish this goal, before detailing the auction-based mechanism CIRCA that provably ensures that $b_i^* > p_\epsilon$ for all agents.

7.1.2 Related Works

Regulation Frameworks for Artificial Intelligence. A handful of work focuses on regulation frameworks for AI deployment [209, 210, 211, 212]. First, de Almeida et al. [209] details the need for AI regulation and surveys existing proposals. The proposals are ethical frameworks that express specific ethical decisions to make and dilemmas to address. However, these proposals lack a mathematical framework to incentivize provably compliant models. Rodríguez et al. [211] utilize AI models to detect collusive auctions. This work is related to this chapter but in reverse: Rodríguez et al. [211] applies AI to regulate auctions and to ensure that they are not collusive. In contrast, this chapter aims to use auctions to regulate AI deployment. Jagadeesan et al. [210] focuses on reducing barriers to entry for smaller companies that are competing against incumbent, larger companies. A multi-objective high-dimensional regression framework is proposed to impose "reputational damage" upon companies that deploy unsafe AI models. This work allows varying levels of safety constraints, where newer companies face less severe constraints in order to spur their entry into the market, which is unrealistic in many settings and only considers simple linear-regression models.

The closest related work, Yaghini et al. [212], proposes a regulation game for ensuring privacy and fairness that is formulated as a Stackelberg game. This game is a multi-agent optimization problem that is also multi-objective (for fairness and privacy). An equilibrium-search algorithm is presented to ensure that agents remain on the Pareto frontier of their objectives (although this frontier is estimated algorithmically). Notably, Yaghini et al. [212] considers only one model builder (agent) and multiple regulators that provide updates to the agent's strategy. Here, a more realistic setup is considered, where there are multiple agents and a single regulator whose goal is to incentivize compliant model deployment. It falls out of the scope of a regulator's job to collaborate with agents to optimize their strategy. Furthermore, the mechanism proposed here is simple and efficient. No Pareto frontier estimation or multiple rounds of optimization are required.

All-Pay Auctions. Compared to the dearth of literature in regulatory frameworks for AI, all-pay auctions are well-researched [206, 207, 208, 213, 214, 215, 216, 217]. These works formulate specific all-pay auctions and determine their equilibria. Some works consider settings where agents have complete information about their rivals' bids [213] while others consider incomplete information, such as only knowing the distribution of agent valuations [206, 207, 208]. One major application of all-pay auctions are crowd-sourcing competitions. Many agents participate to win a reward, with those losing incurring a small cost for their time, effort, *etc*. Di-Palantino and Vojnovic [214] is one of the first works to model crowd-sourcing competitions as an all-pay auction. Further research, such as Gemp et al. [215], have leveraged AI to design all-pay auctions for crowd-sourcing competitions. However, instead of crowd-sourcing, this chapter formulates the AI regulatory process as an asymmetric and incomplete all-pay auction. Previous analysis in this setting [206, 207, 208] is leveraged to derive Nash Equilibria.

7.2 Reserve Thresholding: Bare Minimum Regulation

The simplest method to ensure model compliance is for the regulators to set a reserve price, or minimum compliance level. This mechanism is coined the *multi-winner reserve thresholding auction*, where the regulator awards a deployment reward, v_i^d, to each agent whose model meets or exceeds the compliance threshold ϵ. Within this auction, each agent i's utility is mathematically formulated as,

$$u_i(b_i; \boldsymbol{b}_{-i}) = \begin{cases} -b_i \text{ if } b_i < p_\epsilon, \\ \\ v_i^d - b_i \text{ if } b_i \geq p_\epsilon. \end{cases} \tag{7.2}$$

However, the formulation above is ineffective at incentivizing agents to produce models above ϵ-compliance.

Theorem 13 (Reserve Thresholding Nash Equilibrium)**.** *Under Assumption 9, agents participating in Reserve Thresholding* (7.2) *have an optimal bid and utility of,*

$$b_i^* = p_\epsilon, \quad u_i(b_i^*; \boldsymbol{b}_{-i}) = v_i^d - p_\epsilon, \tag{7.3}$$

and submit models with the following compliance level,

$$s_i^* = \begin{cases} \epsilon & \textit{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ \\ 0 \textit{ (no submission)} & \textit{else.} \end{cases} \tag{7.4}$$

When a regulator implements reserve thresholding, as detailed in Theorem 13, agents exert minimal effort, submitting models that just meet the required compliance threshold ϵ. While this

Algorithm 11 Compliance-Incentivized Regulatory-Centered Auction (CIRCA).
Input: n model-building agents and a compliance level ϵ set by regulator R.

1: Each agent i receives their total value V_i and partition ratio λ_i from "nature"
2: Agents determine optimal bids b_i^* and corresponding utility $u_i(b_i^*)$ ▷ via Corollaries 4 or 5
3: Agents decide to participate, set of participating agents is $P = \{j \in [n] \mid u_j(b_j^*; \mathbf{b}_{-i}) > 0\}$
4: **for** participating agents $j \in P$ **do**
5: Spend b_j^* to build a model, with compliance level $s_j = M^{-1}(b_j^*)$, and submit it to R
6: Regulator verifies model compliances, clearing models for deployment if $s_j \geq \epsilon \; \forall j \in P$
7: Regulator pairs up agents, awarding compensation valued at v_i^p for the compliant model

approach ensures that all deployed models satisfy minimum compliance, it fails to encourage agents to build models with compliance levels exceeding ϵ. Additionally, agents whose deployment rewards are less than the cost of achieving compliance, *i.e.,* $v_i^d < p_\epsilon$, lack incentive to participate in the regulatory process, leading to reduced participation rates (Remark 16).

Remark 16 (Lack of Incentive)**.** *Each agent is only incentivized to submit a model with compliance $s_i^* = \epsilon$. Our goal is to construct a mechanism that incentivizes agents to build models that possess compliance levels exceeding the minimum threshold: $s_i^* > \epsilon$.*

7.3 Compliance-Incentivized Regulatory-Centered Auction (CIRCA)

To alleviate the lack of incentives inherent within simple regulatory auctions, such as the one in Chapter 7.2, a regulatory all-pay auction is proposed that mandates an equilibrium where devices *submit models with compliance levels exceeding* ϵ.

Algorithm Description. The core component of the auction is that agent compliance levels are randomly compared against one another, with the regulator rewarding those having the superior compliant model with premium compensation. Performing the randomization process multiple times reduces the likelihood of unfair outcomes. Only agents with models that achieve a compliance level of ϵ or higher are eligible to participate in the comparison process; models that

do not meet this threshold are automatically rejected. The detailed algorithmic block of CIRCA is depicted in Algorithm 11.

Agent Utility. The utility for each agent i is therefore defined as in Equation (7.5).

$$u_i(b_i; \boldsymbol{b}_{-i}) = \left(v_i^d + v_i^p \cdot 1_{(b_i > b_j)}\right) \cdot 1_{(b_i \geq p_\epsilon)} - b_i. \tag{7.5}$$

Per regulation guidelines, the compliance criteria of an accepted model must at least be ϵ. Equation (7.5) dictates that values are only realized by each agent if its model produces a bid larger than the required cost to reach ϵ-compliance, $1_{(b_i \geq p_\epsilon)}$. Furthermore, agents only realize additional compensation value v_i^p from the regulator if their compliance level outperforms a randomly selected agent j, $1_{(b_i > b_j)}$. Any agent that bids $b_i = 1$ will automatically win and realize both v_i^p and v_j^w. It is important to note that the cost that every agent incurs when building its model is sunk: if the model is not cleared for deployment, the cost $-b_i$ is still incurred. The agent utility is rewritten in a piece-wise manner below,

$$u_i(b_i; \boldsymbol{b}_{-i}) = \begin{cases} -b_i \text{ if } b_i < p_\epsilon, \\ v_i^d - b_i \text{ if } b_i \geq p_\epsilon \text{ and } b_i < b_j \text{ randomly sampled agent bid } b_j, \\ v_i^d + v_i^p - b_i \text{ if } b_i \geq p_\epsilon \text{ and } b_i > b_j \text{ randomly sampled agent bid } b_j. \end{cases} \tag{7.6}$$

By introducing additional compensation, v_i^p, and, crucially, conditioning it on whether an agent's model is more compliant than that of another random agent, it becomes rational for agents to bid more than the price to obtain the minimum compliance threshold (unlike Theorem 13).

Incentivizing Agents to Build Compliant Models. A guarantee is established such that

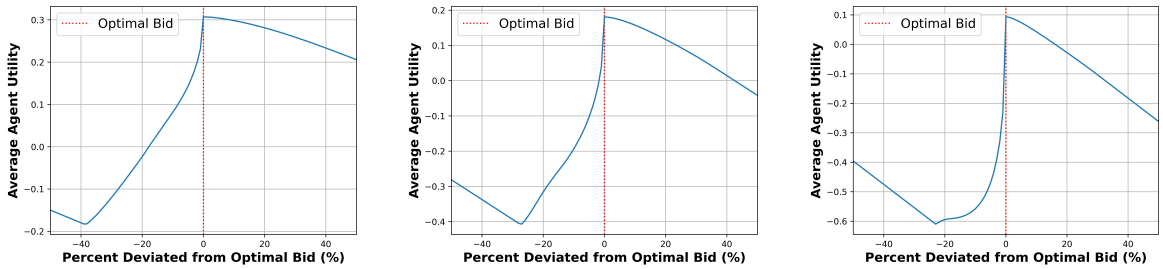


Figure 7.2: **Validation of Uniform Nash Bidding Equilibrium.** Agent utility is maximized when agents follow the theoretically optimal bidding function shown in Equation (7.8). Across varying compliance prices, $p_\epsilon = 0.25$ (left), 0.5 (middle), 0.75 (right), agents attain less utility when they deviate from the optimal bid (red line) derived in Corollary 4.

agents participating in CIRCA *maximize their utility with an optimal bid* b_i^* *that is larger than "the price required to attain* ϵ *compliance" (i.e.,* $b_i^* > p_\epsilon$) in Theorem 14 below. Furthermore, agents bid in proportion to the value for additional compensation v_i^p that the regulator offers for extra-compliant models.

Theorem 14. *Agents participating in* CIRCA (7.6) *will follow an optimal bidding strategy* $\hat{b}_i^*$,

$$\hat{b}_i^* := p_\epsilon + v_i^p F_v(v_i^p) - \int_0^{v_i^p} F_v(z)dz > p_\epsilon, \tag{7.7}$$

where $F_v(\cdot)$ *denotes the cumulative density function of the random premium reward variable corresponding to the premium reward* $v_i^p = V_i\lambda_i$.

Theorem 14 applies to any distribution for V_i and λ_i on $[0, 1]$ and $[0, 1/2]$, i.e., $V_i \sim \mathcal{D}_V(0, 1)$ and $\lambda_i \sim \mathcal{D}_\lambda(0, 1/2)$, respectively. Determining specific optimal bids, utility, and model compliance levels requires given distributions for V_i and λ_i. Analysis of all-pay auctions [206, 207, 208], as well as many other types of auctions, often assume a Uniform distribution for valuations. Therefore, the first analysis of CIRCA, below in Corollary 4, presumes Uniform distributions for V_i and λ_i.

(Special Case 1): Uniform V_i and λ_i: Optimal Agent Strategy. Corollary 4 determines that a participating agent's optimal strategy to maximize its utility is to submit a model with compliance levels larger than ϵ when their values V_i and λ_i come from a Uniform distribution.

Corollary 4 (Uniform Nash Bidding Equilibrium)**.** *Under Assumption 9, for agents having total value V_i and scaling factor λ_i both stemming from a Uniform distribution, with $v_i^d = (1-\lambda_i)V_i$, and $v_i^p = \lambda_i V_i$, their optimal bid and utility participating in* CIRCA (7.6) *are,*

$$b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2(\ln(2v_i^p) - 1/2) + p_\epsilon^2}{8(p_\epsilon - 1)} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}, \end{cases} \tag{7.8}$$

$$u_i(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} \frac{2(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} + v_i^d - b_i^* & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ \frac{2(v_i^p)^2(\ln(2p_\epsilon) - 1) + p_\epsilon}{p_\epsilon - 1} + v_i^d - b_i^* & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}. \end{cases} \tag{7.9}$$

Participating agents submit models with compliance,

$$s_i^* := \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no submission)} & \text{else}. \end{cases} \tag{7.10}$$

(Special Case 2): Beta V_i and Uniform λ_i: Optimal Agent Strategy. In many instances, a realistic distribution for V_i is a Beta distribution with $\alpha, \beta = 2$. This distribution is Gaussian-like, with the bulk of the probability density placed in the middle. As such, it is realistic when agent values do not congregate amongst one another and outliers (near 0 or 1) are rare. The performance of CIRCA in this setting is analyzed in Corollary 5. Corollary 5 states that, under a Beta(2,2) distribution for V_i, agent i maximizes its utility with an optimal bid b_i^* larger than the price of

ϵ compliance, $b_i^* > p_\epsilon$, resulting in a model above the ϵ-compliance threshold. Furthermore, Corollaries 4 and 5 surpass the baseline optimal bid $b_i^* = p_\epsilon$ for Reserve Thresholding detailed in Theorem 13.

Corollary 5 (Beta Nash Bidding Equilibrium)**.** *Under Assumption 9, let agents have total value V_i and scaling factor λ_i stem from Beta ($\alpha = \beta = 2$) and Uniform distributions respectively, with $v_i^d = (1-\lambda_i)V_i$ and $v_i^p = \lambda_i V_i$. Denote the CDF of the Beta distribution on $[0,1]$ as $F_\beta(x) = 3x^2 - 2x^3$. The optimal bid and utility for agents participating in* CIRCA *(7.6) are,*

$$b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{3(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2\left(6(v_i^p)^2 - 8v_i^p + 3\right) + p_\epsilon^3(3p_\epsilon - 4)}{8(1 - F_\beta(p_\epsilon))} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}, \end{cases} \tag{7.11}$$

$$u(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} v_i^d + \frac{6(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ v_i^d + \frac{v_i^p\left(8(v_i^p)^3 - 12(v_i^p)^2 + 6v_i^p + p_\epsilon^2(2p_\epsilon - 3)\right)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } \frac{p_\epsilon}{2} \le v_i^p \le 1/2. \end{cases} \tag{7.12}$$

Participating agents submit models with compliance,

$$s_i^* = \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no submission)} & \text{else}. \end{cases} \tag{7.13}$$

Remark 17 (Improved Model Compliance)**.** *As shown in Corollaries 4 and 5, participating agents will submit models that are more compliant than the regulator's compliance threshold, $s_i^* = M^{-1}\left(b_i^*\right) > \epsilon$.*

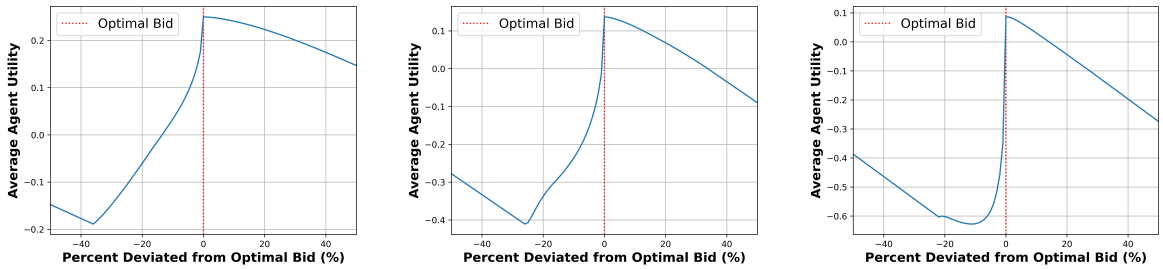


Figure 7.3: **Validation of Beta Nash Bidding Equilibrium.** Akin to the Uniform results, agent utility is maximized when agents follow the theoretically optimal bidding function shown in Equation (7.11). Across varying compliance prices, $p_\epsilon = 0.25$ (left), 0.5 (middle), 0.75 (right), agents attain less utility when they deviate from the optimal bid (red line) derived in Corollary 5.

Remark 18 (Improved Utility & Participation)**.** *Through the introduction of regulator's premium compensation, agent utility is improved, in Equations* (7.9) *and* (7.12)*, versus Reserve Thresholding in Equation* (7.3)*. As a result, more agents break the zero-utility barrier of entry for participation, boosting both overall agent utility and participation rate.*

The proofs of Theorems 13 and 14 as well as Corollaries 4 and 5 are found within Appendix E.2. Since the premium compensation value v_i^p is a product of two random variables, the PDF and CDF of v_i^p becomes a piece-wise function (as shown within Appendix E.2). As a result, the optimal bidding and subsequent utility also becomes piece-wise in both Corollaries 4 and 5. Empirically, the correctness of the computed PDF and CDFs are verified within Appendix E.1.

7.4 Experiments

Chapter 7.3 demonstrates that the compliance-incentivized regulatory-centered auction, CIRCA, creates incentives for any agents to submit compliant-exceeding models and to participate at rates higher than the baseline Reserve Thresholding mechanism in Chapter 7.2. Below, these theoretical results are validated empirically.

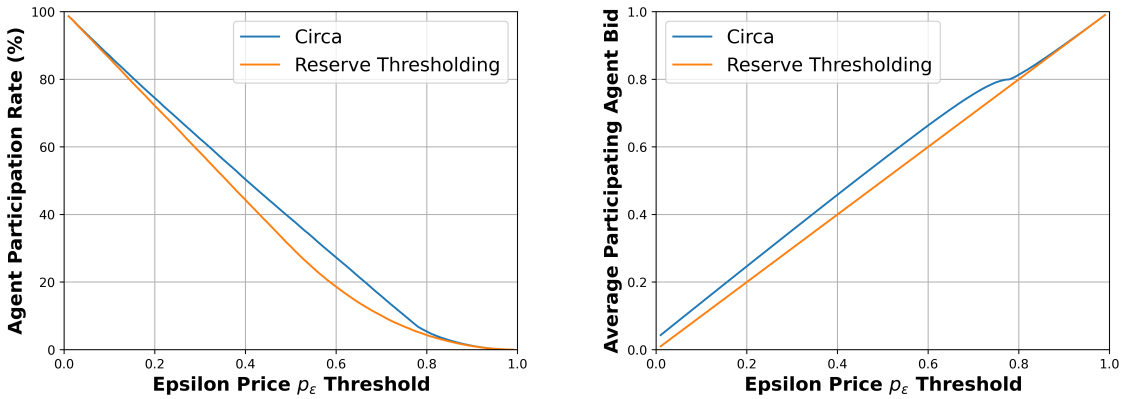


Figure 7.4: **Improved Compliance and Participation with Uniform Values**. When total value stems from a Uniform distribution $V_i \sim U(0,1)$, agents participate at a higher rate in SIRA than Reserve Thresholding (left) and spend more to train compliant models (right).

Experimental Setup. A regulatory setting with $n = 100,000$ agents is simulated below. Each agent i receives a random total value V_i from either a Uniform (Corollary 4) or Beta(2,2) (Corollary 5) distribution. Each agent also receives a scaling factor λ_i that splits the total value into deployment $v_i^d = (1-\lambda_i)V_i$ and premium compensation $v_i^p = \lambda_i V_i$ values. Once the private values are provided, agents calculate their bid according to the optimal strategies in Theorem 13 and Corollaries 4 & 5.

Lack of Baseline Regulatory Mechanisms. To the best of knowledge, there are no other comparable compliance mechanisms to regulate AI. As a result, the Reserve Threshold mechanism that is proposed in Chapter 7.2 is used as a baseline. While simple, the Reserve Threshold mechanism is a realistic baseline to compare against. For example, there exist regulatory bodies, like the Environmental Protection Agency (EPA), that follow similar steps before clearing products (*e.g.,* the EPA authorizes permits for discharging pollutants into water sources once water quality criteria are met).

Verifiable Nash Bidding Equilibria. The first experimental goal is to validate that the

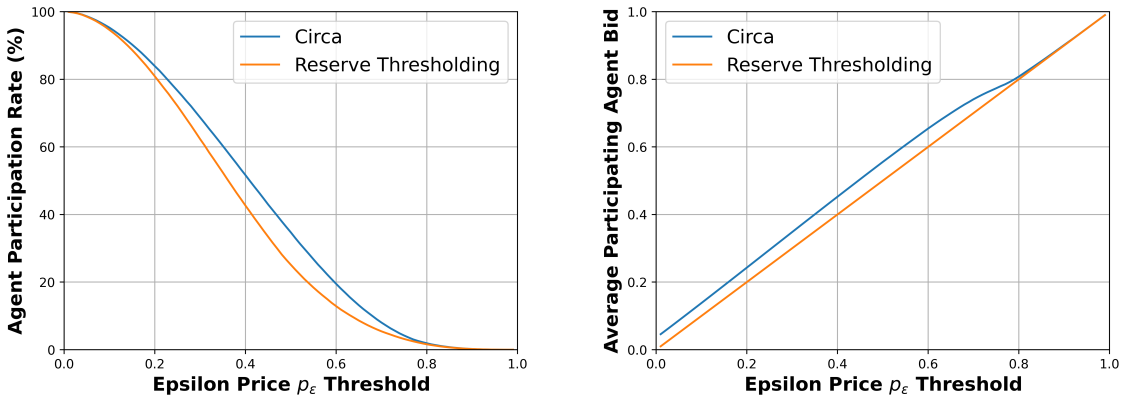


Figure 7.5: **Improved Compliance and Participation with Beta Values**. When total value stems from a Beta distribution $V_i \sim \text{Beta}(\alpha = \beta = 2)$, agents participate at a higher rate in SIRA than Reserve Thresholding (left) and spend more to train compliant models (right).

theoretical bidding functions found in Corollaries 4 and 5 constitute Nash Equilibria. That is, an agent receives worse utility if it deviates from this bidding strategy if other agents abide by it. To test this, the optimal bid for a single agent is compared versus $100,000$ others. The single agent's optimal bid is varied on a range up to $\pm 50\%$. Note that comparisons only occur if the other agent's bid is at least p_ϵ, in order to accurately reflect how the auction mechanism in Algorithm 11 functions.

In Figures 7.2 and 7.3, the average utility over all $100,000$ comparisons is plotted. *One can see that both the Uniform and Beta optimal-bidding functions maximize agent utility and thus constitute Nash Equilibria.* Utility decays much quicker when reducing the bid, since agents are **(i)** less likely to win the premium reward and **(ii)** at risk of losing the value from deployment if the bid does not reach p_ϵ. At a certain point, utility increases linearly once the agent continuously fails to bid p_ϵ. The linear improvement stems from the agent saving the cost of its bid, $-b_i$, shown in Equation (7.6).

Improved Agent Participation and Bid Size. For both Uniform and Beta(2,2) distribu-

tions, shown in Figures 7.4 and 7.5, the proposed mechanism (CIRCA) increases participation rates and average bids by upwards of 15% and 20% respectively. At the endpoints of possible price thresholds, $p_\epsilon = 0$ and 1, both mechanisms perform similarly. The reason is that at a low compliance threshold price $p_\epsilon \approx 0$, agents are highly likely to have a total value V_i larger than a value close to zero. The inverse is true for $p_\epsilon \approx 1$, where it is unlikely that agents will have a total value V_i larger than a value close to 1. The proposed mechanism shines when compliance threshold prices are in the middle; the premium compensation offered by the regulator incentivizes agents to participate and bid more for the chance to win.

7.4.1 Compliance-Cost Ablation Study

Below, an ablation study is conducted to demonstrate that in realistic settings, compliance is mapped to cost in a monotonically increasing way (as detailed in Assumption 9). While there are many compliance metrics to consider when gauging AI deployment, model fairness is analyzed, via equalized odds, for image classification in this study. Equalized odds measures if different groups have similar true positive rates and false positive rates (lower is better). Multiple VGG-16 models are trained on the Fairface dataset [218] for fifty epochs (repeated ten times with different random seeds), and consider a gender classification task with race as the sensitive attribute. Models with the largest validation classification accuracy during training are selected for testing.

Many types of costs exist for training compliant models, such as extensive architecture and hyper-parameter search. In this study, the cost of an agent acquiring more minority class data is considered. Acquiring more minority class data leads to a larger and more balanced dataset.

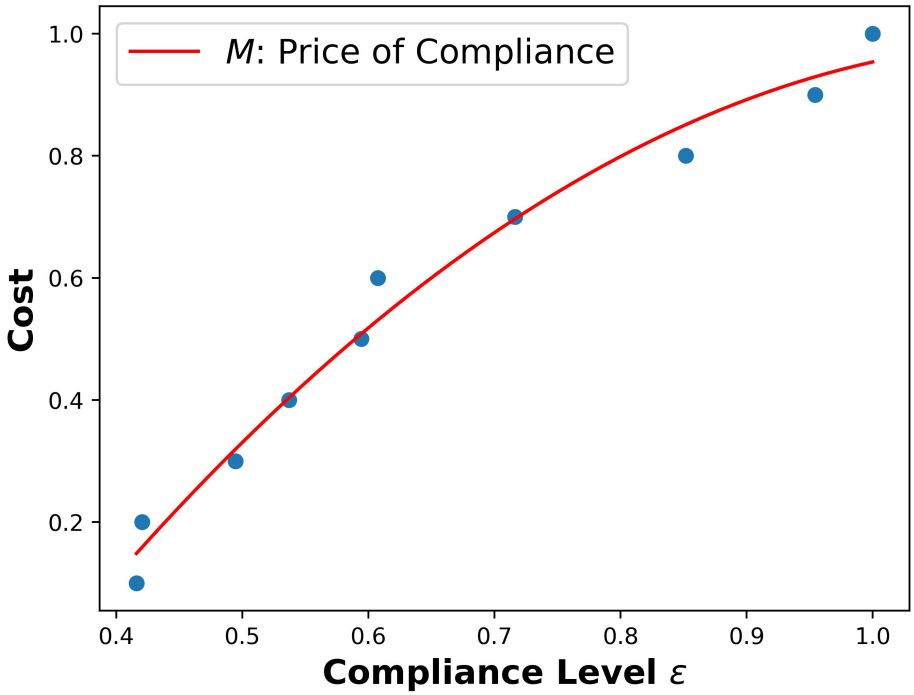


Figure 7.6: **Strictly Monotonic Compliance-Cost Relationship.** As the percentage of minority class data increases (greater cost), equalized odds metric improves (greater compliance) on Fairface.

Various mixtures of training data are simulated, starting from a 95:5 skew and scaling up to fully balanced training data with respect to the sensitive attribute. In this study, equalized odds performance is gauged on well-balanced test data for the models trained on various mixtures of data. The results of this ablation study are shown in Table 7.1 and Figure 7.6.

Table 7.1: Equalized Odds Performance as Minority Class Data Increases.

Minority Class %	**Mean Equalized Odds Score**
5%	22.55
10%	22.31
15%	18.97
20%	17.46
25%	15.78
30%	15.44
35%	13.09
40%	11.01
45%	9.83
50%	9.38

As expected, in Table 7.1, the equalized odds score decreases (more compliant model) when collecting more minority class data (increased cost). To adjust equalized odds to fit into the setting where $\epsilon \in (0, 1)$, the original equalized odds score are inverted and normalized. In Figure 7.6, one can see that compliance level is indeed monotonically increasing with respect to the cost.

Part III

Conclusion, Future Work, Appendices, and Bibliography

Chapter 8: Conclusion and Future Work

This dissertation provides solutions to the overarching questions found on Page 4 and restated below, that detail the gaps between the current and desirable outcomes of edge learning.

In the age of Big Data and large-scale AI . . .

(1) How can memory- and computation-constrained edge devices more efficiently train large, high-performing AI models?

(2) How can edge-learning frameworks incorporate mechanisms to promote outcomes that benefit participants while simultaneously ensuring fairness and AI safety?

Throughout my dissertation, I proposed several algorithms and mechanisms that assist in converting the promise of edge learning into realistic performance. This is accomplished by removing many barriers that resource-constrained devices face within the edge-learning setting. These include easing communication bottlenecks and subsequent slowdowns (Chapter 2), reducing memory and computational costs during training and inference (Chapters 3 and 4), eliminating federated free riding while incentivizing truthfulness and increased contribution (Chapters 5 and 6), and, finally, instituting regulation to ensure safe, compliant AI model deployment (Chapter 7).

To alleviate communication bottlenecks, SWIFT, a wait-free decentralized federated learning algorithm, is introduced in Chapter 2. SWIFT delivers on the promises of decentralized feder-

ated learning: a low communication and run-time algorithm (scalable) that attains state-of-the-art loss (high-performing). Empirically, SWIFT reduces communication time by almost 90% compared to baseline decentralized federated learning algorithms.

In Chapter 3, a new hashing family, PGHash, is introduced. PGHash enables the generation of multiple highly-accurate hash tables, via locality-sensitive hashing, from a compressed version of a large weight matrix. Furthermore, all hashing is performed privately without the assistance of a central server. These hash tables are used to dynamically select the most important weights (neurons) to use during a forward pass, while discarding the rest. This alleviates memory, communication, and privacy costs associated with conventional locality-sensitive hashing approaches for training. Compared to other locality-sensitive hashing baselines, which are unequipped to use within the edge-learning setting, PGHash is competitive at estimating cosine similarity and training large-scale recommender systems.

Chapter 4 addresses the ever-increasing size of recommender systems through intelligent compression and recovery of large and sparse labels. By compressing labels in a manner that preserves label-wise similarity, DECLARE is able to reduce memory footprint and computational costs of recommender systems by up to 99.975%. This is accomplished without sacrificing predictive performance: DECLARE not only maintains near state-of-the-art performance of uncompressed recommender systems, it actually *improves* performance compared to uncompressed systems (by nearly 6%) on large extreme multi-label classification datasets such as Delicious-200K.

Chapters 5 and 6 manage unrestrained free riding in edge-learning settings. It is often the best tactic, for a device, to contribute little to nothing during training while reaping a well-trained final global model. Both chapters propose Individually Rational mechanisms that provably eliminate device free riding; devices are never worse off by participating. In Chapter 5, issues of

device truthfulness are resolved. FACT leverages a novel penalization and sandwich mechanism to shift each device's best strategy to report its true cost with the server (truthfulness) and use as much data as it would on its own (no free riding). Empirically, FACT enforces agent truthfulness while reducing agent loss by upwards of a factor of four. In Chapter 6, the first federated mechanism to simultaneously incentivize device participation and contribution is proposed. Unlike other federated mechanisms, REALFM utilizes a non-linear relationship between model accuracy and utility, allows heterogeneous data distributions, removes data sharing requirements, and models central server utility. Empirically, REALFM (i) improves server and device utility, (ii) increases federated contributions, and (iii) generates higher-performing models during federated training compared to its peer federated mechanisms.

Finally, in Chapter 7, a regulatory framework (CIRCA) is proposed to ensure that AI models are vetted for compliance before deployment for public use. The key contribution of this framework is the development of an auction-based regulatory mechanism that simultaneously (i) enforces compliance standards and (ii) incentivizes agents to exceed minimum compliance thresholds. This approach provably encourages broader participation and the development of safe, compliant models compared to baseline regulatory methods. Empirical results confirm that this mechanism increases agent participation by 15% and raises agent spending on compliance by 20%, demonstrating its effectiveness to promote safe, compliant AI deployment.

Many future avenues of research exist within the areas of learning and development on the edge. The ones that are objectively exciting include:

▷ **Stronger compressed multi-label performance via multi-label logistic loss**. Recent work by Mao et al. [219] has developed new loss functions better suited for multi-label classification performance. Combining this new loss function with our compression algorithm in Chapter 4

may yield further improvements to on-device recommender system training and performance.

▷ **Sequential compressed-model generation via the lottery-ticket hypothesis** [220]. Constructing and training a deep neural network on-device is expensive and sometimes infeasible for resource-constrained devices. To train a model within this setting, a model can grow deeper, layer by layer, over time in a compressed, memory-efficient manner. The idea is to sequentially add a new layer, train it for a few epochs, prune the layer, and then repeat this process until desired performance has been reached. This is something that will be explored in the coming months.

▷ **Improved frameworks for AI regulation**. Ensuring that AI models are safe for deployment is critical as AI becomes more and more intelligent. While the regulatory framework proposed in Chapter 7 is a great start towards a mathematically rigorous and incentive-laden scheme to improve AI regulation, more work remains. For example, developing theoretical guarantees under heterogeneous settings, *e.g.,* device-varying data distributions and compliance-cost relationships, is an open question. Concurrently, fine-grained steps towards implementing frameworks in practice (*e.g.,* a technical policy report) is needed within the AI regulatory space.

▷ **Antitrust mechanisms**. Today's industry consists of sparse competition and concentrated market control by a few participants. Designing mechanisms to incentivize competitive markets and reduce entrance barriers for smaller companies is important.

Appendix A: Appendix to Chapter 2

A.1 Experimental Setup

Below, the hyper-parameters selected for the SWIFT experiments in Chapter 2.4.

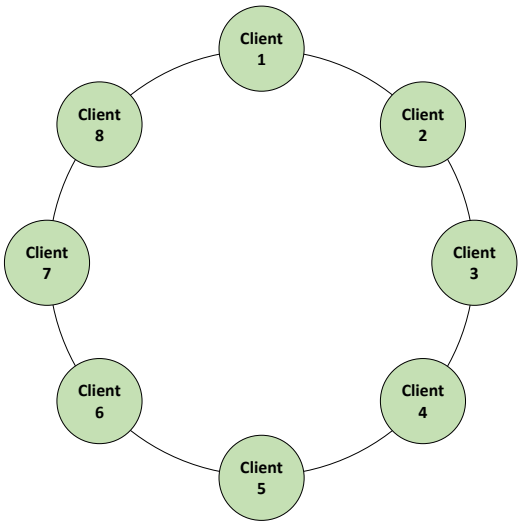
Table A.1: Hyper-parameters for all SWIFT Experiments.

Experiment Type	Model	Epochs E	γ	γ Decay (Rate, E, Freq.)	M	Weight Decay	Momentum
Baseline	ResNet-18	200	0.1	(1/10, 81 & 122, Single)	32	10^{-4}	0.9
Vary non-IIDness	ResNet-18	300	0.8	(1/2, 200, 10)	32	10^{-4}	0.9
Vary Heterogeneity	ResNet-18	100	0.1	(1/2, 50, 10)	32	10^{-4}	0.9
Vary # of Clients	ResNet-18	200	0.1	(1/10, 81 & 122, Single)	32	10^{-4}	0.9
Vary Topology	ResNet-50	200	0.1	(1/2, 100, 10)	64	10^{-4}	0.9

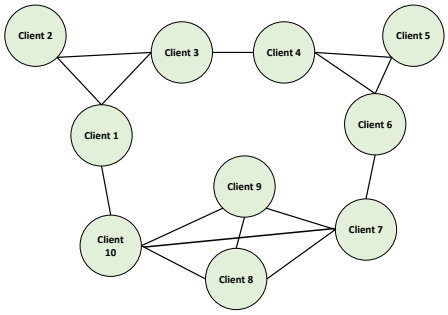
In Table A.1, one can see that the step-size decay column is split into the following sections: rate, E, and frequency. The rate is the decay rate for the step-size. For example, in the Baseline row, the step-size decays by 1/10. The term E is the epoch at which decay begins during training. In the Baseline row, the step-size decays at $E = 81$ and 122. Frequency simply is how often the step-size decays. For example, in the Vary Topology row, the step-size decays every 10 epochs.

A.1.1 Network Topologies

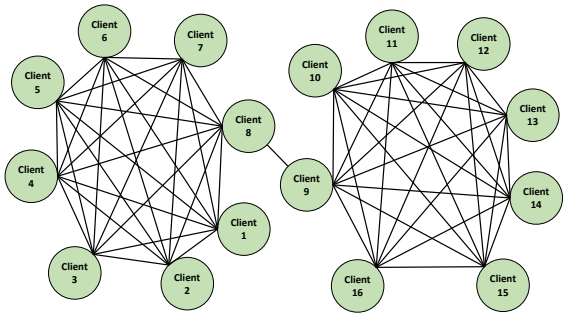
As detailed within the multiple works [221, 222], the ring of cliques network topology is a realistic topology in the decentralized setting. In many real-world setting, like one's home



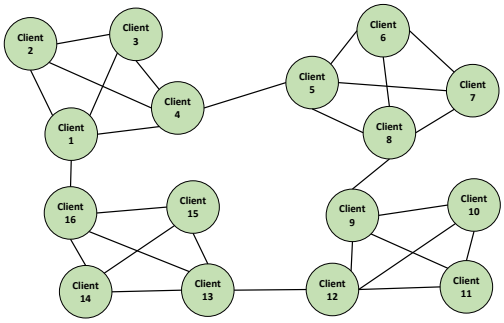
(a) 8-Device Ring.



(b) 10-Device Ring of 3 Clusters.



(c) 16-Device Ring of 2 Clusters.



(d) 16-Device Ring of 4 Clusters.

Figure A.1: **Diagram of Utilized Network Topologies**. The main network topologies used within SWIFT's experiments are a simple ring topology as well as a ring-of-cliques.

(smart appliances, smart speakers/displays, phones, etc.), devices are connected together in a small clique. Only a small amount of these devices have connections to other devices outside the cluster (like a phone or router). This network topology is utilized within experiments due to its realistic nature. Furthermore, [222] shows that a ring of clique topology can be used in the decentralized setting to reduce the impact of label distribution skew.

A.1.2 Data Partitioning

For all experiments, the training set is evenly partitioned amongst the number of devices training the consensus model. While the size of each device's partition is equal, testing is performed with data that is both (1) independent and identically distributed (IID) among all devices and (2) sorted by class and is thus non-IID. For the IID setting, devices are assigned data uniformly at random over all classes. In non-IID settings, each device is assigned a subset of classes from which it will receive data exclusively. The c classes are assigned in the following steps:

1. The class subset size n_c, for all devices, is determined as the ceiling of the number of classes per device $n_c = \lceil \frac{c}{n} \rceil$. Each class k within the class subset will take up $1/n_c$ of the device's total data partition if possible.

2. The classes within each device's class subset are assigned cyclically, starting with the first device. The first device selects the first n_c classes, the second device selects the next n_c classes, and so on. Classes can, in some cases, be assigned to multiple devices. If the final class has been assigned, and more devices have yet to be assigned any classes, classes can be re-assigned starting at the first class.

3. Each device is assigned data from the classes in their class subset cyclically ($1/n_c$ of its partition per class), starting with the first device. If no more data is available from a specific class, the remaining portion is filled by data from the next class.

Since this data partitioning process is followed within our experiments, each device is assigned equal partitions of data. Therefore, following previous works [2, 31, 41, 66], the device influence scores are set as uniform for all devices $\alpha_i = 1/n \; \forall i \in \mathcal{V}$.

A.2 SWIFT Properties

A.2.1 Properties of Communication Matrices

Stochastic Matrices. Within Chapter 2, a non-symmetric, non-doubly-stochastic matrix $C_{i_t}^t$ for device averaging is used. Utilizing $C_{i_t}^t$ comes with some analysis issues (it is non-symmetric), however it provides the wait-free nature of SWIFT. Interestingly, $C_{i_t}^t$ does have some unique properties: it is column-stochastic. Lemma 5 proves that the product of stochastic matrices converges exponentially to a stochastic vector with common ratio $\nu \in [0, 1)$.

Symmetric and Doubly-Stochastcic Matrices. As mentioned in Section 2.3, Algorithm 4 selects device weights such that the communication matrix $\bar{C}^t$ is symmetric and doubly-stochastic under expectation. By Lemma 4, there exists a scalar $\rho \in [0, 1)$ such that,

$$\max\{|\lambda_2\big((\bar{C}^t)^\intercal \bar{C}^t\big)|, |\lambda_n\big((\bar{C}^t)^\intercal \bar{C}^t\big)|\} \leq \rho, \ \forall t.$$

This parameter ρ reflects the connectivity of the underlying graph topology. The value of ρ is inversely proportionate to how fast information spreads in the device network. A small value of ρ results in information spreading faster ($\rho = 0$ in centralized settings).

Within the theoretical analysis below, the parameter ρ_ν is denoted as a combination of ρ and ν,

$$\rho_\nu := \frac{n-1}{n}\left(\frac{7}{2(1-\rho)} + \frac{\sqrt{\rho}}{(1-\sqrt{\rho})^2} + \frac{384}{(1-\nu^2)}\right). \tag{A.1}$$

A.2.2 Instantaneous Communication and Overcoming Computational Delay

Similar to AD-PSGD [31], it is assumed that model transmission is instantaneous between devices. Within [31], two algorithms (Algorithms 2 and 3) are provided within the appendix to provide realistic multi-thread implementation of AD-PSGD. The computational thread of Algorithm 2 is explicitly told to wait at Line 5 until the gradient buffer is empty. Unless a gradient queue is utilized, which alters the effective batch-size from the constant M, the computational thread must wait for model transmission if there exists communication delay. Important future work for SWIFT, and within the field of asynchronous decentralized FL in general, is to remove this instantaneous delay assumption and build protocols to overcome it (and theoretically guarantee convergence in its presence).

Eliminating computational delay is a key novelty of SWIFT. Delayed computations are handled by SWIFT through its communication matrix. Unlike AD-PSGD, no model overwriting can occur while a device is performing gradient computations (Line 9 of Algorithm 3). This only occurs due to the novel non-doubly stochastic communication matrix. Therefore, devices will never be computing gradients with a delayed model. This, of course, assumes that model transmission is instantaneous.

SWIFT deals with a slow device i (having delayed computations) by allowing a faster device j to reuse a stored model of device i until device i finally finishes its model update. The stored model is still up to date (and not delayed) since device i has not finished its gradient computations, updated its model, and sent out its updated model to neighboring devices (as it experiences delayed computations). If a device has not received a message from a neighbor, then their stored model for that neighbor is still its most up-to-date model.

A.2.3 Optimal Step-Size Under Uniform Device Influence

The defined step-size γ and total iterations T for SWIFT is,

$$\gamma := \frac{\sqrt{Mn^2\Delta_f}}{\sqrt{TL}+\sqrt{M}} \leq \sqrt{\frac{Mn^2\Delta_f}{TL}}, \quad T \geq 193^2 LM\Delta_f\rho_\nu^2 n^4\alpha_{max}^2. \tag{A.2}$$

Therefore, γ can be rewritten as,

$$\gamma \leq \sqrt{\frac{Mn^2\Delta_f}{(193^2 LM\Delta_f\rho_\nu^2 n^4\alpha_{max}^2)L}} = \sqrt{\frac{1}{193^2 L^2\rho_\nu^2 n^2\alpha_{max}^2}}. \tag{A.3}$$

When the device influence scores are uniform (*i.e,* $\alpha_i = 1/n \ \forall i \in \mathcal{V}$), the step-size is $\gamma = \mathcal{O}\left(\frac{1}{L}\right)$. This mirrors the optimal gradient descent step-size to reach first-order optima $\mathcal{O}(1/L)$ [223].

A.3 Proof of the Main Theorem

Before beginning, the expected gradient $\mathbb{E}_{i_t}\left[G(w_{i_t}^t, \xi_{*,t}^{i_t})\right]$ is defined as,

$$\bar{G}(W^t, \xi^t) := \mathbb{E}_{i_t}\left[G(w_{i_t}^t, \xi_{i_t}^t)\right] = \sum_{i=1}^{n} \alpha_i G(w_{i_t}^t, \xi_{i_t}^t). \tag{A.4}$$

Theorem 1 (Restated)**.** *Under assumptions 1, 2 and 3 (with Algorithm 4), let* $\Delta_f := f(\bar{w}^0) - f(\bar{w}^*)$*, step-size* γ*, total iteration* T*, and average model* $\bar{w}^t$ *be defined as,*

$$\gamma := \frac{\sqrt{Mn^2\Delta_f}}{\sqrt{TL}+\sqrt{M}} \leq \sqrt{\frac{Mn^2\Delta_f}{TL}}, \quad T \geq 193^2 LM\Delta_f\rho_\nu^2 n^4\alpha_{max}^2, \quad \bar{w}^t := \frac{1}{n}\sum_{i=1}^{n} w_i^t.$$

Then, for the output of Algorithm 3, it holds that,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2\sqrt{\Delta_f}}{T} + \frac{2\sqrt{L\Delta_f}\left(1+\frac{1921}{20}\rho_\nu\left(\sigma^2+6\zeta^2\sqrt{M}\right)\right)}{\sqrt{TM}}.$$

Proof. In this theorem, the convergence of the average of *all* local models is characterized. Using the Gradient Lipschitz assumption with Equation 2.15 yields,

$$\begin{aligned} f\Big(\frac{W^{t+1}\mathbf{1_n}}{n}\Big) \leq & f\Big(\frac{W^t\mathbf{1_n}}{n}\Big) + \left\langle \nabla f\Big(\frac{W^t\mathbf{1_n}}{n}\Big), \frac{W^t(C_{i_t}^t-\bar{C}^t)\mathbf{1_n}}{n} - \gamma\frac{G(w_{i_t}^t,\xi_{*,t}^{i_t})\mathbf{1_n}}{n}\right\rangle \\ & + \frac{L}{2}\left\|\frac{W^t(C_{i_t}^t-\bar{C}^t)\mathbf{1_n}}{n} - \gamma\frac{G(w_{i_t}^t,\xi_{*,t}^{i_t})\mathbf{1_n}}{n}\right\|^2. \end{aligned} \tag{A.5}$$

Denote the average of all models as $\bar{w}^t := \frac{W^t\mathbf{1_n}}{n}$. The expectation w.r.t the updating device i_t is,

$$\begin{aligned} f(\bar{w}^{t+1}) \leq & f(\bar{w}^t) + \left\langle \nabla f(\bar{w}^t), W^t(\bar{C}^t-\bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma\bar{G}(W^t,\xi_{*,t})\frac{\mathbf{1_n}}{n}\right\rangle \\ & + \frac{L}{2}\mathbb{E}_{i_t}\left\|W^t(C_{i_t}^t-\bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t,\xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2, \end{aligned} \tag{A.6}$$

$$\begin{aligned} = & f(\bar{w}^t) + \left\langle \nabla f(\bar{w}^t), -\gamma\bar{G}(W^t,\xi_{*,t})\frac{\mathbf{1_n}}{n}\right\rangle \\ & + \frac{L}{2}\mathbb{E}_{i_t}\left\|W^t(C_{i_t}^t-\bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t,\xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2, \end{aligned} \tag{A.7}$$

$$\begin{aligned} = & f(\bar{w}^t) + \left\langle \nabla f(\bar{w}^t), -\frac{\gamma}{Mn}\sum_{i=1}^{n}\sum_{m=1}^{M}\alpha_i\nabla\ell(w_i^t,\xi_{m,t}^i)\right\rangle \\ & + \frac{L}{2}\mathbb{E}_{i_t}\left\|W^t(C_{i_t}^t-\bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t,\xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2, \end{aligned} \tag{A.8}$$

$$\begin{aligned} = & f(\bar{w}^t) - \gamma\left\langle \nabla f(\bar{w}^t), \frac{1}{Mn}\sum_{i=1}^{n}\sum_{m=1}^{M}\alpha_i\nabla\ell(w_i^t,\xi_{m,t}^i)\right\rangle \\ & + \frac{L}{2}\mathbb{E}_{i_t}\left\|W^t(C_{i_t}^t-\bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t,\xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2. \end{aligned} \tag{A.9}$$

Taking the expectation over all local data $\mathbb{E}_{\xi\sim\mathcal{D}_i}$ yields,

$$f(\bar{w}^{t+1}) - f(\bar{w}^t) \leq -\frac{\gamma}{n}\left\langle \nabla f(\bar{w}^t), \sum_{i=1}^{n} \alpha_i \nabla f_i(w_i^t) \right\rangle + \frac{L}{2}\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\| W^t(C_{i_t}^t - \bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t, \xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2. \tag{A.10}$$

By properties of the inner product,

$$f(\bar{w}^{t+1}) - f(\bar{w}^t) \leq -\frac{\gamma}{2n}\left(\|\nabla f(\bar{w}^t)\|^2 + \|\sum_{i=1}^{n} \alpha_i \nabla f_i(w_i^t)\|^2 - \|\nabla f(\bar{w}^t) - \sum_{i=1}^{n} \alpha_i \nabla f_i(w_i^t)\|^2 \right) + \frac{L}{2}\underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\| W^t(C_{i_t}^t - \bar{C}^t)\frac{\mathbf{1_n}}{n} - \gamma G(w_{i_t}^t, \xi_{*,t}^{i_t})\frac{\mathbf{1_n}}{n}\right\|^2}_{:=A}. \tag{A.11}$$

Bounding Term A: Given the update equation, term A can be transformed into,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\| \left(W^t(C_{i_t}^t - \bar{C}^t) - \gamma G(w_{i_t}^t, \xi_{*,t}^{i_t})\right)\frac{\mathbf{1_n}}{n}\right\|^2 = \mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\| \left(W^{t+1} - W^t\bar{C}^t\right)\frac{\mathbf{1_n}}{n}\right\|^2. \tag{A.12}$$

Due to the symmetric and doubly stochastic property of $\bar{C}^t$ this reduces to,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|\bar{w}^{t+1} - \bar{w}^t\right\|^2 = \mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|\tfrac{1}{n}\sum_{i=1}^{n}\left(w_i^{t+1} - w_i^t\right)\right\|^2 = \mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|\tfrac{1}{n}\left(w_{i_t}^{t+1} - w_{i_t}^t\right)\right\|^2, \tag{A.13}$$

$$= \frac{1}{n^2}\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|w_{i_t}^{t+1} - w_{i_t}^t\right\|^2, \tag{A.14}$$

$$\leq \frac{3}{n^2}\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left(\left\|w_{i_t}^{t+1} - \bar{w}^{t+1}\right\|^2 + \left\|\bar{w}^t - w_{i_t}^t\right\|^2 + \left\|\bar{w}^{t+1} - \bar{w}^t\right\|^2 \right). \tag{A.15}$$

Combining like terms yields,

$$(1 - \frac{3}{n^2})\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|\bar{w}^{t+1} - \bar{w}^t\right\|^2 \leq \frac{3}{n^2}\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left(\left\|w_{i_t}^{t+1} - \bar{w}^{t+1}\right\|^2 + \left\|\bar{w}^t - w_{i_t}^t\right\|^2 \right). \tag{A.16}$$

Since $n \geq 2$ (it is assumed at least 2 devices run the algorithm) one finds the following result,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left\|\bar{w}^{t+1}-\bar{w}^t\right\|^2 \leq \frac{3}{(n^2-3)}\mathbb{E}_{\xi\sim\mathcal{D}_i,i_t}\left(\left\|w_{i_t}^{t+1}-\bar{w}^{t+1}\right\|^2+\left\|\bar{w}^t-w_{i_t}^t\right\|^2\right)$$
$$=\frac{3}{(n^2-3)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{i=1}^{n}\alpha_i\left(\left\|\bar{w}^{t+1}-w_i^{t+1}\right\|^2+\left\|\bar{w}^t-w_i^t\right\|^2\right) \quad \text{(A.17)}$$

Thus, Term A has been bounded. Substituting this back into Equation A.11 results in,

$$\leq -\frac{\gamma}{2n}\left(\left\|\nabla f(\bar{w}^t)\right\|^2+\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2-\left\|\nabla f(\bar{w}^t)-\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\right)]$$
$$+\frac{3L}{2(n^2-3)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{i=1}^{n}\alpha_i\left(\left\|\bar{w}^{t+1}-w_i^{t+1}\right\|^2+\left\|\bar{w}^t-w_i^t\right\|^2\right). \quad \text{(A.18)}$$

Taking the sum from $t=0$ to $t=T-1$ yields,

$$f(\bar{w}^T)-f(\bar{w}^0) \leq -\frac{\gamma}{2n}\left(\sum_{t=0}^{T-1}\left\|\nabla f(\bar{w}^t)\right\|^2-\sum_{t=0}^{T-1}\left\|\nabla f(\bar{w}^t)-\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\right.$$
$$\left.+\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\right)$$
$$+\frac{3L}{2(n^2-3)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left(\left\|\bar{w}^{t+1}-w_i^{t+1}\right\|^2+\left\|\bar{w}^t-w_i^t\right\|^2\right). \quad \text{(A.19)}$$

Using the Lipschitz Gradient assumption, the following term is bounded as,

$$\sum_{t=0}^{T-1}\left\|\nabla f(\bar{w}^t)-\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2=\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(\bar{w}^t)-\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2, \quad \text{(A.20)}$$
$$\leq\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i^2\left\|\nabla f_i(\bar{w}^t)-\nabla f_i(w_i^t)\right\|^2, \quad \text{(A.21)}$$
$$\leq L^2\alpha_{max}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^t-w_i^t\right\|^2. \quad \text{(A.22)}$$

Placing this back into Equation A.19 and rearranging yields,

$$f(\bar{w}^T) - f(\bar{w}^0) \leq -\frac{\gamma}{2n}\sum_{t=0}^{T-1}\left\|\nabla f(\bar{w}^t)\right\|^2 - \frac{\gamma}{2n}\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i \nabla f_i(w_i^t)\right\|^2 + \frac{3L}{2(n^2-3)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left(\left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2 + \left\|\bar{w}^t - w_i^t\right\|^2\right) + \frac{\gamma L^2 \alpha_{max}}{2n}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^t - w_i^t\right\|^2. \tag{A.23}$$

Given that $\bar{w}^0 = w_i^0$ for all devices i,

$$\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2 = \sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2 + \sum_{i=1}^{n}\alpha_i\left\|\bar{w}^0 - w_i^0\right\|^2, \tag{A.24}$$

$$= \sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^t - w_i^t\right\|^2 + \sum_{i=1}^{n}\alpha_i\left\|\bar{w}^T - w_i^T\right\|^2, \tag{A.25}$$

$$\geq \sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^t - w_i^t\right\|^2. \tag{A.26}$$

Using the result of Equation A.24 condenses Equation A.23 into,

$$f(\bar{w}^T) - f(\bar{w}^0) \leq -\frac{\gamma}{2n}\sum_{t=0}^{T-1}\left\|\nabla f(\bar{w}^t)\right\|^2 - \frac{\gamma}{2n}\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i \nabla f_i(w_i^t)\right\|^2 + \left(\frac{3L}{(n^2-3)} + \frac{\gamma L^2\alpha_{max}}{2n}\right)\underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2}_{:=B}. \tag{A.27}$$

Bounding Term B. The recursion of this update rule can be written as,

$$W^{t+1} = W^0 - \gamma\sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j})\prod_{q=j+1}^{t}\bar{C}^q - \gamma\sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q - \bar{C}^q). \tag{A.28}$$

The recursion equation for the expected consensus model $\bar{w}^t$ and expected local model for device i can be computed by multiplying by $\frac{\mathbf{1_n}}{n}$ and $\boldsymbol{e}_i$ respectively,

$$\bar{w}^{t+1} = \bar{w}^0 - \gamma \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \frac{\mathbf{1_n}}{n} - \gamma \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q) \frac{\mathbf{1_n}}{n}, \tag{A.29}$$

$$w_i^{t+1} = w_i^0 - \gamma \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} \bar{C}^q \boldsymbol{e}_i - \gamma \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q) \boldsymbol{e}_i. \tag{A.30}$$

Using the recursive equations above transforms the bound on term B,

$$\mathbb{E}_{\xi \sim \mathcal{D}_i} \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_i \left\| \bar{w}^{t+1} - w_i^{t+1} \right\|^2$$

$$= \gamma^2 \mathbb{E}_{\xi \sim \mathcal{D}_i} \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_i \left\| - \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \left(\tfrac{\mathbf{1_n}}{n} - \prod_{q=j+1}^{t} \bar{C}^q \boldsymbol{e}_i \right) - \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\tfrac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2 \tag{A.31}$$

$$\leq 2\gamma^2 \mathbb{E}_{\xi \sim \mathcal{D}_i} \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_i \Bigg(\left\| \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \left(\frac{\mathbf{1_n}}{n} - \prod_{q=j+1}^{t} \bar{C}^q \boldsymbol{e}_i \right) \right\|^2$$

$$+ \left\| \sum_{j=0}^{t} G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2 \Bigg), \tag{A.32}$$

$$= 2\gamma^2 \mathbb{E}_{\xi \sim \mathcal{D}_i} \sum_{t=0}^{T-1} \sum_{i=1}^{n} \alpha_i \Bigg(\underbrace{\sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \left(\frac{\mathbf{1_n}}{n} - \prod_{q=j+1}^{t} \bar{C}^q \boldsymbol{e}_i \right) \right\|^2}_{:=B_1}$$

$$+ \underbrace{\sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2}_{:=B_3}$$

$$+ \underbrace{2 \sum_{j=0}^{t} \sum_{j'=j+1}^{t} \langle G(w_{i_j}^j, \xi_{*,j}^{i_j}) (\frac{\mathbf{1_n}}{n} - \prod_{q=j+1}^{t} \bar{C}^q \boldsymbol{e}_i), G(w_{i_{j'}}^{j'}, \xi_{*,j'}^{i_{j'}}) (\frac{\mathbf{1_n}}{n} - \prod_{q=j'+1}^{t} \bar{C}^q \boldsymbol{e}_i) \rangle}_{:=B_2}$$

$$+ \underbrace{2 \sum_{j=0}^{t} \sum_{j'=j+1}^{t} \langle G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i), G(w_{i_{j'}}^{j'}, \xi_{*,j'}^{i_{j'}}) \prod_{q=j'+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \rangle}_{:=B_4} \Bigg). \tag{A.33}$$

Bounding Term B_1. Using Lemma 4,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\left(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\right)\right\|^2$$

$$\leq \mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\right\|^2\left\|\left(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\right)\right\|^2, \tag{A.34}$$

$$= \mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_{i_j}^j;\xi_{m,j}^{i_j})\right\|^2\left\|\left(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\right)\right\|^2, \tag{A.35}$$

$$\leq \mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_{i_j}^j;\xi_{m,j}^{i_j})\right\|^2(\frac{n-1}{n})\rho^{t-j}. \tag{A.36}$$

Using Lemma 6, the equation above becomes,

$$= 2\sum_{j=0}^{t}\left(\frac{\sigma^2}{M}+\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2\right)(\frac{n-1}{n})\rho^{t-j}, \tag{A.37}$$

$$= 2\left(\sum_{j=0}^{t}\frac{\sigma^2}{M}(\frac{n-1}{n})\rho^{t-j}+\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2(\frac{n-1}{n})\rho^{t-j}\right), \tag{A.38}$$

$$\leq \frac{2(n-1)\sigma^2}{(1-\rho)Mn}+2\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2(\frac{n-1}{n})\rho^{t-j}. \tag{A.39}$$

Taking the expectation over worker i_j yields the desired bound,

$$\mathbb{E}_{i_j}\left[\frac{2(n-1)\sigma^2}{(1-\rho)Mn}+2\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2(\frac{n-1}{n})\rho^{t-j}\right]$$

$$= \frac{2(n-1)\sigma^2}{(1-\rho)Mn}+\frac{2(n-1)}{n}\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2\rho^{t-j}. \tag{A.40}$$

Bounding Term B_2. Using Lemma 4,

$$2\sum_{j=0}^{t}\sum_{j'=j+1}^{t}\langle G(w_{i_j}^{j},\xi_{*,j}^{i_j})\big(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\big),G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\big(\frac{\mathbf{1_n}}{n}-\prod_{q=j'+1}^{t}\bar{C}^q\boldsymbol{e}_i\big)\rangle$$

$$=2\textstyle\sum_{j=0}^{t}\sum_{j'=j+1}^{t}\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|\left\|\big(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\big)\right\|\left\|G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\right\|\left\|\big(\frac{\mathbf{1_n}}{n}-\prod_{q=j'+1}^{t}\bar{C}^q\boldsymbol{e}_i\big)\right\|. \quad \text{(A.41)}$$

For any $\alpha_{j,j'}>0$ one finds,

$$\leq 2\textstyle\sum_{j=0}^{t}\sum_{j'=j+1}^{t}\left(\frac{\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2\left\|G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\right\|^2}{2\alpha_{j,j'}}+\frac{\alpha_{j,j'}\left\|\left(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i\right)\right\|^2\left\|\left(\frac{\mathbf{1_n}}{n}-\prod_{q=j'+1}^{t}\bar{C}^q\boldsymbol{e}_i\right)\right\|^2}{2}\right), \quad \text{(A.42)}$$

$$\leq\sum_{j\neq j'}^{t}\left(\frac{\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2\left\|G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\right\|^2}{2\alpha_{j,j'}}+\frac{\alpha_{j,j'}\rho^{t-\min\{j,j'\}}}{2}(\frac{n-1}{n})^2\right),\alpha_{j,j'}=\alpha_{j',j}. \quad \text{(A.43)}$$

Applying inequality of arithmetic & geometric means to the last term above, one can let $\alpha_{j,j'}>0$,

$$\leq\frac{n-1}{n}\sum_{j\neq j'}^{t}\left(\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|\left\|G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\right\|\rho^{\frac{t-\min\{j,j'\}}{2}}\right), \quad \text{(A.44)}$$

$$\leq\frac{n-1}{n}\sum_{j\neq j'}^{t}\left(\frac{\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2+\left\|G(w_{i_{j'}}^{j'},\xi_{*,j'}^{i_{j'}})\right\|^2}{2}\rho^{\frac{t-\min\{j,j'\}}{2}}\right), \quad \text{(A.45)}$$

$$=\frac{n-1}{n}\sum_{j\neq j'}^{t}\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2\rho^{\frac{t-\min\{j,j'\}}{2}}, \quad \text{(A.46)}$$

$$=\frac{n-1}{n}\sum_{j=0}^{t}\sum_{j'=j+1}^{t}\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2\rho^{\frac{t-j}{2}}, \quad \text{(A.47)}$$

$$=\frac{n-1}{n}\sum_{j=0}^{t}\left\|G(w_{i_j}^{j},\xi_{*,j}^{i_j})\right\|^2 2(t-j)\rho^{\frac{t-j}{2}}, \quad \text{(A.48)}$$

$$=\frac{n-1}{n}\sum_{j=0}^{t}2(t-j)\rho^{\frac{t-j}{2}}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_{i_j}^{j};\xi_{m,j}^{i_j})\right\|^2. \quad \text{(A.49)}$$

Using Lemma 6 (and the expectation $\mathbb{E}_{\xi\sim\mathcal{D}_i}$ that was omitted above but is present) yields,

$$= \frac{2(n-1)}{n}\sum_{j=0}^{t} 2(t-j)\rho^{\frac{t-j}{2}}\left(\frac{\sigma^2}{M} + \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2\right), \tag{A.50}$$

$$\leq \frac{4(n-1)\sqrt{\rho}\sigma^2}{Mn(1-\sqrt{\rho})^2} + \frac{2(n-1)}{n}\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 2(t-j)\rho^{\frac{t-j}{2}}. \tag{A.51}$$

Taking the expectation over worker i_j yields the desired bound,

$$\mathbb{E}_{i_j}\left[\frac{4(n-1)\sqrt{\rho}\sigma^2}{Mn(1-\sqrt{\rho})^2} + \frac{2(n-1)}{n}\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 2(t-j)\rho^{\frac{t-j}{2}}\right]$$

$$= \frac{4(n-1)\sqrt{\rho}\sigma^2}{Mn(1-\sqrt{\rho})^2} + \frac{2(n-1)}{n}\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 2(t-j)\rho^{\frac{t-j}{2}}. \tag{A.52}$$

Bounding Term B_3.

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2,$$

$$=\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal+\phi 1^\intercal-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2, \tag{A.53}$$

$$=\mathbb{E}_{\xi\sim\mathcal{D}_i}\textstyle\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\left[\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)-\prod_{q=j+1}^{t}(\bar{C}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right]\right\|^2, \tag{A.54}$$

$$\leq 2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2,$$

$$+\,2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(\bar{C}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2. \tag{A.55}$$

Due to the structure of $\phi 1^\intercal$, multiplying this matrix by $\frac{\mathbf{1_n}}{n}$ or $\boldsymbol{e}_i$ yields the same result. Using this, as well as the double stochasticity of $\bar{W}$, one finds,

$$= 2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left[\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\tfrac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2+\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}\bar{C}^q(\tfrac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\right], \quad \text{(A.56)}$$

$$= 2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left[\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\tfrac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2+\underbrace{\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i)\right\|^2}_{=B_1}\right]. \quad \text{(A.57)}$$

Using the result of Lemma 5, as the communication graph $\mathcal{G}$ is uniformly strongly connected and $[C_{i_t}^t]_{i,i}\geq 1/n$ by construction,

$$|\big[\prod_{q=j+1}^{t}W_{i_q}^q-\phi 1^\intercal\big]_{h,k}|\leq 4\nu^{t-j-1}\;\forall\; h,k. \quad \text{(A.58)}$$

Using this result,

$$|\big[\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\big]_h|\leq 8(\frac{n-1}{n})\nu^{t-j-1}\;\forall\; h. \quad \text{(A.59)}$$

One can remove the -1 exponent by doubling the constant out front,

$$|\big[\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\big]_h|\leq 16(\frac{n-1}{n})\nu^{t-j}\;\forall\; h. \quad \text{(A.60)}$$

Finally, using the fact that $G(w_{i_j}^j,\xi_{*,j}^{i_j})$ is all zeros except for one column, the i_j-th column, yields the desired result,

$$\begin{aligned}&\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(W_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\\&\leq\sum_{j=0}^{t}256(\frac{n-1}{n})^2\nu^{2(t-j)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_{i_j}^j,\xi_{*,j}^{i_j})\right\|^2.\end{aligned} \quad \text{(A.61)}$$

Utilizing Lemma 6 yields,

$$\leq 256(\frac{n-1}{n})^2 \sum_{j=0}^{t} \nu^{2(t-j)} \left(\frac{\sigma^2}{M} + \left\| \nabla f_{i_j}(w_{i_j}^t) \right\|^2 \right). \tag{A.62}$$

By properties of geometric series, and taking the expectation over worker i_j,

$$\leq \frac{256\sigma^2}{(1-\nu^2)M}(\frac{n-1}{n})^2 + 256(\frac{n-1}{n})^2 \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\| \nabla f_{i_j}(w_{i_j}^t) \right\|^2 \nu^{2(t-j)}. \tag{A.63}$$

Using the bound of B_1 in the main proof above, one arrives at the final bound of B_3,

$$\begin{aligned}
&\mathbb{E}_{\xi\sim\mathcal{D}_i} \sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2 \\
&\leq \frac{512\sigma^2}{(1-\nu^2)M}(\frac{n-1}{n})^2 + \frac{4(n-1)\sigma^2}{(1-\rho)Mn} + 512(\frac{n-1}{n})^2 \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\| \nabla f_{i_j}(w_{i_j}^t) \right\|^2 \nu^{2(t-j)} \\
&\quad + \frac{4(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\| \nabla f_{i_j}(w_{i_j}^j) \right\|^2 \rho^{t-j}, \qquad \text{(A.64)} \\
&\leq \frac{4(n-1)\sigma^2}{Mn} \left(\frac{1}{(1-\rho)} + \frac{128}{(1-\nu^2)} \right) \\
&\quad + \frac{4(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\| \nabla f_{i_j}(w_{i_j}^j) \right\|^2 \left(\rho^{t-j} + 128\nu^{2(t-j)} \right). \qquad \text{(A.65)}
\end{aligned}$$

Bounding Term B_4. Following similar steps as bounding Term B_2,

$$\begin{aligned}
&2\mathbb{E}_{\xi\sim\mathcal{D}_i} \textstyle\sum_{j=0}^{t} \sum_{j'=j+1}^{t} \langle G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i), G(w_{i_{j'}}^{j'}, \xi_{*,j'}^{i_{j'}}) \prod_{q=j'+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \rangle, \\
&\leq 2\mathbb{E}_{\xi\sim\mathcal{D}_i} \sum_{j=0}^{t} \sum_{j'=j+1}^{t} \left(\frac{\left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2}{2} \right. \\
&\quad \left. + \frac{\left\| G(w_{i_{j'}}^{j'}, \xi_{*,j'}^{i_{j'}}) \prod_{q=j'+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2}{2} \right), \qquad \text{(A.66)}
\end{aligned}$$

$$\leq 2 \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i} \sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (W_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2}_{=B_3}. \tag{A.67}$$

Once again, one can use the proof of Lemma 1 to bound this result.

Finishing Bound of Term $\boldsymbol{B}$. Putting all terms together, Term B is bounded above by,

$$\begin{aligned}
&\sum_{t=0}^{T-1}\sum_{i=1}^{n} \alpha_i \left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2 \\
\leq& 2\gamma^2 \sum_{t=0}^{T-1}\sum_{i=1}^{n} \alpha_i \Bigg(\frac{2(n-1)\sigma^2}{(1-\rho)Mn} + \frac{2(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 \rho^{t-j} \\
&+ \frac{4(n-1)\sqrt{\rho}\sigma^2}{Mn(1-\sqrt{\rho})^2} + \frac{2(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 2(t-j)\rho^{\frac{t-j}{2}} \\
&+ \frac{12(n-1)\sigma^2}{Mn} \left(\frac{1}{(1-\rho)} + \frac{128}{(1-\nu^2)} \right) \\
&+ \frac{12(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 \left(\rho^{t-j} + 128\nu^{2(t-j)} \right) \Bigg).
\end{aligned} \tag{A.68}$$

Simplifying results in,

$$\begin{aligned}
\leq& 2\gamma^2 \sum_{t=0}^{T-1}\sum_{i=1}^{n} \alpha_i \Bigg(\frac{4(n-1)\sigma^2}{Mn} \left(\frac{7}{2(1-\rho)} + \frac{\sqrt{\rho}}{(1-\sqrt{\rho})^2} + \frac{384}{(1-\nu^2)} \right) \\
&+ \frac{4(n-1)}{n} \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 \left(\frac{7}{2}\rho^{t-j} + (t-j)\rho^{\frac{t-j}{2}} + 384\nu^{2(t-j)} \right) \Bigg),
\end{aligned} \tag{A.69}$$

$$\begin{aligned}
\leq& \frac{8\gamma^2\sigma^2 T}{M} \Bigg(\underbrace{\frac{n-1}{n} \left(\frac{7}{2(1-\rho)} + \frac{\sqrt{\rho}}{(1-\sqrt{\rho})^2} + \frac{384}{(1-\nu^2)} \right)}_{:=\rho_\nu} \Bigg) \\
&+ \frac{8(n-1)\gamma^2}{n} \sum_{t=0}^{T-1}\sum_{i=1}^{n} \alpha_i \sum_{j=0}^{t} \mathbb{E}_{i_j} \left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2 \left(\frac{7}{2}\rho^{t-j} + (t-j)\rho^{\frac{t-j}{2}} + 384\nu^{2(t-j)} \right)
\end{aligned} \tag{A.70}$$

Using Lemma 7 results in,

$$
\begin{aligned}
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + \frac{8(n-1)\gamma^2}{n}\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\sum_{j=0}^{t}\left(2\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^j)\right\|^2\right. \\
& \left. + 12L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^j - w_i^j\right\|^2 + 6\zeta^2\right)\left(\frac{7}{2}\rho^{t-j} + (t-j)\rho^{\frac{t-j}{2}} + 384\nu^{2(t-j)}\right), \qquad \text{(A.71)} \\
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + \frac{16(n-1)\gamma^2}{n}\sum_{t=0}^{T-1}\sum_{j=0}^{t}\left(\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^j)\right\|^2\right. \\
& \left. + 6L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^j - w_i^j\right\|^2 + 3\zeta^2\right)\left(\frac{7}{2}\rho^{t-j} + (t-j)\rho^{\frac{t-j}{2}} + 384\nu^{2(t-j)}\right), \qquad \text{(A.72)} \\
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + \frac{16(n-1)\gamma^2}{n}\sum_{j=0}^{T-1}\sum_{t=j+1}^{\infty}\left(\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^j)\right\|^2\right. \\
& \left. + 6L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^j - w_i^j\right\|^2 + 3\zeta^2\right)\left(\frac{7}{2}\rho^{t-j} + (t-j)\rho^{\frac{t-j}{2}} + 384\nu^{2(t-j)}\right), \qquad \text{(A.73)} \\
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + \frac{16(n-1)\gamma^2}{n}\sum_{j=0}^{T-1}\left(\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^j)\right\|^2\right. \\
& \left. + 6L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^j - w_i^j\right\|^2 + 3\zeta^2\right)\left(\sum_{h=0}^{\infty}\frac{7}{2}\rho^h + h\rho^{\frac{h}{2}} + 384\nu^{2h}\right), \qquad \text{(A.74)} \\
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + 16\rho_\nu\gamma^2\sum_{j=0}^{T-1}\left(\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^j)\right\|^2 + 6L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^j - w_i^j\right\|^2 + 3\zeta^2\right). \qquad \text{(A.75)}
\end{aligned}
$$

Using Equation A.24 ends with,

$$
\begin{aligned}
\leq & \frac{8\gamma^2\sigma^2 T\rho_\nu}{M} + 48T\rho_\nu\gamma^2\zeta^2 + 16\rho_\nu\gamma^2\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2 \\
& + 96L^2\rho_\nu\gamma^2\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1} - w_i^{t+1}\right\|^2. \qquad \text{(A.76)}
\end{aligned}
$$

Now subtract the final term on the right hand side from both sides,

$$\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1}-w_i^{t+1}\right\|^2\left(\underbrace{1-96L^2\rho_\nu\gamma^2}_{:=z}\right)\leq\frac{8\gamma^2\sigma^2T\rho_\nu}{M}+48T\rho_\nu\gamma^2\zeta^2 + 16\rho_\nu\gamma^2\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2. \quad \text{(A.77)}$$

By Lemma 8, one can divide z from both sides,

$$\textstyle\sum_{t=0}^{T-1}\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^{t+1}-w_i^{t+1}\right\|^2\leq\frac{8\gamma^2\sigma^2T\rho_\nu}{Mz}+\frac{48T\rho_\nu\gamma^2\zeta^2}{z}+\frac{16\rho_\nu\gamma^2}{z}\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2. \quad \text{(A.78)}$$

Finishing Bound of Term $\boldsymbol{A}$. Substituting the bound of B above into Equation A.27 yields,

$$\begin{aligned}f(\bar{w}^T)-f(\bar{w}^0)\leq&-\frac{\gamma}{2n}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2-\frac{\gamma}{2n}\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\\&+\Bigg(\underbrace{\frac{3L}{(n^2-3)}+\frac{\gamma L^2\alpha_{max}}{2n}}_{:=\phi}\Bigg)\Bigg(\frac{8\gamma^2\sigma^2T\rho_\nu}{Mz}+\frac{48T\rho_\nu\gamma^2\zeta^2}{z}\\&+\frac{16\rho_\nu\gamma^2}{z}\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\Bigg). \quad \text{(A.79)}\end{aligned}$$

Rearranging terms simplifies the inequality above to

$$\begin{aligned}f(\bar{w}^T)-f(\bar{w}^0)\leq&-\frac{\gamma}{2n}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2+\frac{\gamma}{2n}\left(\frac{32\rho_\nu\phi\gamma n}{z}-1\right)\sum_{t=0}^{T-1}\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2\\&+\frac{8\phi\gamma^2\sigma^2T\rho_\nu}{Mz}+\frac{48\phi T\rho_\nu\gamma^2\zeta^2}{z}. \quad \text{(A.80)}\end{aligned}$$

By Lemma 10, $(1-\frac{32\rho_\nu\phi\gamma n}{z})\geq 0$. Therefore, the 2nd term of the right hand side can be removed.

$$f(\bar{w}^T) - f(\bar{w}^0) \leq -\frac{\gamma}{2n}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 + \frac{8\phi\gamma^2\sigma^2 T\rho_\nu}{Mz} + \frac{48\phi T\rho_\nu\gamma^2\zeta^2}{z}. \tag{A.81}$$

Rearranging the inequality above and dividing by T yields,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2n\big(f(\bar{w}^0) - f(\bar{w}^T)\big)}{T\gamma} + \frac{16n\phi\gamma\sigma^2\rho_\nu}{Mz} + \frac{96n\phi\rho_\nu\gamma\zeta^2}{z} \tag{A.82}$$

$$= \frac{2n\big(f(\bar{w}^0) - f(\bar{w}^T)\big)}{T\gamma} + \frac{16\gamma\phi n\rho_\nu}{z}\big(\frac{\sigma^2}{M} + 6\zeta^2\big). \tag{A.83}$$

From Lemmas 8 and 9, the inequality above becomes,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2n\big(f(\bar{w}^0) - f(\bar{w}^T)\big)}{T\gamma} + \frac{\frac{1921}{10}L\gamma\rho_\nu}{n}\big(\frac{\sigma^2}{M} + 6\zeta^2\big). \tag{A.84}$$

Substituting in the defined step-size γ (as well as its bound) yields,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2n\big(f(\bar{w}^0) - f(\bar{w}^T)\big)}{T}\left(\frac{\sqrt{TL}+\sqrt{M}}{\sqrt{Mn^2\Delta_f}}\right) + \frac{\frac{1921}{10}L}{n}\left(\frac{\sqrt{Mn^2\Delta_f}}{\sqrt{TL}}\right)\rho_\nu\big(\frac{\sigma^2}{M} + 6\zeta^2\big) \tag{A.85}$$

$$= \frac{2\sqrt{\Delta_f}}{T} + \frac{2\sqrt{L\Delta_f}}{\sqrt{TM}} + \frac{\frac{1921}{10}\sqrt{L\Delta_f}\rho_\nu\big(\sigma^2 + 6\zeta^2\sqrt{M}\big)}{\sqrt{TM}}. \tag{A.86}$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f(\bar{w}^t)\right\|^2 \leq \frac{2\sqrt{\Delta_f}}{T} + \frac{2\sqrt{L\Delta_f}\left(1 + \frac{1921}{20}\rho_\nu\big(\sigma^2 + 6\zeta^2\sqrt{M}\big)\right)}{\sqrt{TM}}. \tag{A.87}$$

□

A.4 Supplemental Lemmas

Lemma 4 (From Lemma 3 in Lian et al. 2018 [31]). *Let C^t be a symmetric doubly stochastic matrix for each iteration t. Then,*

$$\left\| \frac{\mathbf{1}_n}{n} - \prod_{t=1}^{T} C^t e_i \right\|^2 \leq \frac{n-1}{n} \rho^T, \ \forall T \geq 0.$$

Lemma 5 (From Corollary 2 in Nedic and Olshevsky 2014 [224]). *Let the communication graph $\mathcal{G}$ be uniformly strongly connected (otherwise known as B-strongly-connected for some integer $B > 0$), and $A(t) \in \mathbb{R}^{n \times n}$ be a column stochastic matrix with $[A(t)]_{i,i} \geq 1/n \ \forall i, t$. Define the product of matrices $A(t)$ through $A(s)$ (for $t \geq s \geq 0$) as $A(t:s) := A(t) \dots A(s)$. Then, there exists a stochastic vector $\phi(t) \in \mathbb{R}^n$ such that $|[A(t:s)]_{i,j} - \phi_i(t)| \leq C\nu^{t-s}$ will always hold for the following values of C and ν,*

$$C = 4, \quad \nu = (1 - 1/n^{nB})^{1/B} < 1.$$

A.5 Lemma Proofs

A.5.1 Proof of Lemma 1

Lemma 1 (Restated). *Following Algorithm 4, the product of the difference between the expected and actual device communication matrices is bounded by,*

$$\mathbb{E} \sum_{j=0}^{t} \left\| G(w_{i_j}^j, \xi_{*,j}^{i_j}) \prod_{q=j+1}^{t} (C_{i_q}^q - \bar{C}^q)(\frac{\mathbf{1_n}}{n} - \boldsymbol{e}_i) \right\|^2 = \mathcal{O}\Big(\frac{\sigma^2}{M} + \mathbb{E} \sum_{j=0}^{t} \|\nabla f_{i_j}(w_{i_j}^j)\|^2 \Big).$$

Proof. These steps are shown in the Main Theorem proof. Due to the symmetry and double stochasticity of $\bar{C}$,

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2 \tag{A.88}$$

$$=\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal+\phi 1^\intercal-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2, \tag{A.89}$$

$$=\mathbb{E}_{\xi\sim\mathcal{D}_i}\textstyle\sum_{j=0}^{t}\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\left[\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)-\prod_{q=j+1}^{t}(\bar{C}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right]\right\|^2, \tag{A.90}$$

$$\leq 2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2$$

$$+2\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(\bar{C}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2. \tag{A.91}$$

Due to the structure of $\phi 1^\intercal$, multiplying this matrix by $\frac{\mathbf{1_n}}{n}$ or $\boldsymbol{e}_i$ yields the same result. Using this, as well as the double stochasticity of $\bar{C}$,

$$=2\mathbb{E}_{\xi\sim\mathcal{D}_i}\textstyle\sum_{j=0}^{t}\left[\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2+\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}\bar{C}^q(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\right], \tag{A.92}$$

$$=2\mathbb{E}_{\xi\sim\mathcal{D}_i}\textstyle\sum_{j=0}^{t}\left[\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2+\underbrace{\left\|G(w_{ij}^j,\xi_{*,j}^{i_j})(\frac{\mathbf{1_n}}{n}-\prod_{q=j+1}^{t}\bar{C}^q\boldsymbol{e}_i)\right\|^2}_{=B_1}\right]. \tag{A.93}$$

Using the result of Lemma 5, as the communication graph $\mathcal{G}$ is uniformly strongly connected and $[C_{i_t}^t]_{i,i}\geq 1/n$ by construction,

$$|\big[\prod_{q=j+1}^{t}C_{i_q}^q-\phi 1^\intercal\big]_{h,k}|\leq 4\nu^{t-j-1}\;\forall\;h,k. \tag{A.94}$$

Using this result,

$$|\big[\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\big]_h|\leq 8(\frac{n-1}{n})\nu^{t-j-1}\;\forall\; h. \tag{A.95}$$

One can remove the -1 exponent by doubling the constant out front,

$$|\big[\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\big]_h|\leq 16(\frac{n-1}{n})\nu^{t-j}\;\forall\; h. \tag{A.96}$$

Since $G(w_{i_j}^j,\xi_{*,j}^{i_j})$ is all zeros except for one column, the i_j-th column,

$$\begin{aligned}&\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\phi 1^\intercal)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\\&\leq\sum_{j=0}^{t}256(\frac{n-1}{n})^2\nu^{2(t-j)}\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_{i_j}^j,\xi_{*,j}^{i_j})\right\|^2.\end{aligned} \tag{A.97}$$

Utilizing Lemma 6 yields,

$$\leq 256(\frac{n-1}{n})^2\sum_{j=0}^{t}\nu^{2(t-j)}\left(\frac{\sigma^2}{M}+\left\|\nabla f_{i_j}(w_{i_j}^t)\right\|^2\right). \tag{A.98}$$

By properties of geometric series, and taking the expectation over worker i_j,

$$\leq\frac{256\sigma^2}{(1-\nu^2)M}(\frac{n-1}{n})^2+256(\frac{n-1}{n})^2\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^t)\right\|^2\nu^{2(t-j)}. \tag{A.99}$$

Using the bound Equation A.40 (term B_1) in the main proof above, one arrives at the final bound,

$$\begin{aligned}
&\mathbb{E}_{\xi\sim\mathcal{D}_i}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\\
\leq&\frac{512\sigma^2}{(1-\nu^2)M}(\frac{n-1}{n})^2+\frac{4(n-1)\sigma^2}{(1-\rho)Mn}+512(\frac{n-1}{n})^2\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^t)\right\|^2\nu^{2(t-j)}\\
&+\frac{4(n-1)}{n}\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2\rho^{t-j}, \qquad \text{(A.100)}\\
\leq&\frac{4(n-1)\sigma^2}{Mn}\left(\frac{1}{(1-\rho)}+\frac{128}{(1-\nu^2)}\right)\\
&+\frac{4(n-1)}{n}\sum_{j=0}^{t}\mathbb{E}_{i_j}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2\left(\rho^{t-j}+128\nu^{2(t-j)}\right). \qquad \text{(A.101)}
\end{aligned}$$

Thus, the desired result is derived,

$$\mathbb{E}\sum_{j=0}^{t}\left\|G(w_{i_j}^j,\xi_{*,j}^{i_j})\prod_{q=j+1}^{t}(C_{i_q}^q-\bar{C}^q)(\frac{\mathbf{1_n}}{n}-\boldsymbol{e}_i)\right\|^2\leq\mathcal{O}(\frac{\sigma^2}{M}+\mathbb{E}\sum_{j=0}^{t}\left\|\nabla f_{i_j}(w_{i_j}^j)\right\|^2). \qquad \text{(A.102)}$$

□

A.5.2 Proof of Lemma 6

Lemma 6. *Under Assumption 1, the following inequality holds,*

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_i^t,\xi_{m,t}^i)\right\|^2\leq\frac{\sigma^2}{M}+\left\|\nabla f_i(w_i^t)\right\|^2.$$

Proof.

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla\ell(w_i^t,\xi_{m,t}^i)\right\|^2 = \tfrac{1}{M^2}\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\sum_{m=1}^{M}\nabla\ell(w_i^t,\xi_{m,t}^i)-\nabla f_i(w_i^t)+\nabla f_i(w_i^t)\right\|^2, \tag{A.103}$$

$$=\frac{1}{M^2}\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\sum_{m=1}^{M}\nabla\ell(w_i^t,\xi_{m,t}^i)-\nabla f_i(w_i^t)\right\|+\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\sum_{m=1}^{M}\nabla f_i(w_i^t)\right\|^2$$

$$+\frac{2}{M^2}\mathbb{E}_{\xi\sim\mathcal{D}_i}\Big\langle\sum_{m=1}^{M}\big(\nabla\ell(w_i^t,\xi_{m,t}^i)-\nabla f_i(w_i^t)\big),\sum_{m=1}^{M}\nabla f_i(w_i^t)\Big\rangle, \tag{A.104}$$

$$=\frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}_{\xi\sim\mathcal{D}_i}\left\|\nabla\ell(w_i^t,\xi_{m,t}^i)-\nabla f_i(w_i^t)\right\|+M^2\left\|\nabla f_i(w_i^t)\right\|^2, \tag{A.105}$$

$$\leq\frac{\sigma^2}{M}+\left\|\nabla f_i(w_i^t)\right\|^2. \tag{A.106}$$

□

A.5.3 Proof of Lemma 7

Lemma 7. *Under Assumption 1, the following inequality holds,*

$$\mathbb{E}_i\left\|\nabla f_i(w_i^t)\right\|^2\leq 2\left\|\sum_{i=1}^{n}\alpha_i\nabla f_i(w_i^t)\right\|^2+12L^2\sum_{i=1}^{n}\alpha_i\left\|\bar{w}^t-w_i^t\right\|^2+6\zeta^2.$$

Proof.

$$\mathbb{E}_i\left\|\nabla f_i(w_i^t)\right\|^2=\mathbb{E}_i\left\|\nabla f_i(w_i^t)-\sum_{i'=1}^{n}\alpha_{i'}\nabla f_i(w_{i'}^t)+\sum_{i'=1}^{n}\alpha_{i'}\nabla f_i(w_{i'}^t)\right\|^2, \tag{A.107}$$

$$\leq 2\mathbb{E}_i\left(\left\|\nabla f_i(w_i^t)-\sum_{j=1}^{n}\alpha_j\nabla f_j(w_j^t)\right\|^2+\left\|\sum_{j=1}^{n}\alpha_j\nabla f_j(w_j^t)\right\|^2\right). \tag{A.108}$$

$$
\begin{aligned}
&\mathbb{E}_i \left\| \nabla f_i(w_i^t) - \sum_{j=1}^n \alpha_j \nabla f_j(w_j^t) \right\|^2 \\
=&3\mathbb{E}_i \Bigg(\left\| \nabla f_i(w_i^t) - \nabla f_i(\frac{W^t \mathbf{1_n}}{n}) \right\|^2 + \left\| \sum_{j=1}^n \alpha_j \nabla f_j(\frac{W^t \mathbf{1_n}}{n}) - \sum_{j=1}^n \alpha_j \nabla f_j(w_j^t) \right\|^2 \\
&+ \left\| \nabla f_i(\frac{W^t \mathbf{1_n}}{n}) - \sum_{j=1}^n \alpha_j \nabla f_j(\frac{W^t \mathbf{1_n}}{n}) \right\|^2 \Bigg), \qquad \text{(A.109)} \\
\leq&3\mathbb{E}_i \Bigg(L^2 \left\| w_i^t - \bar{w}^t \right\|^2 + \textstyle\sum_{j=1}^n \alpha_j \left\| \nabla f_j(\bar{w}^t) - \nabla f_j(w_j^t) \right\|^2 + \left\| \nabla f_i(\bar{w}^t) - \nabla f(\bar{w}^t) \right\|^2 \Bigg), \qquad \text{(A.110)} \\
\leq&3\mathbb{E}_i \Bigg(L^2 \left\| w_i^t - \bar{w}^t \right\|^2 + L^2 \sum_{j=1}^n \alpha_j \left\| \bar{w}^t - w_j^t \right\|^2 \Bigg) + 3\zeta^2, \qquad \text{(A.111)} \\
=&6L^2 \sum_{i=1}^n \alpha_i \left\| \bar{w}^t - w_i^t \right\|^2 + 3\zeta^2. \qquad \text{(A.112)}
\end{aligned}
$$

Combining all terms yields the final result,

$$
\mathbb{E}_i \left\| \nabla f_i(w_i^t) \right\|^2 \leq 2 \left\| \sum_{i=1}^n \alpha_i \nabla f_i(w_i^t) \right\|^2 + 12L^2 \sum_{i=1}^n \alpha_i \left\| \bar{w}^t - w_i^t \right\|^2 + 6\zeta^2. \qquad \text{(A.113)}
$$

□

A.5.4 Proof of Lemma 8

Lemma 8. *Given step-size γ and total iterations T in Theorem 1, z is bounded by,*

$$
1 > z := 1 - 96L^2 \rho_\nu \gamma^2 \geq 1 - \frac{384}{193^2(775)}.
$$

Proof. It is trivial to see that $z < 1$. Now the lower bound of z is determined given $n \geq 2$,

$\alpha_{max} \geq 1/n$, and $\rho_\nu \geq 775/4$

$$z = 1 - 96L^2\rho_\nu\gamma^2 = 1 - 96L^2\rho_\nu(\frac{Mn^2\Delta_f}{TL}) = 1 - 96L^2\rho_\nu(\frac{1}{193^2L^2\rho_\nu^2n^2\alpha_{max}^2}), \tag{A.114}$$

$$= 1 - \frac{96}{193^2\rho_\nu n^2\alpha_{max}^2} \leq 1 - \frac{384}{193^2(775)}. \tag{A.115}$$

□

A.5.5 Proof of Lemma 9

Lemma 9. *Given step-size γ and total iterations T in Theorem 1, ϕ is bounded by,*

$$\phi := \frac{3L}{(n^2-3)} + \frac{L^2\gamma\alpha_{max}}{2n} \leq \frac{L}{n^2}(12 + \frac{2}{775(193)}).$$

Proof. Given $n \geq 2$ and $\rho_\nu \geq 775/4$, and the definition of γ and T,

$$\phi = (\frac{3L}{(n^2-3)} + \frac{L^2\gamma\alpha_{max}}{2n}) = \frac{L}{n^2}(\frac{3}{(1-3/n^2)} + \frac{L\gamma n\alpha_{max}}{2}), \tag{A.116}$$

$$\leq \frac{L}{n^2}(\frac{3}{(1-3/n^2)} + \frac{Ln\alpha_{max}}{2}(\frac{\sqrt{Mn^2\Delta_f}}{\sqrt{TL}})), \tag{A.117}$$

$$\leq \frac{L}{n^2}(\frac{3}{(1-3/n^2)} + \frac{Ln\alpha_{max}}{2}(\frac{1}{193L\rho_\nu n\alpha_{max}})) \leq \frac{L}{n^2}(\frac{3}{(1-3/4)} + \frac{1}{386\rho_\nu}), \tag{A.118}$$

$$= \frac{L}{n^2}(12 + \frac{2}{775(193)}). \tag{A.119}$$

□

A.5.6 Proof of Lemma 10

Lemma 10. *Given step-size γ and total iterations T in Theorem 1, the following bound exists,*

$$\big(1 - \frac{32\rho_\nu\phi\gamma n}{z}\big) \geq 0.$$

Proof. Given $n \geq 2$, $\alpha_{max} \geq 1/n$ Lemma 8, and Lemma 9,

$$1 - 1 - \frac{32\rho_\nu\phi\gamma n}{z} = 1 - \frac{32\rho_\nu\phi n}{z}\big(\frac{1}{193L\rho_\nu n\alpha_{max}}\big) = 1 - \frac{32n\phi}{193Lz}, \tag{A.120}$$

$$\geq 1 - \frac{32(12 + \frac{2}{775(193)})}{193zn} \geq 1 - \frac{16(12 + \frac{2}{775(193)})}{193(1 - \frac{384}{193^2(775)})} \geq 0. \tag{A.121}$$

□

Appendix B: Appendix to Chapter 3

B.1 Experimental Setup

In this section, a deeper background into how the experiments were run is detailed as well as some additional results and observations. First, details on the hyper-parameters used in order to reproduce the results are provided. Then, additional comments and details into the sampling approach are provided. Finally, some of the interesting observations encountered while solving the Amazon-670K and Wiki-325K recommender system problems are mentioned.

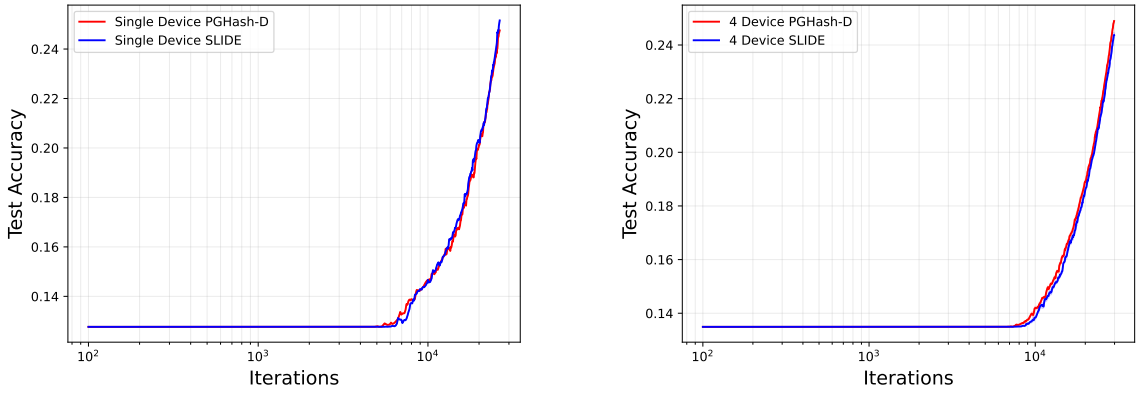
Below, the hyper-parameters used when running the federated experiments are detailed. What one can immediately see from Table B.1, is that a Densified Winner Take All (DWTA) variant of PGHash is used for the larger output datasets Amazon-670K and Wiki-325K. As experienced in previous work [77, 78, 83], SimHash fails to perform well on these larger datasets. It is surmised that SimHash fails due in part to its inability to select a large enough number of neurons per sample (the dearth of activated neurons is observed empirically). Reducing the hash length k does increase the number of neurons selected, however this decreases the accuracy. Therefore, DWTA is used because it utilizes more neurons per sample on these larger problems and also still achieves good accuracy. As a quick note, test accuracy is recorded every 100 iterations for Delicious-200K and Amazon-670K. Similar to Chen et al. [77], test accuracy is computed on 30 randomly sampled large batches of test data to reduce the test computations.

Table B.1: Hyper-parameters for federated experiments (PGHash & Federated SLIDE).

Dataset	Algorithm	Hash Type	LR	Batch Size	Steps per LSH	k	c	Tables	CR
Delicious-200K	PGHash	PGHash	1e-4	128	1	8	8	50	1
Delicious-200K	SLIDE	SimHash	1e-4	128	1	8	N/A	50	1
Amazon-670K	PGHash	PGHash-D	1e-4	256	50	8	8	50	1
Amazon-670K	SLIDE	DWTA	1e-4	256	50	8	N/A	50	1
Wiki-325K	PGHash	PGHash-D	1e-4	256	50	5	16	50	1
Wiki-325K	SLIDE	DWTA	1e-4	256	50	5	N/A	50	1

Table B.2: Hyper-parameters for compression experiments (PGHash).

Dataset	Algorithm	Hash Type	LR	Batch Size	Steps per LSH	k	c	Tables	CR
Delicious-200K	PGHash	PGHash	1e-4	128	1	8	8	50	0.1/0.25/1



(a) 1 Device Results.

(b) 4 Device Results.

Figure B.1: **PGHash-D results on Wiki-325K.** Model accuracy of a large recommendation network is recorded on Wiki-325K trained in a federated setting. PGHash-D matches or outperforms the convergence of SLIDE without requiring LSH to be performed by the central server.

B.2 Amazon-670K and Wiki-325K Experiment Analysis

Sub-par SimHash Performance. SimHash is known to perform worse than DWTA on Amazon-670K and Wiki-325K. Utilizing SimHash for these experiments is unfair as it is shown in previous work [77, 78], for example, that DWTA achieves much higher performance on Amazon-670K. For this reason, DWTA is the chosen hash function in Chen et al. [77] for

Table B.3: **PGHash and SLIDE performance on Amazon-670K using SimHash.** Accuracy across the first 5,000 iterations for a single device. Batch size $M = 1024$, $k = 8$, and $c = 8$.

Iteration	SLIDE	PGHash
1,000	10.82%	10.04%
2,000	18.27%	15.99%
3,000	21.83%	19.51%
4,000	23.72%	21.65%
5,000	25.08%	23.38%

Amazon-670K experiments. To verify this observation, experiments on Amazon-670K with PGHash (not PGHash-D) and SLIDE (with a SimHash hash function) are performed. Table B.3 compares the performance of the PGHash and SimHash approaches for Amazon-670K.

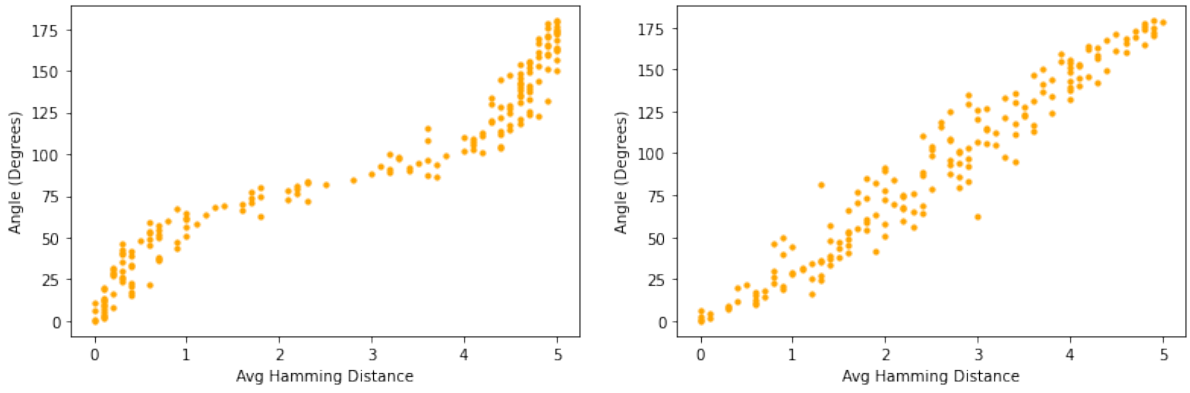
As shown in Table B.3, even with a much larger batch size, SLIDE and PGHash are unable to crack 30% on Amazon-670K. As a note, using a smaller batch size (like the $M = 256$ value used in the Amazon-670K experiments) resulted in an even further drop in accuracy. These empirical results back-up the notion that SimHash is ill-fit for Amazon-670K.

Wiki-325K Performance. In Figure B.1, the performance of PGHash-D on Wiki-325K is displayed. Quite similar to the Amazon-670K results (shown in Figure 3.5), PGHash-D almost exactly matches up with SLIDE. In order to map how well training progresses, test accuracy is periodically checked. However, since the test set is very large, determining test accuracies over the entire test set is infeasible due to time constraints on the cluster. Therefore, test accuracies are determined over 30 batches of test data as a substitute as is done in Chen et al. [77, 78]. For Delicious-200K and Amazon-670K, full test accuracy matched the randomly sampled batches, however the randomly sampled batches underestimate the full test accuracy for Wiki-325K. For Wiki-325K, the true test accuracy ran about 5% greater than the sampled test accuracy values.

Matching Full-Training Performance. Along with the failure for SimHash to perform well on Amazon-670K and Wiki-325K, SLIDE and PGHash(-D) are unable to match the performance of full-training on these data-sets. This is observed empirically for Amazon-670K by GResearch: www.gresearch.co.uk/blog/article/implementing-slide. It is surmised that the failure of SLIDE and PGHash(-D) to match full-training performance on Amazon-670K and Wiki-325K arises due to the small average labels per point in these two data-sets (5.45 and 3.19 respectively). Early on in training, SLIDE and PGHash(-D) do not utilize enough activated neurons. This is detrimental to performance when there are only a few labels per sample, as the neurons corresponding to the true label are rarely selected at the beginning of training (and these final layer weights are tuned much slower). In full-training, the true neurons are always selected and therefore the final layer weights are better adjusted from the beginning. As a note, Yan et al. [100] also requires a hidden layer size of 1024 for a distributed version of SLIDE to achieve improved test accuracies for Amazon-670K. Thus, increasing the hidden layer size may have improved the performance (it is kept as 128 to match the original SLIDE paper [77]).

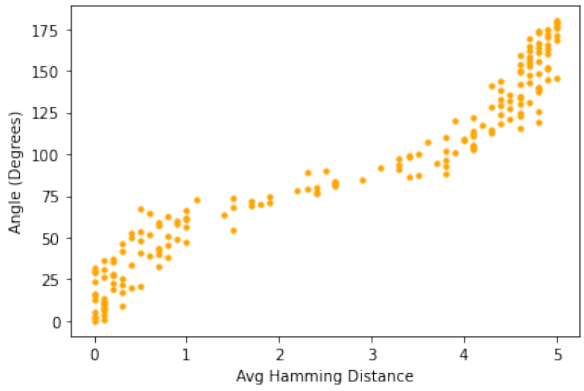
B.3 PGHash: Angle Versus Hamming Distance

In this section, the degree to which PGHash is a consistent estimator of angular similarity is visually explored. Specifically, let $x, y \in \mathbb{R}^d$: then, by Theorem 2, $\mathcal{H}^{PG}(c, d)$ is an LSH for $\cos(x_c, y_c)$. It is demonstrated that in the unit vector regime, $\theta_c = \arccos(\cos(x_c, Y_c))$ is an acceptable surrogate for $\theta = \arccos(x, Y)$, where $Y = \{y^i\}_{i=1}^N$ and $Y_c = \{y_c^i\}_{i=1}^N$.

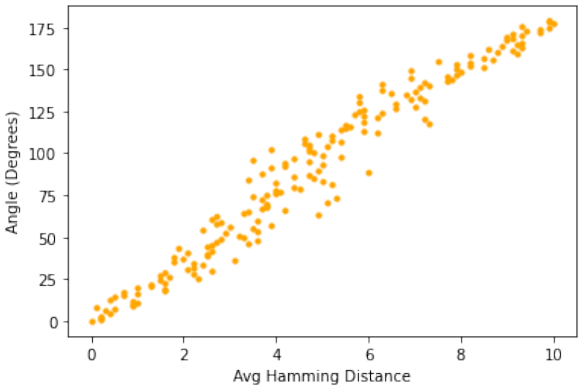


(a) $c = 5$

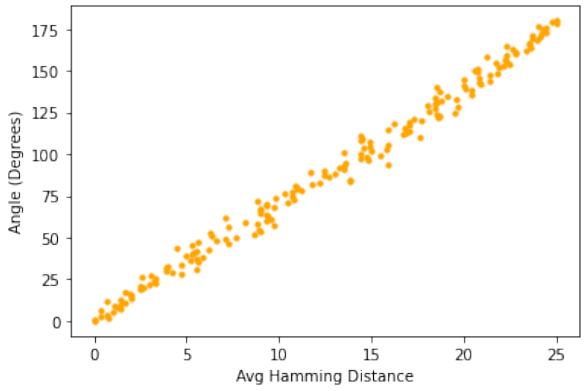
(d) $c = 5$



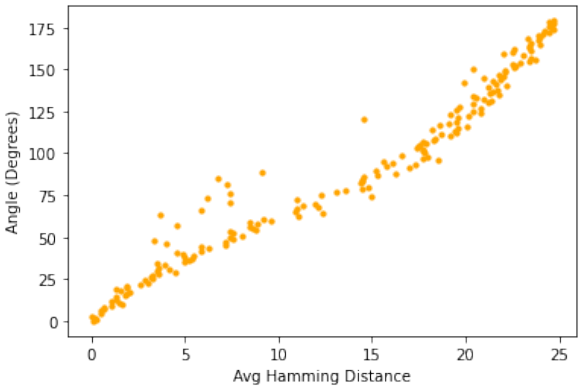
(b) $c = 10$



(e) $c = 10$

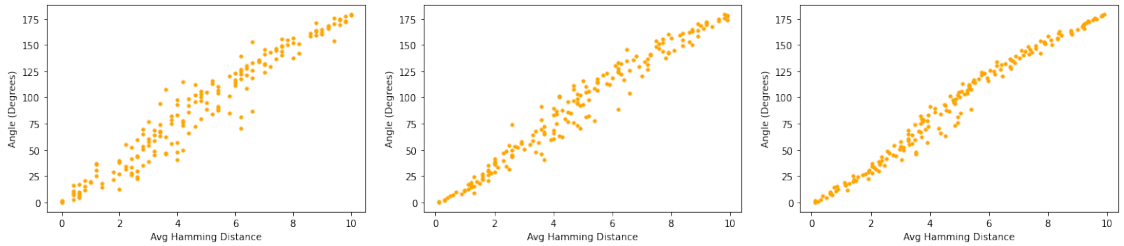


(c) $c = 25$



(f) $c = 25$

Figure B.2: **Angle vs. Hamming Distance as a function of sketch dimension.** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^{100}$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x. Increasing sketch dimension c smooths and reduces the variance of the scatter towards linear correlation. Furthermore, the Hamming scales linearly with c, improving discernibility. (a)-(c) & (d)-(f) are independent series.



(a) 5 Tables (b) 10 Tables (c) 50 Tables

Figure B.3: **Angle vs. Hamming Distance as a function of tables** The average Hamming distance between a PGHashed fixed unit vector $x \in \mathbb{R}^100$ and a collection of vectors $y_i \in \mathbb{R}^{100}$ which form different angles with x and fixed sketch dimension $c = 10$. Increasing the number of tables reduces variance.

B.4 Proofs

Fact 1. *Let x, y, e_1, e_2 be d-dimensional unit vectors such that that the e_i lie on the unit circle contained with the plane spanned by x and y (denoted as $S_{x,y}$) and $e_1 \perp e_2$. Consider the point v on $S_{x,y}$ such that the line through it bisects the angle of the lines passing through x and y. Let $\eta = \arccos(\cos(v, e_1))$. Denote $\theta = \frac{1}{2}\arccos(\cos(x, y))$. Then it may be written that $x = \cos(\eta + \theta)e_1 + \sin(\eta + \theta)e_2$ and $y = \cos(\eta - \theta)e_1 + \sin(\eta - \theta)e_2$.*

B.4.1 Proof of Theorem 3

Theorem 3 (Restated)**.** *Let $x, y \in \mathbb{S}^{d-1}$. Assume that neither x nor y vanish under multiplication by B and define the set $S_{x,y} := \{v \in \text{span}(x, y) : ||v|| = 1\}$. Let $\lambda = d/c$, $\theta := \frac{1}{2}\arccos(\cos(x, y))$, $\alpha = \inf\{c > 0 \mid |||Bv|| \geq c, \forall v \in S_{x,y}\}$, and $\beta = \alpha/\lambda$. Assume $\alpha > 0$. Then $\cos(x_c, y_c)$ lives between $\frac{1-\beta^2\tan^2\theta}{1+\beta^2\tan^2\theta}$ and $-\frac{\tan^2\theta^2-\beta^2}{\tan^2\theta^2+\beta^2}$.*

Proof. Let $\theta = \frac{1}{2}\arccos(\cos(x, y))$ where $x, y \in \mathbb{S}^{d-1}$ and $A = B^\top B$. The cosine similarity

between $x_c = Bx$ and $y_c = By$ (for B correspondent to a (d, c)-folding), is expressible as

$$\cos(x_c, y_c) = \frac{x^\top B^\top By}{(x^\top B^\top Bx)(y^\top B^\top By)} = \frac{x^\top Ay}{\sqrt{(x^\top Ax)(y^\top Ay)}}. \tag{B.1}$$

Consider the SVD $B = UDV^\top$ where U and V are orthogonal and D is $c \times d$ rectangular diagonal matrix. Therefore, $A = B^\top B = V\hat{D}^2V^\top$ (here $\hat{D}$ is now a square diagonal matrix containing squared D_{ii} along the diagonal and 0 everywhere else). Notice that choice of U nor the ordering of columns v_i of V affects the angle calculation in Equation B.1. First, the columns of V are reordered so as to order the diagonal entries d_i of D (*i.e.,* the squared singular values) in decreasing order, and as an abuse of notation set $B = \frac{1}{d_1}DV^\top$. Denoting $\hat{\lambda}_i = d_i/d_1$ for $1 \leq i \leq n$, then $Bv_i = \hat{\lambda}_i e_i$. By construction of B, it is true that $d_i \in \{\frac{d}{c}, 0\}$, therefore, $\hat{\lambda}_i \in \{1, 0\}$).

Consider B acting on S^{d-1}: it scales each dimension by $\hat{\lambda}_i$, thus (as with any linear transformation of a sphere), transforms it into an ellipsoid, with c principal axes determined by the v_i. The greatest possible distance from the origin to the ellipsoid BS^{d-1} is 1 while the shortest possible distance is 0. Now consider the unit circle $S_{x,y} = \{v \in \text{span}(x, y) \ : ||v|| = 1\}$. Now, $BS_{x,y} \subset BS^{d-1} \cap BU$ is an ellipse (since the intersection of an ellipsoid and plane is always an ellipse).

Choose unit w_1 and w_2 belonging to $S_{x,y}$ such that $w_1 \perp w_2$. By Fact 1, the vectors can be parameterized as $x = \cos(\eta - \theta)w_1 + \sin(\eta - \theta)w_2$ and $y = \sin(\eta + \theta)w_1 + \sin(\eta + \theta)w_2$, where η is the angle made with w_1 with the bisector of x and y. By assumption, $||Bw|| \geq \alpha$ (the minimal shrinking factor of B on $S_{x,y}$) for some positive α. Denoting $\lambda = \frac{d}{c}$ (the maximal

stretching factor of B on $S_{x,y}$), the angle between Bx and By is upper-bounded by,

$$f(\eta) = \arctan(\frac{\alpha}{\lambda}\tan(\eta+\theta)) - \arctan(\frac{\alpha}{\lambda}\tan(\eta-\theta)). \tag{B.2}$$

The numerator of $\frac{df}{d\eta}$ is $\beta(1-\beta)(1+\beta)\sin(2\theta)\sin(2\eta)$ where $\beta = \alpha/\lambda$. The derivative is trivially 0 if (1) $\beta = 0$, (2) $\beta = 1$, or (3) $\theta = 0$. (1) will not occur as it is assumed that $S_{x,y}$ does not contain a 0-eigenvector of $A = B^\top B$. (2) can only occur if A is a multiple of the identity matrix (which it is not by construction), and (3) implies that x and y are parallel, in which case their angle will not be distorted. Aside from these pathological cases, the critical points occur at $\eta = 0, \pi/2$. Therefore, $\cos(Bx, By)$ lives between $\cos(f(0)) = \frac{1-\beta^2\tan^2\theta}{1+\beta^2\tan^2\theta}$ and $\cos(f(\pi/2)) = -\frac{\tan^2\theta^2-\beta^2}{\tan^2\theta^2+\beta^2}$. □

Remark 19. *The constant β has an enormous influence on the bounds in Theorem 3. The smaller the α (i.e., shrinking of $||w||$), the greater the bounds on distortion. Although constraints have been imposed on x, y, if they are treated as any possible pair of random unit vectors, then the w in $S_{x,y}$ effectively becomes a random unit vector as well. One can exactly characterize the distribution of $||BX||$ where X denotes a random variable which selects a d-dimensional unit vector uniformly at random.*

B.4.2 Proof of Proposition 1

Proposition 1 (Restated)**.** *Let $u \in S^{d-1}$ be drawn uniformly at random.*

$||Bu||^2 \sim \text{Beta}(\frac{c}{2}, \frac{d-c}{2}, 0, \frac{d}{c})$,

the four parameter Beta distribution pdf $f(x) = \frac{(2x/c)^{c/2-1}(1-2x/c)^{(d-c)/2-1}}{(d/c)\beta(c/2,(d-c)/2}$, and $\mathbb{E}||Bu||^2 = 1$.

Proof. One can sample a d-dimensional vector uniformly at random from the unit sphere S^{d-1} by drawing a d-dimensional Gaussian vector with IID entries and normalizing. Represent this as the random variable $X = Z'/||Z'||$ where $Z' \sim \mathcal{N}(0, I_d)$. Consider a (c, d)-folding matrix B, *i.e.,* a d/c horizontal stack of $c \times c$ identity matrices (assuming $c|d$). The aim is determining the distribution of $||BX||^2$. For ease of notation, consider the permutation Z of Z' where $Z_i = Z'_{(\lfloor \frac{d/c}{i} \rfloor - 1)*(d/c)+i(\mod d/c)}$. Since this permutation is representable as an orthogonal matrix P (and multi-variate Gaussians are invariant in distribution under orthogonal transformations), one may instead consider $X := P(Z'/||Z'||)^2 = Z/||Z||^2$. The norm-squared is written as,

$$||BX||^2 = \frac{(Z_1+\cdots+Z_{d/c})^2}{||Z||^2} + \frac{(Z_{d/c+1}+\cdots+Z_{2d/c})^2}{||Z||^2} + \cdots + \frac{(Z_{(c-1)(d/c)+1}+\cdots+Z_d)^2}{||Z||^2}. \quad \text{(B.3)}$$

Consider the first term $\frac{(Z_1+\cdots+Z_{d/c})^2}{||Z||^2}$. First note that for any unit vector u, the distribution of $\frac{(u^\top Z)^2}{||Z||^2}$ does not depend on choice of u. Consider the unit vector u' then which contains $\sqrt{d/c}$ in the first d/c entries and 0 otherwise. Then $\frac{(u'^\top Z)^2}{||Z||^2}$ is equivalent to d/c times the first term. Of course, since $\frac{(e_1^\top Z)^2}{||Z||^2}$ has the same distribution as $\frac{(u'^\top Z)^2}{||Z||^2}$, by transitivity $\frac{Z_1^2}{||Z||^2} \overset{d}{=} (n/q)\frac{(Z_1+\cdots+Z_{d/c})^2}{||Z||^2}$.

By extending the discussion above to the other terms, and by their independence with respect to rotation of Z (since their numerators contain squared sums of mutually disjoint Z coordinates),

$$||BX||^2 \overset{d}{=} \frac{d}{c} \cdot \frac{Z_1^2 + Z_{d/c}^2 + Z_{2d/c}^2 + \cdots + Z_d^2}{||Z||^2}. \quad \text{(B.4)}$$

The distribution of $\frac{Z_1^2+Z_{d/c}^2+Z_{2d/c}^2+\cdots+Z_d^2}{||Z||^2}$ is well-known to follow a $\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2})$ distribution [225]. In totality, $||BX||^2 \overset{d}{=} \frac{d}{c}\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2})$. However, one will move to the four parameter description of this scaled Beta distribution which is $\mathrm{Beta}(\frac{c}{2}, \frac{d-c}{2}, 0, \frac{d}{c})$. The pdf and expected value follows

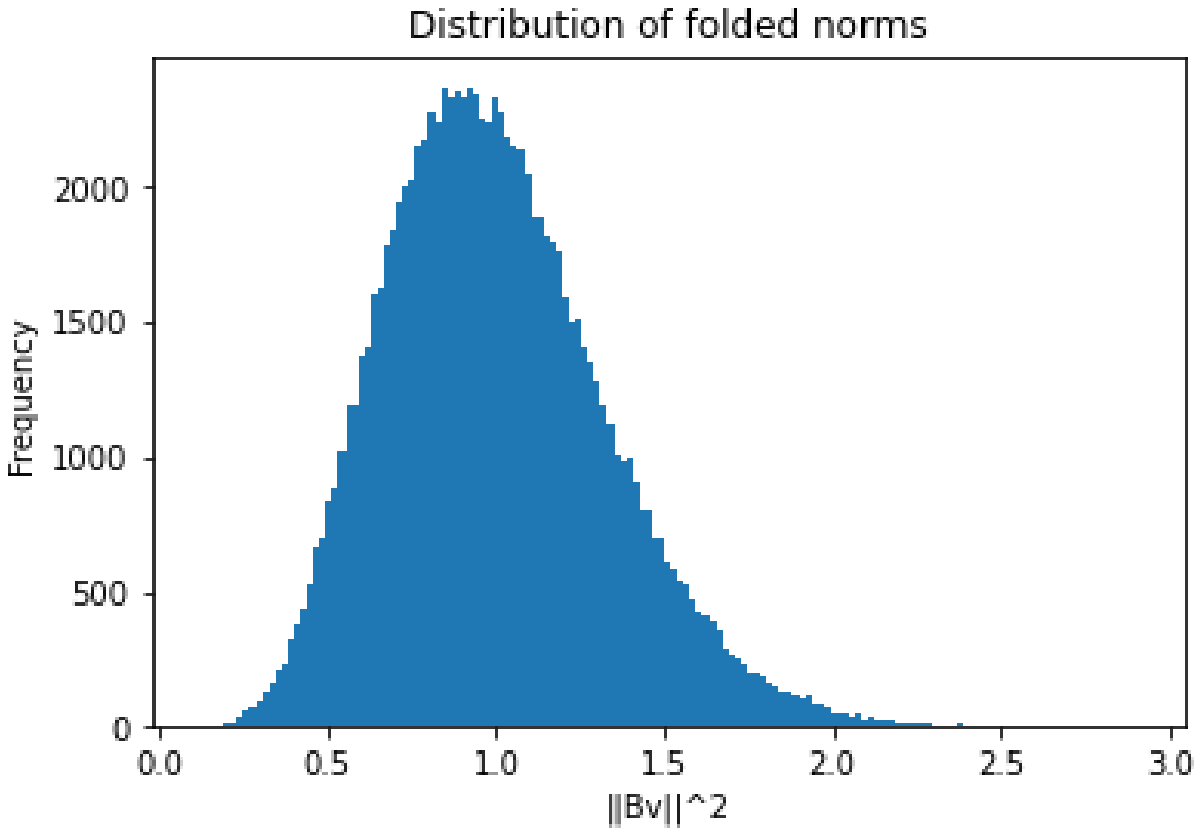


Figure B.4: **Distribution of Folded Norms.** One hundred thousand randomly drawn unit vectors ($d = 128$) are folded down to length 16 by are usual (d, c)-folding procedure. Depicted is a binned histogram of the norms. As predicted by the statistical description of $||BX||^2$, where X is a randomly drawn unit vector, the mass is centered at 1, *i.e.,* most norms are preserved. Empirically one can observe that folded rarely exceed $\sqrt{128}16$, although the theoretical support is $[0, 8]$: this concurs with the pdf.

by the usual statistical descriptions of this distribution, which can also be found in Gupta and Nadarajah [225]. □

Figure B.4 depicts how (d, c)-foldings affect the norms of unit vectors.

B.4.3 Additional Theory

Here, additional theory relevant to SimHash is provided. Several well-known results regarding SimHash are presented.

Proposition 2 (SimHash estimation)**.** *Let $x, y \in \mathbb{S}$, i.e., unit d-dimensional vectors. Denote $\theta = \arccos(|\cos(x, y)|)$. Let $v \in S^d$ be a unit vector drawn uniformly at random (according to the Haar measure, for example). Then,*

$$Pr[sgn(v^\top x) \neq sgn(v^\top y)] = \frac{\theta}{\pi}. \tag{B.5}$$

Proof. The argument of Goemans and Williamson [81] is reproduced here. Through symmetry, $Pr[\text{sgn}(v^\top x) \neq \text{sgn}(v^\top y)] = 2Pr[v^\top x > 0, v^\top y < 0]$. The set $\mathcal{U} = \{v \in S^d : v^\top x > 0, v^\top y \leq 0\}$ corresponds to the intersection of two half-spaces whose dihedral angle (*i.e.,* angle between the normals of both spaces) is exactly θ. Intersecting with the d-dimensional unit sphere produces gives a subspace of measure $\frac{\theta}{2\pi}$, therefore, $2Pr[v^\top x > 0, v^\top y < 0] = \frac{\theta}{\pi}$. □

Corollary 6. *Let v instead be a d-dimensional random Gaussian vector with IID entries $\sim \mathcal{N}(0, 1)$. Then for $x, y \in \mathbb{R}^d$,*

$$Pr[\text{sgn}(v^\top x) \neq \text{sgn}(v^\top y)] = \frac{\theta}{\pi}. \tag{B.6}$$

Proof. Randomly drawn, normalized Gaussian vectors are well-known to be uniformly distributed on the unit sphere. □

In the setup as above, let the X be a random variable which returns 1 if x and y have differing signs when taking the standard inner product with a randomly drawn Gaussian v. Let $X_1, X_2, \ldots, X_n$ represent a sequence of independent X events.

Proposition 3. $\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n} X_i] = 1 - \frac{\theta}{\pi}$ *and* $\mathbb{V}[X] = \frac{1}{N}\frac{\theta}{\pi}(1 - \frac{\theta}{\pi})$.

Given that PGHash is equivalent to a SimHash over (d, c)-foldings of R^d, the variance reduction one can observe by using multiple tables (Figure B.3) is explainable by Proposition 3.

Appendix C: Appendix to Chapter 5

C.1 Experimental Setup

Within the experiments, 16 devices to train a ResNet18 and a small convolutional neural network (for CIFAR-10 and MNIST respectively). The optimizer used is Stochastic Gradient Descent for CIFAR-10 and Adam for MNIST. A standard FedAvg algorithm is used to perform federated training. All experiments are run on a cluster of 2-4 GPUs, with the 16 CPUs (devices) pinned to a GPU. A table of the hyper-parameters not listed in Chapter 5.4 is shown below.

Test Loss and Accuracy Plots. Test loss and accuracy plots, including error bars, are included below for federated and local training for CIFAR-10, MNIST, and HAM10000. Each experiment is run three times. As expected, federated learning outperforms local training in the IID (left), mild non-IID (middle) and strong non-IID (right) settings. Federated training, via FedAvg, is more robust to non-IID distributions than local training. As detailed in Chapter 5.4, the non-IID distributions are Dirichlet with $\alpha = 0.3, 0.6$.

Algorithm 12 DeviceUpdate.
Input: model parameters $\boldsymbol{w}$, device index i, data used for training m_i, loss f, h local steps.

Output: updated model parameters $\boldsymbol{w}$.
$\mathcal{B} \leftarrow$ batch m_i data points
for each epoch $e = 1, \ldots, E$ **do**
    for batch $b \in \mathcal{B}$ **do**
        $\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \nabla f_i(\boldsymbol{w}; b)$
        every h batches pause training and return $\boldsymbol{w}$ (resume at next call)

Table C.1: Hyper-parameters for CIFAR-10 FACT Experiments.

Model	Batch Size	Learning Rate	Training Cost	Epochs	Local FedAvg Steps h
ResNet18	128	0.05	1.024e-07	100	6

Table C.2: Hyper-parameters for MNIST FACT Experiments.

Model	Batch Size	Learning Rate	Training Cost	Epochs	Local FedAvg Steps h
CNN	128	1e-3	7.111e-08	100	6

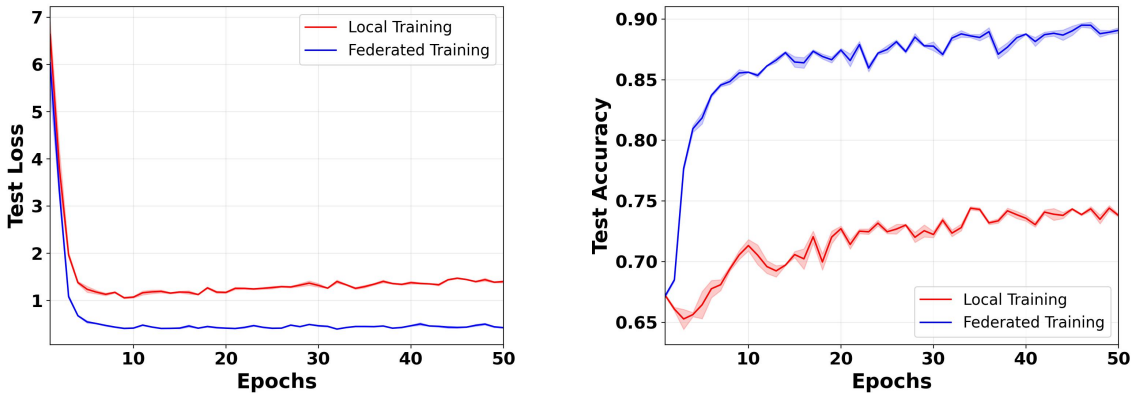


Figure C.1: **HAM10000 Test Loss (Left) and Accuracy (Right) for Federated and Local Training.** Consistent with the experiments for CIFAR-10 and MNIST, federated learning outperforms local training even though the HAM10000 dataset is highly skewed.

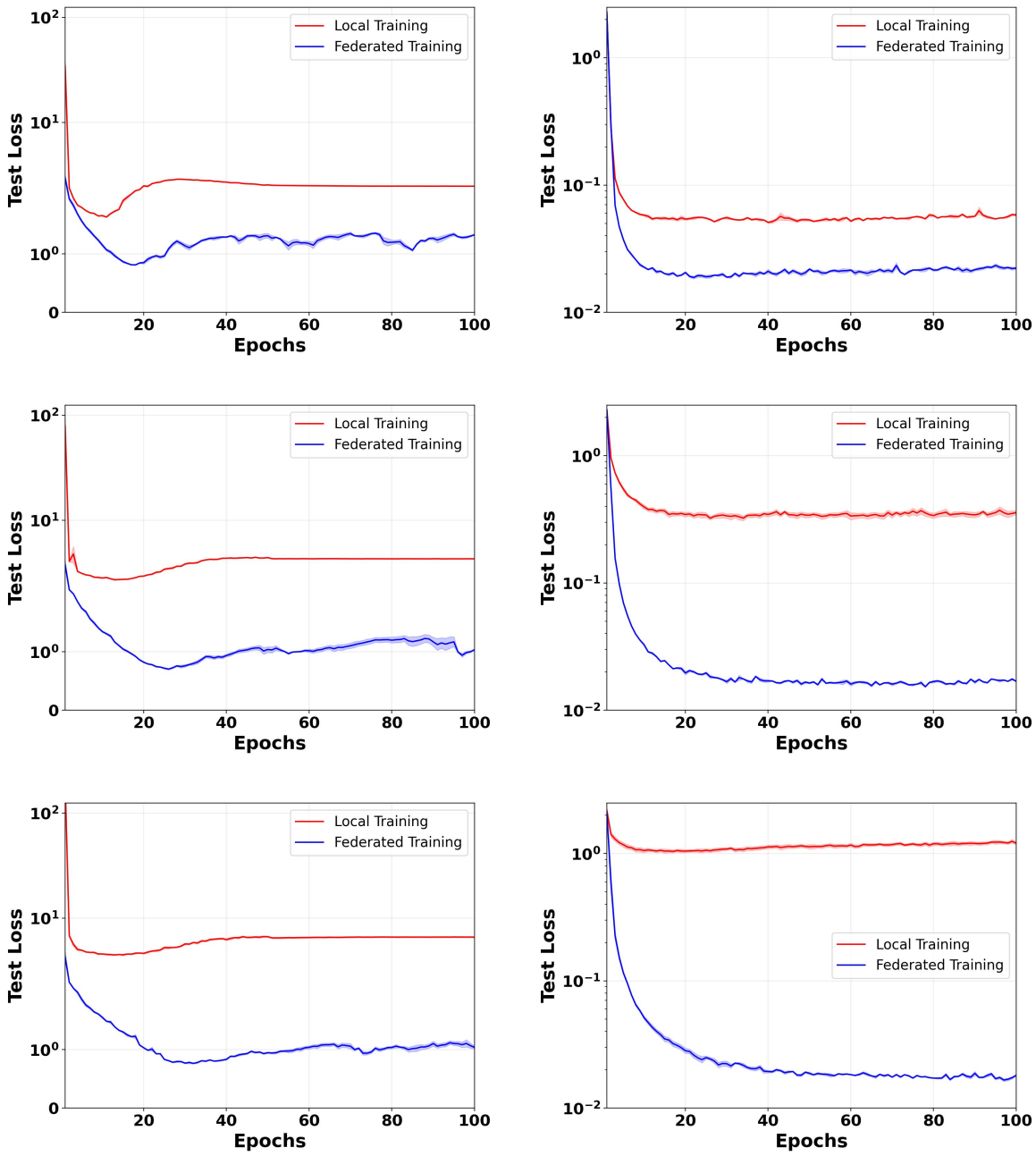


Figure C.2: **Test Loss for CIFAR-10 (left) and MNIST (right) in Heterogeneous Settings.** Federated learning outperforms local training on IID (top) and mild (center) & strong (bottom) non-IID Dirichlet settings.

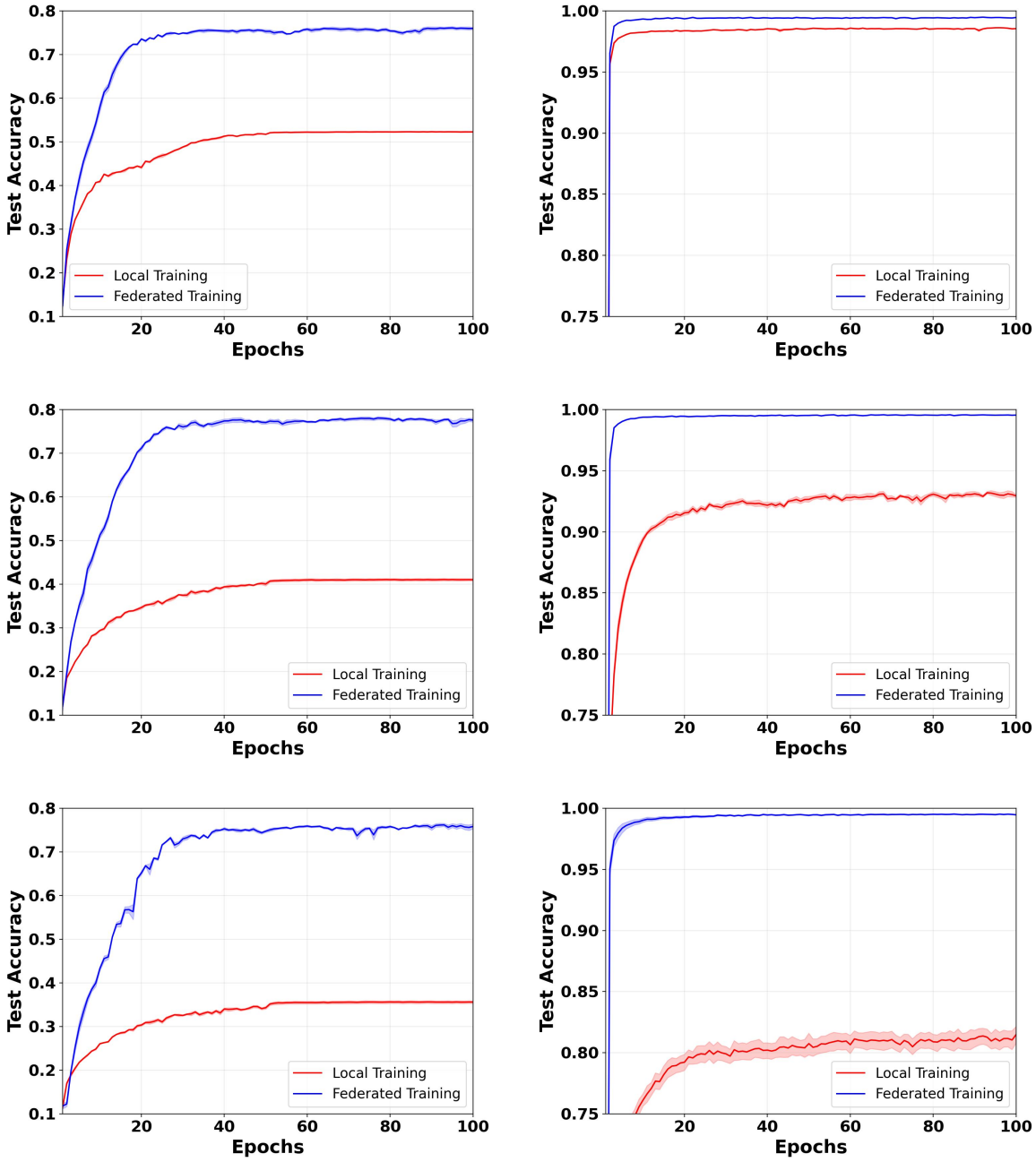


Figure C.3: **Test Accuracy for CIFAR-10 (left) and MNIST (right) in Heterogeneous Settings.** Federated learning outperforms local training on IID (top) and mild (center) & strong (bottom) non-IID Dirichlet settings.

C.2 Proofs

C.2.1 Proof of Theorem 4

Theorem 4 (Restated). *For a device i with marginal cost c_i, the optimal amount of data $m^*_{i,l}$ used for local training is $m^*_{i,l} := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$.*

Proof. Each device has a local loss function as shown in Equation (5.2). Taking the derivative of $\ell_{i,l}(m)$ with respect to the data contribution amount m yields,

$$\frac{d\ell_{i,l}}{dm} = -\frac{\gamma\sigma^2 L}{2m^2} + c_i = 0 \quad \longrightarrow \quad m^*_{i,l} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}. \tag{C.1}$$

□

C.2.2 Proof of Theorem 5

Theorem 5 (Restated). *For a device i with marginal cost c_i, the optimal amount of data $m^*_{i,F}$ used for federated training is $m^*_{i,F} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}} - \sum \boldsymbol{m}_{-i}$.*

Proof. Each device has a federated loss function as shown in Equation (5.3). Taking the derivative of $\ell_{i,F}(m)$ with respect to the data contribution amount m yields,

$$\frac{d\ell_{i,l}}{dm} = -\frac{\gamma\sigma^2 L}{2(m + \sum \boldsymbol{m}_{-i})^2} + c_i = 0, \tag{C.2}$$

$$(m + \sum \boldsymbol{m}_{-i})^2 = \frac{\gamma\sigma^2 L}{2c_i} \quad \longrightarrow \quad m^*_{i,F} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}} - \sum \boldsymbol{m}_{-i}. \tag{C.3}$$

□

C.2.3 Proof of Theorem 6

Theorem 6 (Restated). *For a device i with marginal cost c_i, the optimal amount of data $m^*_{i,PFL}$ used for federated training under Equation (5.4) is $m^*_{i,PFL} := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$.*

Proof. Each device has a penalized federated loss function as shown in Equation (5.5). Taking the derivative of $\ell_{i,PFL}(m)$ with respect to the data contribution amount m yields,

$$\begin{aligned}\frac{d\ell_{i,PFL}}{dm} &= -\frac{\gamma\sigma^2 L}{2(m+\sum \boldsymbol{m}_{-i})^2} + c_i + \frac{\gamma\sigma^2 L}{2(\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + \sum \boldsymbol{m}_{-i})^2} - c_i - 2\lambda_i\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + 2\lambda_i m = 0, \\ &= 2\lambda_i\big(m - \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}\big) - \frac{\gamma\sigma^2 L}{2(m+\sum \boldsymbol{m}_{-i})^2} + \frac{\gamma\sigma^2 L}{2(\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + \sum \boldsymbol{m}_{-i})^2} = 0. \end{aligned} \tag{C.4}$$

Due to the convexity of Equation (5.5), as each piece of the utility function is convex, there is a single minimum which is carefully constructed to be at $m^* = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$,

$$\begin{aligned}\frac{d\ell_{i,PFL}(m^*)}{dm} &= 2\lambda_i\big(\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} - \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}\big) - \frac{\gamma\sigma^2 L}{2(\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + \sum \boldsymbol{m}_{-i})^2} + \frac{\gamma\sigma^2 L}{2(\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + \sum \boldsymbol{m}_{-i})^2}, \\ &= 0 \quad \longrightarrow m^*_{i,PFL} = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}. \end{aligned} \tag{C.5}$$

□

C.2.4 Proof of Lemma 2

Lemma 2 (Restated). *Let c_i be the marginal cost for a device i, $m^* := \sqrt{\frac{\gamma\sigma^2 L}{2c_i}} = m^*_{i,l} = m^*_{i,PFL}$ be the device's locally optimal data usage, and $\alpha \in [0, 2)$ be a server-specified hyper-parameter. In order to ensure that the optimal penalized federated loss is at least lower than the optimal*

local loss, $\ell_{i,PFL}(m_i^*) \leq \ell_{i,l}(m_i^*)$, one must select λ_i such that,

$$\lambda_i := \frac{m^*(\sum \boldsymbol{m}_{-i} + m^*)}{(2-\alpha)\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}} \left(c_i - \frac{\gamma\sigma^2 L}{2(\sum \boldsymbol{m}_{-i} + m^*)^2} \right)^2.$$

Selection of such λ_i results in a loss gap between $\ell_{i,PFL}(m^*)$ and $\ell_{i,l}(m^*)$ of,

$$\Delta\ell_i(m^*) := \ell_{i,l}(m^*) - \ell_{i,PFL}(m^*) = \frac{\alpha}{4} \left(\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{m^*(\sum \boldsymbol{m}_{-i} + m^*)} \right).$$

Proof. The range of λ_i values which ensure IR can be found by plugging in $m_{i,l}^* = m_{i,PFL}^* = m^*$ into Equations (5.2) and (5.5) and finding the difference,

$$\begin{aligned} \ell_{i,l}(m^*) - \ell_{i,PFL}(m^*) &= \frac{\gamma\sigma^2 L}{2} \left(\frac{1}{m^*} - \frac{1}{m^* + \sum \boldsymbol{m}_{-i}} \right) - \lambda_i \left(\frac{c_i}{2\lambda_i} - \frac{\gamma\sigma^2 L}{4\lambda_i (\sqrt{\frac{\gamma\sigma^2 L}{2c_i}} + \sum \boldsymbol{m}_{-i})^2} \right)^2, \\ &= \frac{\gamma\sigma^2 L}{2} \frac{\sum \boldsymbol{m}_{-i}}{m^*(m^* + \sum \boldsymbol{m}_{-i})} - \frac{1}{4\lambda_i} \left(c_i - \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})^2} \right)^2. \end{aligned} \tag{C.6}$$

To ensure the Equation (C.6) is greater than or equal to 0, λ_i must be selected as the following,

$$\begin{aligned} &\frac{\gamma\sigma^2 L}{2} \frac{\sum \boldsymbol{m}_{-i}}{m^*(m^* + \sum \boldsymbol{m}_{-i})} - \frac{1}{4\lambda_i} \left(c_i - \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})^2} \right)^2 \geq 0, \\ &\lambda_i \geq \frac{m^*(m^* + \sum \boldsymbol{m}_{-i})}{2\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}} \left(c_i - \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})^2} \right)^2. \end{aligned} \tag{C.7}$$

Equation (C.7) shows that λ_i must be non-negative (all values are positive and the squared term can be zero at a minimum). One can think of λ_i as the parameter controlling the benefit received by a device i. A larger value of λ_i will result in a larger gap between $\ell_{i,l}(m^*)$ and $\ell_{i,PFL}(m^*)$ (Equation (C.6)). Inversely, a smaller value of λ_i will result in a smaller gap between local and

federated training utility. Therefore, λ_i is defined with a user or server specified hyper-parameter $\alpha \in [0, 2)$,

$$\lambda_i := \frac{m_i(\sum \boldsymbol{m}_{-i} + m^*)}{(2-\alpha)\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}} \left(c_i - \frac{\gamma\sigma^2 L}{2(\sum \boldsymbol{m})^2} \right)^2. \tag{C.8}$$

Plugging this term into Equation (C.6) yields the following utility gap between $\ell_{i,l}(m^*)$ and $\ell_{i,PFL}(m^*)$,

$$\Delta\ell_i(m^*) := \ell_{i,l}(m^*) - \ell_{i,PFL}(m^*) = \frac{\alpha}{4} \left(\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{m^*(\sum \boldsymbol{m_{-i}} + m^*)} \right). \tag{C.9}$$

□

C.2.5 Proof of Theorem 7

Theorem 7 (Restated)**.** *PFL (Algorithm 8) using λ_i from Lemma 2 is IR and eliminates the free-rider dilemma when devices are truthful.*

Proof Reproduced from Section 5.2. The result of Theorem 6 is that each device i's optimal strategy within the penalized federated scheme is to use their locally optimal amount of data $m_i^* = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$. Furthermore, Lemma 2 states that using m_i^* within the penalized federated scheme results in a reward, or improvement over local training, of $\Delta\ell_i = \frac{\alpha}{4}\left(\frac{\gamma\sigma^2 L \sum \boldsymbol{m}_{-i}}{m_i^*(m_i^* + \sum \boldsymbol{m}_{-i})}\right)$. Thus, by combining Theorem 6 and Lemma 2, truthful devices which choose to participate in PFL attain a reshaped and lower loss at the same optimum. Individually rational devices would therefore prefer to participate in PFL (Algorithm 8) over local training due to the reshaped optimum's lower loss. This optimum achieves the same data usage as local training, thereby eliminating the free-rider dilemma. □

C.2.6 Proof of Lemma 3

Lemma 3 (Restated)**.** *The probability of a device "winning" in the truthfulness mechanism in Equation* (5.10)*, given a reported cost c, is $v(c) := 2F_C(c)(1 - F_C(c))$, where F_C is the CDF of f_C.*

Proof. The probability that a device's cost c is sandwiched in between two other randomly sampled devices' costs c_u, c_v is equal to,

$$P(\text{sandwiched } c) = v(c) = P(c_u \leq c \leq c_v) + P(c_v \leq c \leq c_u). \tag{C.10}$$

Let F_C be the cumulative distribution function (CDF) of the device cost distribution f_C. The probability that a random cost, let's say c_u, is smaller than c is equal to $F_C(c)$. The probability that a random cost, let's say c_v, is larger than c is $1 - F_C(c)$. Since the costs are random, and the above situation could be flipped (*i.e.,* c_v is smaller than c with probability $F_C(c)$ and c_u is larger than c with probability $1 - F_C(c)$), the two probabilities in Equation (C.10) are equivalent. Thus, due to symmetry, $P(c_u \leq c \leq c_v) = P(c_v \leq c \leq c_u)$. Therefore, one can rewrite the equation above as,

$$v(c) = 2F_C(c)(1 - F_C(c)). \tag{C.11}$$

□

C.2.7 Proof of the Main Theorem

Theorem 8 (Restated)**.** *Each device i's best strategy, when participating in* FACT *(Equation 5.11), is to report its true cost and use its locally optimal amount of data* $(m_i, c)_i^* = (\sqrt{\frac{\gamma\sigma^2 L}{2c_i}}, c_i)$.

Proof. Devices participating in FACT follow the loss function shown in Equation (5.11),

$$\ell_{i,Fact}(m_i, c) = \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i + P_{fr}(m_i, c) + P_{ct}(c). \tag{C.12}$$

Taking the expectation of the equation above over the distribution of agent costs f_C, and using the results of Lemma 3, yields,

$$\begin{aligned}
\mathbb{E}_{f_C}[\ell_{i,Fact}(m_i, c)] &= \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i + P_{fr}(m_i, c) + \mathbb{E}_{f_C}[P_{ct}(c)], \\
&= \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} + c_i m_i + P_{fr}(m_i, c) + \Delta\ell_i(m_i, c) - \frac{3\upsilon(c)}{n} \sum_{j \neq i \in [n]} \Delta\ell_j.
\end{aligned} \tag{C.13}$$

The term $\Delta\ell_i$ is defined in Lemma 2, specifically Equation 5.7. It is the contract fee paid to the server, and thus is also a function of the reported cost c,

$$\begin{aligned}
\Delta\ell_i(m_i, c) &= \ell_{i,l}(m_i, c) - \ell_{i,PFL}(m_i, c), \\
&= \frac{\gamma\sigma^2 L}{2m_i} + cm_i - \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} - cm_i - P_{fr}(m_i, c), \\
&= \frac{\gamma\sigma^2 L}{2m_i} - \frac{\gamma\sigma^2 L}{2(m_i + \sum \boldsymbol{m}_{-i})} - P_{fr}(m_i, c).
\end{aligned} \tag{C.14}$$

Substituting this back into Equation (C.13) yields,

$$\frac{\gamma\sigma^2 L}{2m_i} + c_i m_i - \frac{3\upsilon(c)}{n} \sum_{j \neq i \in [n]} \Delta\ell_j. \tag{C.15}$$

The partial derivative of $\mathbb{E}_{f_C}[\ell_{i,Fact}(m_i, c)]$ with respect to c yields,

$$\begin{aligned}
&\frac{\partial}{\partial c}\left[\frac{\gamma\sigma^2 L}{2m_i} + c_i m_i - \frac{3\upsilon(c)}{n} \sum_{j \neq i \in [n]} \Delta\ell_j\right] = 0, \\
&\frac{\partial}{\partial c}\left[-\frac{3\upsilon(c)}{n} \sum_{j \neq i \in [n]} \Delta\ell_j\right] = 0, \\
&-\frac{3}{n} \sum_{j \neq i \in [n]} \Delta\ell_j \frac{\partial}{\partial c}\Big[\upsilon(c)\Big] = -\frac{3}{n} \sum_{j \neq i \in [n]} \Delta\ell_j \frac{\partial}{\partial c}\Big[2F_C(c)(1 - F_C(c))\Big] = 0.
\end{aligned} \tag{C.16}$$

The final line follows from Lemma 3. Since $F_C(c)$ is a CDF, its range is $[0, 1]$. The function $\upsilon(c) = 2F_C(c)(1 - F_C(c))$ achieves is global maximum when $F_C(c) = 1/2$ (its critical point). Therefore, the value of c which satisfies Equation (C.16) is one such that $F_C(c) = 1/2$. By Assumption 4, the median value is assumed by each agent i as c_i, since each agent believes its cost is equally likely to be larger or smaller than any others'. Thus, $F_C(c_i) = 1/2$ and $c^* = c_i$. The partial derivative of $\mathbb{E}_{f_C}[\ell_{i,Fact}(m_i, c)]$ with respect to m_i yields,

$$\begin{aligned}
&\frac{\partial}{\partial m_i}\left[\frac{\gamma\sigma^2 L}{2m_i} + c_i m_i - \frac{3\upsilon(c)}{n} \sum_{j \neq i \in [n]} \Delta\ell_j\right] = 0, \\
&c_i - \frac{\gamma\sigma^2 L}{2m_i^2} = 0, \\
&\frac{\gamma\sigma^2 L}{c_i} = 2m_i^2 \longrightarrow m_i = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}.
\end{aligned} \tag{C.17}$$

Thus, the optimal value for m_i is $m_i^* = \sqrt{\frac{\gamma\sigma^2 L}{2c_i}}$. Plugging this optimal value of c back into the

optimal value for m_i leads to the following optimum: $(m_i, c)^* = (\sqrt{\frac{\gamma\sigma^2 L}{2c_i}}, c_i)$. □

Corollary 3 (Assurance of IR at Optimum)**.** *When using their best strategy, devices participating in* FACT *will never receive worse loss than training alone,* $\ell_{i,Fact}(m_i^*, c_i^*) \leq \ell_{i,l}(m_{i,l}^*)$.

Proof. First, it is important to note that the optimal data contributions in FACT and locally are equivalent, $m_{i,l}^* = m_i^*$, as shown in Theorems 4 and 8. Thus, both are denoted below as simply m^*. Similarly, the optimal cost c_i^* in FACT is simply the true cost c_i. This too will be used below instead of c_i^* Now, to prove this Lemma, it must be true that $\ell_{i,Fact}(m^*, c_i) - \ell_{i,l}(m^*) \leq 0$,

$$
\begin{aligned}
&\ell_{i,Fact}(m^*, c_i) - \ell_{i,l}(m^*) \\
&= \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + c_i m^* + P_{fr}(m^*, c_i) + P_{ct}(m^*, c_i) - (\frac{\gamma\sigma^2 L}{2m^*} + c_i m^*), \\
&= \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + P_{fr}(m^*, c_i) + P_{ct}(m^*, c_i) - \frac{\gamma\sigma^2 L}{2m^*}, \\
&\leq \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + P_{fr}(m^*, c_i) + \Delta\ell_i(m^*, c_i) - \frac{\gamma\sigma^2 L}{2m^*}, \\
&= \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + P_{fr}(m^*, c_i) + \ell_{i,l}(m^*, c_i) - \ell_{i,PFL}(m^*, c_i) - \frac{\gamma\sigma^2 L}{2m^*}, \\
&= \frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + \ell_{i,l}(m^*, c_i) - \big(\frac{\gamma\sigma^2 L}{2(m^* + \sum \boldsymbol{m}_{-i})} + c_i m^*\big) - \frac{\gamma\sigma^2 L}{2m^*}, \\
&= 0.
\end{aligned}
\tag{C.18}
$$

□

C.3 Impact & Limitations

Current federated systems are inhibited by device free riding. The result is an unfair system. Some devices perform the bulk of training, while others sit idle. In the end, all the devices receive

the same model performance. While there are current methods which eliminate device free riding, they do not take into account device truthfulness. Many devices can still provide false information to the server in order to free ride. The impact of FACT is that it provides an easily implementable mechanism which can make federated training more robust to free riding. Using FACT, devices are incentivized to no longer free ride even if they can lie about their training costs.

Chapter 5 faces limitations when devices and the central server act maliciously. On the device side, it is assumed that devices are rational and do not collude with one another. A future research direction is to prove that equilibria for FACT exist when devices are boundedly rational or colluding. On the server side, it is assumed that the central server acts honestly. In settings where the server cannot be trusted, new incentives or avenues must be built in order to ensure server honesty.

Appendix D: Appendix to Chapter 6

D.1 Experimental Setup

Below, further details into how the image classification experiments were run are provided. As a note, each experiment is run three times, varying the random seeds. All bar plots in Chapter 6 showcase the mean results of the three experiments. In Figures D.1 and D.2, test accuracies and error bars are plotted for CIFAR-10 & MNIST. The error bars are thin, as the results did not vary much between each of the three experiments. Finally, for simplicity and conservative results within all of the experiments, **the server's profit margin is set as $p_m = 1$ (greedy server)**. All experiments were run using a cluster of 2-4 GPUs shared across 8/16 CPUs. GeForce GTX 1080 Ti GPUs (11GB of memory) and Xeon 4216 CPUs are used for the experiments.

Within the experiments, both 8 and 16 devices train a ResNet18 and a small convolutional neural network for CIFAR-10 and MNIST respectively. The SGD optimizer is used for CIFAR-10 and Adam for MNIST during training. As detailed in Appendix D.1, $a_C(m)$ is carefully selected and tuned to match the empirical training results on both datasets as closely as possible. Once tuned, a fixed marginal cost c_e is selected: 2.5e-4 (4e-5) for CIFAR-10 (MNIST) on all baselines. When performing uniform cost experiments, each device uses c_e as its marginal cost (and thus each device has the same amount of data). For non-uniform cost experiments, c_e is the mean of a Gaussian distribution from which the marginal costs are sampled. The uniform accuracy payoff

is $\phi_i(a) = \frac{1}{(1-a)^2} - 1$ for each device i. For non-uniform payoff experiments, the accuracy payoff function is set as $\phi_i(a) = z_i(\frac{1}{(1-a)^2} - 1)$, where z_i is uniformly sampled within $[0.9, 1.1]$.

Table D.1: Hyper-parameters for CIFAR-10 Experiments.

Model	Batch Size	Learning Rate	Marginal Cost c_e	a_{opt}	k	Epochs	Local FedAvg Steps h
ResNet18	128	0.05	2.5e-4	0.95	10	100	6

Table D.2: Hyper-parameters for MNIST Experiments.

Model	Batch Size	Learning Rate	Marginal Cost c_e	a_{opt}	k	Epochs	Local FedAvg Steps h
CNN	128	1e-3	4e-5	0.9975	0.25	100	6

Experimental Process. The experimental process involved careful tuning of the theoretical accuracy function $\hat{a}_C(m)$ in order to match the empirical accuracy results found. In fact, for CIFAR-10, $\hat{a}_C(m)$ defined in Equation 6.4 is used with carefully selected values for k and $\bar{a}_{opt}$ to precisely reflect the empirical results found for the CIFAR-10 training. For MNIST, however, an accuracy function of $\hat{a}_C(m) = \bar{a}_{opt} - 2\sqrt{k/m}$ was more reflective of the ease of training on MNIST. Overall, to ensure precise empirical results, the following process is followed for each experiment (for both CIFAR-10 and MNIST as well as for 8 and 16 devices):

1. Determine $\bar{a}_{opt}$ and k such that $\hat{a}_C(m)$ is tuned to most precisely reflect the empirical results.
2. Select a marginal cost c_e low enough for a non-zero amount of data to be used by devices.
3. For non-uniform experiments, draw a marginal cost at random from a Gaussian with mean c_e and/or draw a payoff function ϕ_i with a z_i sampled within $[0.9, 1.1]$ (detailed in Chapter 6.4).
4. Given the marginal cost and $a_C(m)$, derive the locally optimal data m_i for each device i.
5. Save an initial model for training.
6. Train this initial model locally on each device until convergence, generating a_{local}.

7. Using the initial model, train it in a federated manner until convergence, generating a_{fed}.
8. Using the accuracy-shaping function, compute the amount of additional data contributions required to raise a_{local} to a_{fed}. *This is the incentivized amount of added contributions*.

Also, the following expected payoff function is used for the central server: $\phi_C = 1/(1-a)^2 - 1$. A learning rate scheduler is used for CIFAR-10 but not MNIST. Below, the hyper-parameters used in the CIFAR-10 and MNIST experiments are detailed.

D.2 Additional Experiments

It is interesting to note how well REALFM performs on MNIST (in terms of the vastly improved utility seen in Figures 6.4 and D.3) while FedAvg only improves model accuracy by a mere couple of percentage points. The reason stems from the payoff function ϕ_i which heavily rewards models that have accuracies close to 100%. This scenario is rational in real-world settings. Competing companies in industry will all likely have models which are high-performing (above 95% accuracy). Since the competition is stiff, companies with the best model performance will likely attract the most customers since their product is the best. Therefore, the utility for achieving model performance close to 100% should become larger and larger as one gets closer to 100%. In Figures D.1 and D.2, additional accuracy curves for the 16 and 8 device experiments are displayed as well as device utility comparisons for REALFM versus its baseline algorithms.

D.2.1 8 Device Experiments

Below, the 8-device experimental results are provided for CIFAR-10 and MNIST. These plots mirror those shown in Chapter 6.4 for 16 devices. First, test accuracy curves are provided

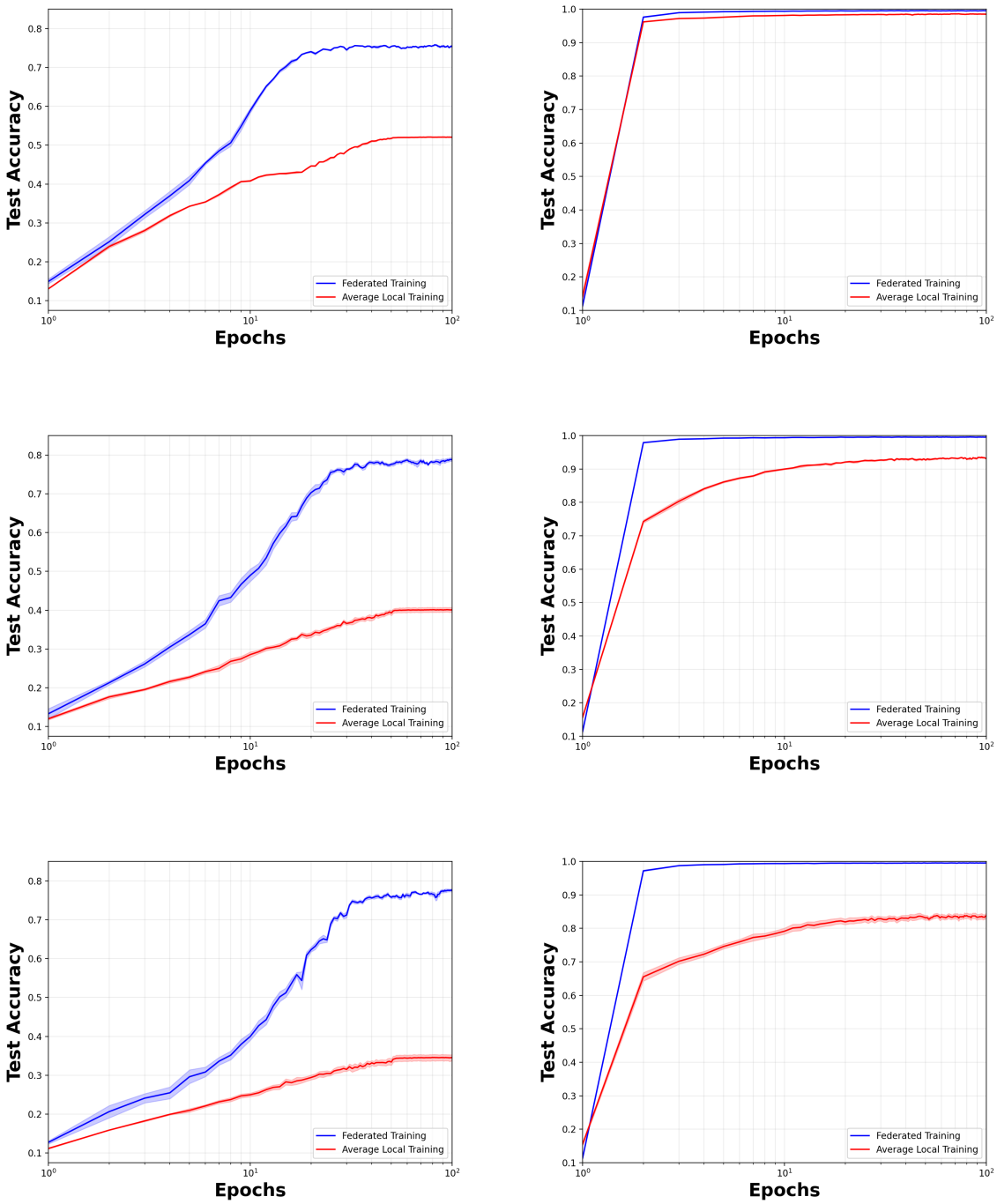


Figure D.1: **Accuracy Comparison on CIFAR-10 & MNIST.** The difference between local training (red line) and federated training (blue line) on CIFAR-10 (left column) and MNIST (right column) for 16 devices with uniform marginal costs and payoff functions is displayed. As expected, federated training always outperforms local training. Both uniform and various heterogeneous Dirichlet data distributions are shown above (top: uniform, middle: D-0.6, bottom: D-0.3).

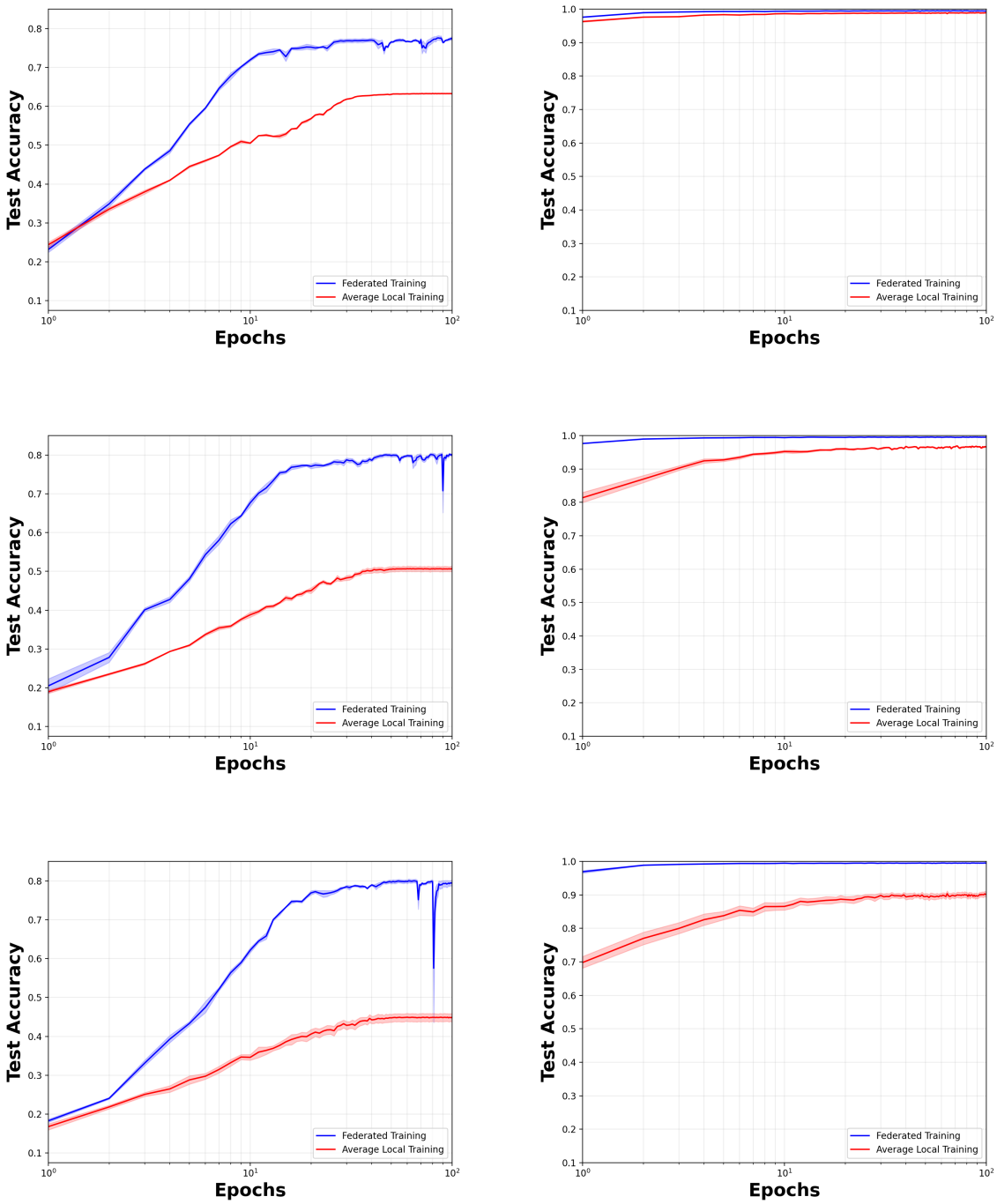


Figure D.2: **Accuracy Comparison on CIFAR-10 & MNIST.** The difference between local training (red line) and federated training (blue line) on CIFAR-10 (left column) and MNIST (right column) for 8 devices with uniform marginal costs and payoff functions is displayed. As expected, federated training always outperforms local training. Both uniform and various heterogeneous Dirichlet data distributions are shown above (top: uniform, middle: D-0.6, bottom: D-0.3).

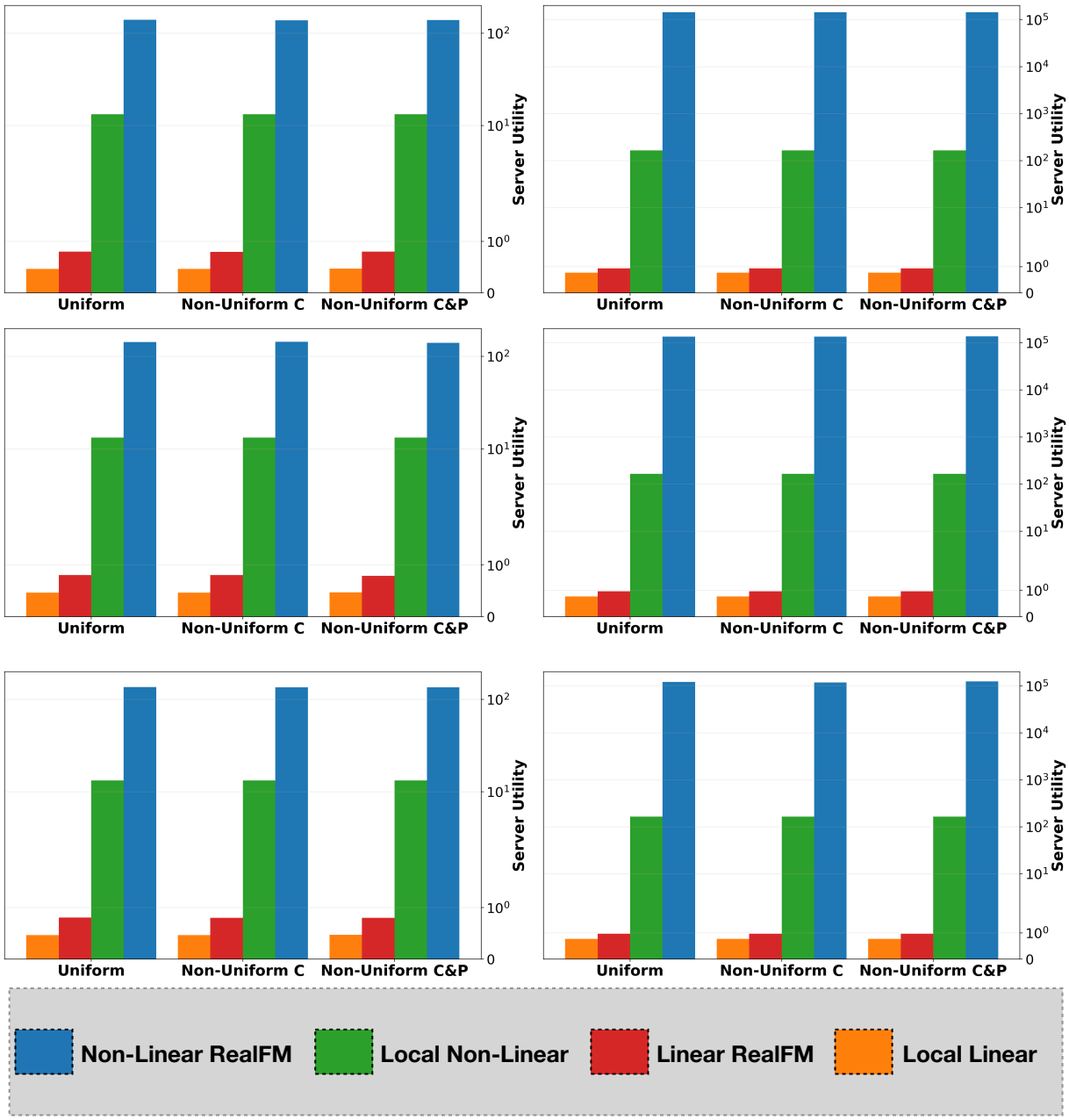


Figure D.3: **Improved Server Utility on CIFAR-10 & MNIST.** REALFM increases server utility on CIFAR-10 (top row) and MNIST (bottom row) for 8 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a FL version of [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

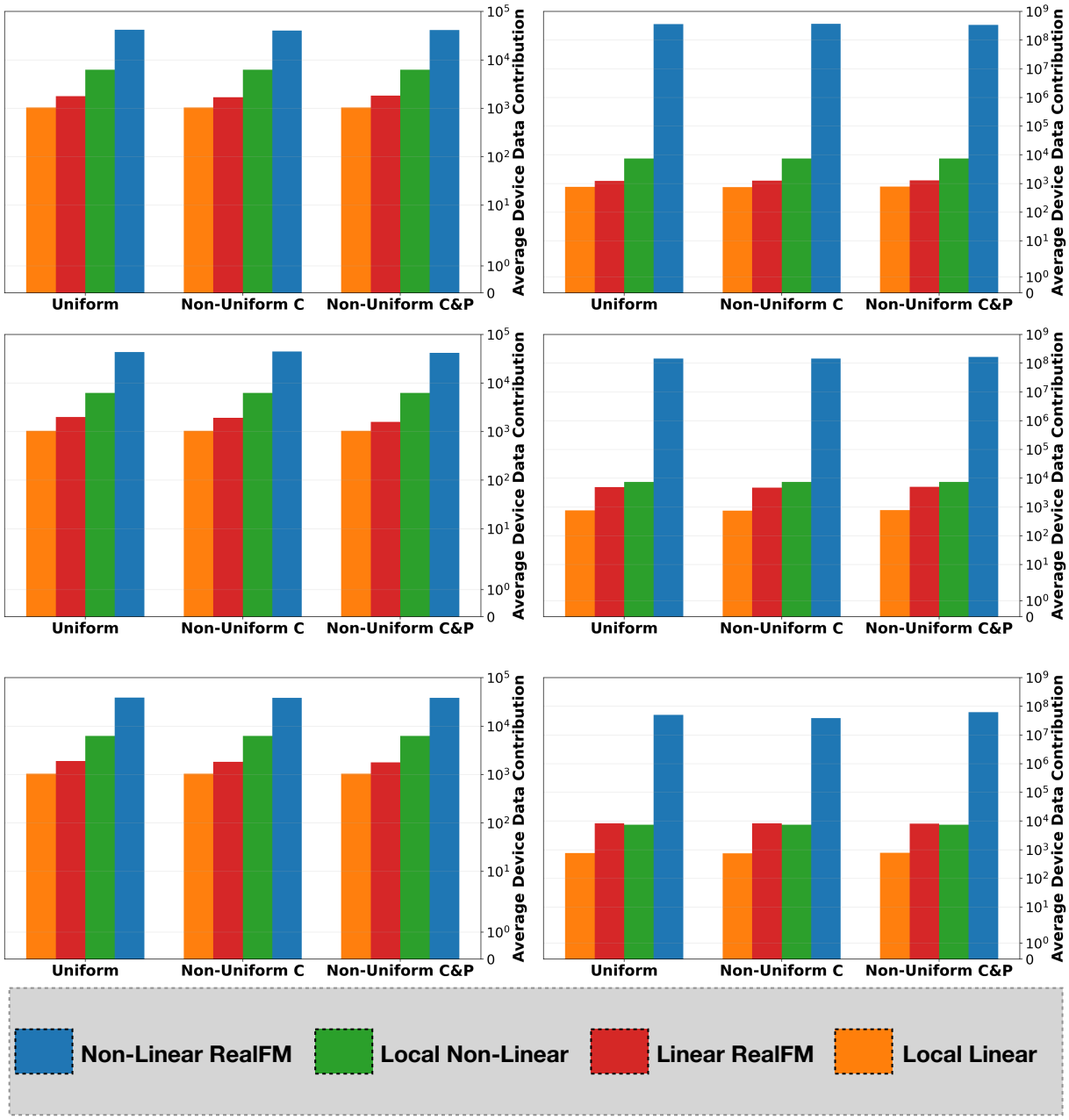


Figure D.4: **Increased Federated Contribution on CIFAR-10 & MNIST.** REALFM incentivizes devices to use more local data during federated training on CIFAR-10 (left column) and MNIST (right column) for 8 devices compared to relevant baselines. REALFM achieves upwards of 4 magnitudes more federated contributions than a federated learning version of Karimireddy et al. [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (top: uniform, middle: D-0.6, bottom: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

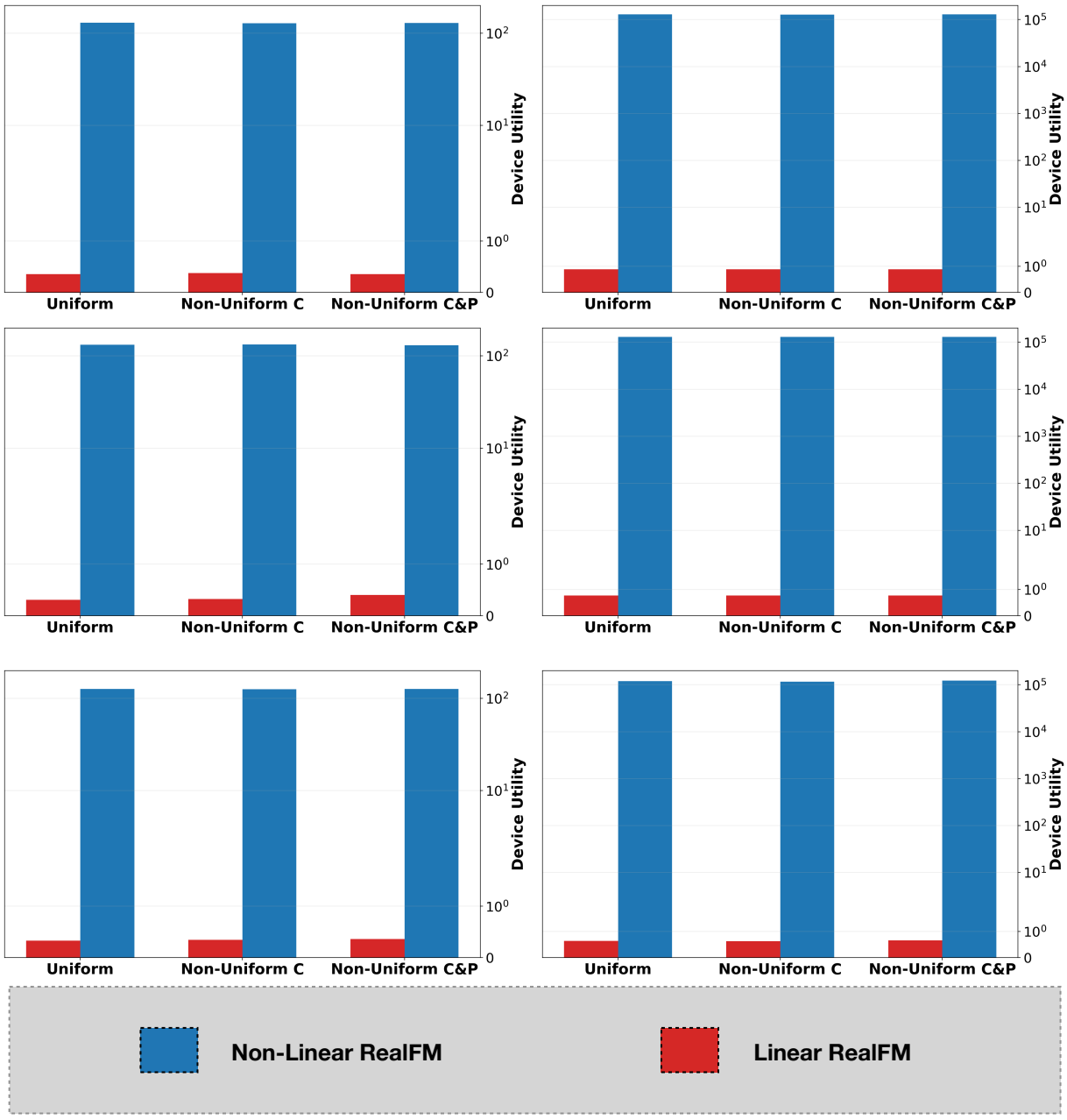


Figure D.5: **Improved Device Utility on CIFAR-10 & MNIST.** REALFM increases device utility on CIFAR-10 (top row) and MNIST (bottom row) for 8 devices compared to baselines. REALFM achieves upwards of 5 magnitudes more utility than a federated learning version of [4], denoted as LINEAR REALFM, across both uniform and various heterogeneous Dirichlet data distributions (left: uniform, center: D-0.6, right: D-0.3) as well as non-uniform costs (C) and accuracy payoff functions (P).

for both datasets in Figure. In both cases, it is shown that utility sharply improves for the central server and participating devices. Data contribution also improves for CIFAR-10 and MNIST. Under all scenarios REALFM performs the best compared to all other baselines.

D.3 Additional Properties

Honest Devices. The setting considered in Chapter 6 assumes honest devices. They only store the rewarded model returned by the server after training. If devices are dishonest, slight alterations to the federated training scheme can be made. Namely, schemes that train varying-sized models such as [226, 227, 228, 229] can be implemented (where model sizes correspond to data contribution size) to alleviate honesty issues.

Corollary 7. *Mechanism $\mathcal{M}_R$ (Equation 6.9) is contribution-maximizing for linear accuracy payoffs. The proof follows Theorem 4.2 in Karimireddy et al. [4].*

Contribution Maximization. The usage of a non-linear accuracy payoff function ϕ_i promotes increased contributions compared to a linear payoff (see Chapter 6.4). However, proof of contribution maximization for non-linear payoffs is not possible, as one cannot tightly bound the composition function $\phi(a + \gamma(m))$ as one can with the linear payoff $a + \gamma(m)$. The accuracy-shaping function is still contribution maximizing when linear.

Accuracy Modeling. The model used for accuracy stems from Example 2.1 in Karimireddy et al. [4], which in turn comes from Theorem 11.8 in Mohri et al. [230]. Below is the mentioned generalization bound,

Proposition 4 (Generalization Bounds, Karimireddy et al. [4] Example 2.1)**.** *Suppose one wants to learn a model h from a hypothesis class $\mathcal{H}$ which minimizes the error over data distribution $\mathcal{D}$, defined to be $R(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[e(h(x), y)]$, for some error function $e(\cdot) \in [0, 1]$. Let such an optimal model have error $(1 - a_{opt}) \leq 1$. Now, given access to $\{(x_l, y_l)\}l \in [m]$ which are m IID samples from $\mathcal{D}$, one can compute the empirical risk minimizer (ERM) as $\hat{h}_m = \arg\min_{h\in\mathcal{H}} \sum_{l\in[m]} e(h(x), y)$. Finally, let $k > 0$ be the pseudo-dimension of the set of functions $\{(x, y) \to e(h(x), y) : h \in \mathcal{H}\}$, which is a measure of the difficulty of the learning task. Then, standard generalization bounds imply that with probability at least 99% over the sampling of the data, the accuracy is at least*

$$1 - R(\hat{h}_m) \geq \left\{\hat{a}(m) := a_{opt} - \frac{\sqrt{2k(2 + \log(m/k)) + 4}}{\sqrt{m}}\right\}. \tag{D.1}$$

A simplified expression for analytical use is,

$$\hat{a}(m) = a_{opt} - 2\sqrt{k/m}. \tag{D.2}$$

D.4 Proofs

D.4.1 Proof of Theorem 9

Theorem 9 (Restated)**.** *Consider a feasible mechanism $\mathcal{M}$ returning utility $[\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i$ to device i ($[\mathcal{M}^U(0; \boldsymbol{m}_{-i})]_i = 0$). Define the utility of a participating device i as,*

$$u_i^r(m_i; \boldsymbol{m}_{-i}) := [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i - c_i m_i.$$

If $u_i^r(m_i, \boldsymbol{m}_{-i})$, is quasi-concave for $m_i \geq m_i^u := \inf\{m_i \mid [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i > 0\}$ and continuous in $\boldsymbol{m}_{-i}$, then a pure Nash equilibrium with $\boldsymbol{m^{eq}}$ data contributions exists such that,

$$u_i^r(\boldsymbol{m^{eq}}) = [\mathcal{M}^U(\boldsymbol{m^{eq}})]_i - c_i \boldsymbol{m_i^{eq}} \geq u_i^r(m_i; \boldsymbol{m_{-i}^{eq}}) = [\mathcal{M}^U(m_i; \boldsymbol{m_{-i}^{eq}})]_i - c_i m_i \textit{ for } m_i \geq 0.$$

Proof. The proof begins by examining two scenarios.

Case 1: $\max_{m_i} u_i^r(m_i; \boldsymbol{m}_{-i}) \leq 0$.

In the case where the marginal cost of producing updates $-c_i m_i$ is so large that the device utility u_i will always be non-positive, the best response $[B(\boldsymbol{m})]_i$ given a set of contributions $\boldsymbol{m}$ for device i is,

$$[B(\boldsymbol{m})]_i = \arg\max_{m_i \geq 0} u_i^r(m_i; \boldsymbol{m}_{-i}) = 0. \tag{D.3}$$

As expected, device i will not perform any updates $\boldsymbol{m_i^{eq}} = 0$. Therefore, Equation 6.3 is fulfilled since as $\max_{m_i} u_i^r(m_i; \boldsymbol{m}_{-i}) \leq 0$ one can see,

$$[\mathcal{M}^U(\boldsymbol{m^{eq}})]_i - c_i \boldsymbol{m_i^{eq}} = [\mathcal{M}^U(0; \boldsymbol{m}_{-i})]_i - c_i(0) = 0 \geq [\mathcal{M}^U(m_i; \boldsymbol{m_{-i}^{eq}})]_i - c_i m_i \text{ for all } m_i \geq 0. \tag{D.4}$$

Case 2: $\max_{m_i} u_i(m_i; \boldsymbol{m}_{-i}) > 0$.

Denote $m_i^u := \inf\{m_i \mid [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i > 0\}$. On the interval of integers $m_i \in [0, m_i^u]$, device i 's utility is non-positive,

$$u_i^r(m_i; \boldsymbol{m}_{-i}) = [\mathcal{M}^U(m_i; \boldsymbol{m_{-i}})]_i - c_i m_i = -c_i m_i \leq 0, \quad \forall m_i \in [0, m_i^u]. \tag{D.5}$$

For $m_i \geq m_i^u$, it is true that $u_i(m_i; \boldsymbol{m})$ is quasi-concave. Let the best response for a given set of

contributions $\boldsymbol{m}_{-i}$ for device i be formally defined as,

$$[B(\boldsymbol{m})]_i := \arg\max_{m_i \geq 0} u_i^r(m_i; \boldsymbol{m}_{-i}) = \arg\max_{m_i \geq 0} [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i - c_i m_i. \tag{D.6}$$

Suppose there exists a fixed point $\tilde{\boldsymbol{m}}$ to the best response, $\tilde{\boldsymbol{m}} \in B(\tilde{\boldsymbol{m}})$. This would mean that $\tilde{\boldsymbol{m}}$ is an equilibrium since, by Equation D.6, for any $m_i \geq 0$,

$$[\mathcal{M}^U(\tilde{m}_i; \tilde{\boldsymbol{m}}_{-i})]_i - c_i \tilde{m}_i \geq [\mathcal{M}^U(m_i; \boldsymbol{m}_{-i})]_i - c_i m_i. \tag{D.7}$$

Thus, now one must show that B has a fixed point (which is subsequently an equilibrium). To do so, a convex and compact search space is first determined. As detailed in Case 1, $u_i^r(0, \boldsymbol{m}_{-i}) = 0$. Therefore, one can bound $0 \leq \max_{m_i} u_i^r(m_i, \boldsymbol{m}_{-i})$. Since $\mathcal{M}$ is feasible (Definition 7), $\mathcal{M}(\boldsymbol{m})$ is bounded above by $\mathcal{M}_{max}^U$. Thus,

$$0 \leq \max_{m_i} u_i^r(m_i; \boldsymbol{m}_{-i}) \leq \mathcal{M}_{max}^U - c_i m_i. \tag{D.8}$$

Rearranging yields $m_i \leq \mathcal{M}_{max}^U / c_i$. Since $m_i \geq m_i^u$, one can restrict the search space to $\mathcal{C} := \prod_j [m_j^u, \mathcal{M}_{max}^U / c_j] \subset \mathbb{R}^n$, where the best response mapping is now over $B : \mathcal{C} \to 2^{\mathcal{C}}$.

Lemma 11 (Kakutani's Theorem)**.** *Consider a multi-valued function $F : \mathcal{C} \to 2^{\mathcal{C}}$ over convex and compact domain $\mathcal{C}$ for which the output set $F(\boldsymbol{m})$ (i) is convex and closed for any fixed $\boldsymbol{m}$, and (ii) changes continuously as $\boldsymbol{m}$ changes. For any such F, there exists a fixed point $\boldsymbol{m}$ such that $\boldsymbol{m} \in F(\boldsymbol{m})$.*

Since within this interval of m_i $u_i^r(m_i, \boldsymbol{m}_{-i})$ is quasi-concave, $B(\boldsymbol{m})$ must be continuous

in $\boldsymbol{m}$ (from Acemoglu and Ozdaglar [231]). Now by applying Lemma 11, Kakutani's fixed point theorem, there exists such a fixed point $\tilde{\boldsymbol{m}}$ such that $\tilde{\boldsymbol{m}} \in B(\tilde{\boldsymbol{m}})$ where $\tilde{\boldsymbol{m}}_i \geq m_i^u$. Since $\max_{m_i} u_i^r(m_i; \boldsymbol{m}_{-i}) > 0$ and $u_i^r(m_i; \boldsymbol{m}_{-i}) \leq 0$ for $m_i \in [0, m_i^u]$, $\tilde{\boldsymbol{m}}_i$ is certain to not fall within $[0, m_i^u]$ $\forall i$ due to the nature of the arg max in Equation D.6. Therefore, Equation D.7 holds as the fixed point $\tilde{\boldsymbol{m}}$ exists and is the equilibrium of $\mathcal{M}$. □

D.4.2 Proof of Theorem 10

Theorem 10 (Restated)**.** *Consider a device i with marginal cost per data point c_i, accuracy function $\hat{a}_i(m)$ satisfying Assumption 5, and accuracy payoff ϕ_i satisfying Assumption 6. This device will collect the following optimal amount of data m_i^o:*

$$m_i^o = \begin{cases} 0 & \textit{if } \max_{m_i \geq 0} u_i(m_i) \leq 0, \\ m^*, \textit{ such that } \phi_i'(\hat{a}_i(m^*)) \cdot \hat{a}_i'(m^*) = c_i & \textit{else.} \end{cases}$$

Proof. Let $m_0 := \sup\{m \mid \hat{a}_i(m) = 0\}$ (the point where $a_i(m)$ begins to increase from 0 and become equivalent to $\hat{a}_i(m)$). Thus, $\forall m_i > m_0, a_i(m_i) = \hat{a}_i(m_i) > 0$. Also, given Assumptions 5 and 6 and Equation 6.7, $u_i(0) = 0$. The derivative of Equation 6.7 for device i is,

$$u_i'(m_i) = \phi_i'(a_i(m_i)) \cdot a_i'(m_i) - c_i. \tag{D.9}$$

Case 1: $\max_{m_i \geq 0} u_i(m_i) \leq 0$.

Each device i starts with a utility of 0 since by Assumptions 5 and 6 $u_i(0) = 0$. Since $\max_{m_i \geq 0} u_i(m_i) \leq 0$, there is no utility gained by device i to contribute more data. Therefore,

the optimal amount of contributions remains at zero: $m_i^* = 0$.

Case 2: $\max_{m_i \geq 0} u_i(m_i) > 0$.

Sub-Case 1: $0 \leq m_i \leq m_0$. By definition of m_0, $a_i(m_i) = 0 \ \forall m_i \in [0, m_0]$. Therefore, from Equation 6.7 and Assumptions 5 and 6, $u_i(m_i) = -c_i m_i < 0 \ \forall m_i \in (0, m_0]$. Since $u_i(0) = 0$ and $u_i(m_i) < 0$ for $m_i > 0$, device i will not collect any contribution: $m_i^* = 0$.

Sub-Case 2: $m_i > m_0$. Since $\forall m_i > m_0, a_i(m_i) = \hat{a}_i(m_i) > 0$, Equation D.9 becomes,

$$u_i'(m_i) = \phi_i'(\hat{a}_i(m_i)) \cdot \hat{a}_i'(m_i) - c_i. \tag{D.10}$$

This sub-case begins by showing that $\phi_i(\hat{a}_i(m_i))$ is bounded. By Assumption 5, $\hat{a}_i(m_i) < a_{opt}^i < 1 \ \forall m_i$. Thus, $\phi_i(\hat{a}_i(m_i)) < \phi_i(a_{opt}^i) < \infty \ \forall m_i$ since $\hat{a}_i(m_i)$ and ϕ_i are non-decreasing and continuous by Assumptions 5 and 6. Due to $\phi_i(\hat{a}_i(m_i))$ being concave, non-decreasing, and bounded (by Assumption 6), by Equation D.10, the limit of its derivative is,

$$\lim_{m_i \to \infty} \phi_i'(\hat{a}_i(m_i)) \cdot \hat{a}_i'(m_i) = 0 \implies \lim_{m_i \to \infty} u_i'(m_i) = -c_i < 0. \tag{D.11}$$

Since $\phi_i(\hat{a}_i(m_i))$ is concave and non-decreasing, its gradient $\phi_i'(\hat{a}_i(m_i)) \cdot \hat{a}_i'(m_i)$ is maximized when $m_i = m_0$ and is non-increasing afterwards. Using the maximal derivative location $m_i = m_0$ in union with Case 2 ($\max_{m_i \geq 0} u_i(m_i) > 0$) and Sub-Case 2 ($u_i(m) \leq 0 \ \forall m \in [0, m_0]$) yields $u_i'(m_0) > 0$ (the derivative must be positive in order to increase utility above 0).

Now that $u_i'(m_0) > 0$, $\lim_{m_i \to \infty} u_i'(m_i) < 0$, and $\phi_i'(\hat{a}_i(m_i)) \cdot \hat{a}_i'(m_i)$ is non-increasing, there must exist a maximum $m_i = m^*$ such that $\phi_i'(\hat{a}_i(m^*)) \cdot \hat{a}_i'(m^*) = c_i$. □

Corollary 8. *For uniform accuracy-payoff functions & data distributions:* $u_j(m_j^o) \geq u_k(m_k^o)$, $\phi_j(a_j(m)) = \phi_k(a_k(m))$, *and* $m_j^o \geq m_k^o$ *if marginal costs satisfy* $c_j \leq c_k \; \forall j, k \in [n]$.

Proof. With uniform payoff functions, each device i's utility and utility derivative become

$$u_i(m_i) = \phi(a_i(m_i)) - c_i m_i, \quad u_i'(m_i) = \phi'(a_i(m_i)) \cdot a_i'(m_i) - c_i. \tag{D.12}$$

Due to $\phi_i(a_i(m))$ being concave and non-decreasing, its derivative $\phi'(a_i(m_i)) \cdot a_i'(m_i)$ is non-negative and non-increasing. Let m_k^o be the optimal amount of data contribution by device k.

Case 1: $c_j = c_k$. By Equation D.12, if $c_j = c_k$ then $0 = u_k'(m_k^o) = u_j'(m_k^o)$. This implies $m_k^o = m_j^o$ and subsequently $u_k(m_k^o) = u_j(m_k^o)$.

Case 2: $c_j < c_k$. By Equation D.12, if $c_j < c_k$, then $0 = u_k'(m_k^o) < u_j'(m_k^o)$. Since $\phi(a_i(m_i))$ is concave and non-decreasing, its derivative is non-increasing with its limit going to 0. Therefore, more data $\epsilon > 0$ must be collected in order for u_j' to reach zero (*i.e.* $u_j'(m_k^o + \epsilon) = 0$). This implies that $m_j^o = m_k^o + \epsilon > m_k^o$. Furthermore, since $0 = u_k'(m_k^o) < u_j'(m_k^o)$, device j's utility is still increasing at m_k^o and is maximized at m_j^o. This implies $u_j(m_j^o) > u_k(m_k^o)$. □

D.4.3 Proof of Theorem 11

Theorem 11 (Restated)**.** *Consider a device* i *with marginal cost* c_i *and accuracy payoff function* ϕ_i *satisfying Assumptions 6 and 7. Denote device* i*'s optimal local data contribution as* m_i^o *and its subsequent accuracy* $\bar{a}_i := a_i(m_i^o)$. *Define the derivative of* $\phi_i(a)$ *with respect to* a *as* $\phi_i'(a)$. *For any* $\epsilon \to 0^+$ *and marginal server reward* $r(\boldsymbol{m}) \geq 0$, *device* i *has the following accuracy-shaping function* $\gamma_i(m)$ *for* $m \geq m_i^o$,

$$\gamma_i := \begin{cases} \frac{-\phi_i'(\bar{a}_i)+\sqrt{\phi_i'(\bar{a})^2+2\phi_i''(\bar{a}_i)(c_i-r(\boldsymbol{m})+\epsilon)(m-m_i^o)}}{\phi_i''(\bar{a}_i)}, \\ \\ \frac{(c_i-r(\boldsymbol{m})+\epsilon)(m-m_i^o)}{\phi_i'(\bar{a}_i)} \quad \textit{if } \phi_i''(\bar{a}_i)=0 \end{cases}$$

Given the defined $\gamma_i(m)$, the following inequality is satisfied for $m \in [m_i^o, m_i^*]$,

$$\phi_i(\bar{a}_i + \gamma_i(m)) - \phi_i(\bar{a}_i) > (c_i - r(\boldsymbol{m}))(m - m_i^o).$$

The new optimal contribution for each device i becomes $m_i^* := \{m \geq m_i^o \mid a_C(m + \sum_{j \neq i} m_j) = \bar{a}_i + \gamma_i(m)\}$. Device i's data contribution increases $m_i^* \geq m_i^o$ for any contribution $\boldsymbol{m}_{-i}$.

Proof. By the mean value version of Taylor's theorem,

$$\phi_i(\bar{a} + \gamma_i) = \phi_i(\bar{a}) + \gamma_i \phi_i'(\bar{a}) + 1/2\gamma_i^2 \phi_i''(z), \quad \text{for some } z \in [\bar{a}, \bar{a} + \gamma_i]. \tag{D.13}$$

Since $\phi_i(a)$ is both increasing and convex with respect to a,

$$\phi_i(\bar{a} + \gamma_i) - \phi_i(\bar{a}) \geq \gamma_i \phi_i'(\bar{a}) + 1/2\gamma_i^2 \phi_i''(\bar{a}). \tag{D.14}$$

In order to ensure $\phi_i(\bar{a} + \gamma) - \phi_i(\bar{a}) > (c_i - r(\boldsymbol{m}))(m - m_i^o)$, one must select γ such that,

$$\gamma_i \phi_i'(\bar{a}) + 1/2\gamma_i^2 \phi_i''(\bar{a}) > (c_i - r(\boldsymbol{m}))(m - m_i^o). \tag{D.15}$$

Case 1: $\phi_i''(\bar{a}) = 0$**.** In this case, Equation D.15 becomes,

$$\gamma_i \phi_i'(\bar{a}) > (c_i - r(\boldsymbol{m}))(m - m_i^o). \tag{D.16}$$

In order for Equation D.15, and thereby Equation 6.12, to hold one selects $\epsilon \to 0^+$ such that,

$$\gamma_i := \frac{(c_i - r(\boldsymbol{m}) + \epsilon)(m - m_i^o)}{\phi_i'(\bar{a})}. \tag{D.17}$$

Case 2: $\phi_i''(\bar{a}) > 0$**.** Determining when the left- and right-hand sides of Equation D.15 are equal is equivalent to solving the quadratic equation for γ_i,

$$\gamma_i = \frac{-\phi_i'(\bar{a}) \pm \sqrt{\phi_i'(\bar{a})^2 + 2\phi_i''(\bar{a})(c_i - r(\boldsymbol{m}))(m - m_i^o)}}{\phi_i''(\bar{a})} \tag{D.18}$$

$$= \frac{-\phi_i'(\bar{a}) + \sqrt{\phi_i'(\bar{a})^2 + 2\phi_i''(\bar{a})(c_i - r(\boldsymbol{m}))(m - m_i^o)}}{\phi_i''(\bar{a})}. \tag{D.19}$$

The second equality follows from γ_i having to be positive. In order for Equation D.15, and thereby Equation 6.12, to hold one selects $\epsilon \to 0^+$ such that,

$$\gamma_i := \frac{-\phi_i'(\bar{a}) + \sqrt{\phi_i'(\bar{a})^2 + 2\phi_i''(\bar{a})(c_i - r(\boldsymbol{m}) + \epsilon)(m - m_i^o)}}{\phi_i''(\bar{a})} \tag{D.20}$$

As a quick note, for $m = m_i^o$ one can immediately see that $\gamma_i(m) = 0$. To finish the proof, now that Equation 6.12 is proven to hold for the prescribed γ_i, device i is incentivized to contribute more as the added utility $\phi_i(\bar{a} + \gamma_i) - \phi_i(\bar{a})$ is larger than the incurred cost $(c_i - r(\boldsymbol{m}))(m - m_i^o)$. There is a limit to this incentive, however. The maximum value that γ_i can be is bounded by the accuracy from all contributions: $a_i(m_i^o) + \gamma_i \leq a_C(\sum_j m_j)$. The existence of the bound is

guaranteed by Assumption 7. Thus, device i reaches a new optimal contribution m_i^* which is determined by,

$$m_i^* := \{m \geq m_i^o \mid a_C(m + \sum_{j \neq i} m_j) = a_i(m_i^o) + \gamma_i(m)\} \geq m_i^o. \tag{D.21}$$

□

D.4.4 Proof of the Main Theorem

Theorem 12 (Restated)**.** *RealFM, $\mathcal{M}_R$ (Equation 6.9), performs accuracy-shaping with γ_i defined in Theorem 11 for each device $i \in [n]$ and some $\epsilon \to 0^+$. As such, $\mathcal{M}_R$ is Individually Rational (IR) and has a unique Nash equilibrium at which device i will contribute $m_i^* \geq m_i^o$ updates, thereby eliminating free riding. Since $\mathcal{M}_R$ is IR, devices are incentivized to participate, as they never receive more utility by not participating.*

Proof. Existence of a unique Nash equilibrium is proven by showcasing how the mechanism $\mathcal{M}_R$ fulfills the criteria laid out in Theorem 9. The criteria in Theorem 9 largely surrounds the utility of a participating device i,

$$u_i^r(m_i; \boldsymbol{m}_{-i}) := [\mathcal{M}_R^U(m_i; \boldsymbol{m}_{-i})]_i - c_i m_i. \tag{D.22}$$

Feasibility. Before beginning, it should be noted that $\mathcal{M}_R$ trivially satisfies the only non-utility requirement that $[\mathcal{M}_R^U(0; \boldsymbol{m}_{-i})]_i = 0$ (as $a_i(0) = \phi_i(0) = 0$). As shown in Equation 6.9, $\mathcal{M}_R$ returns accuracies between 0 and $a_i(\sum \boldsymbol{m})$ to all devices. This satisfies the bounded accuracy requirement. Furthermore, the utility provided by the mechanism $\mathcal{M}_R^U$ is bounded as well.

Since a_{opt} is the largest attained accuracy by the defined accuracy function $\hat{a}_i(m)$ and $a_{opt} < 1$, the maximum utility is $\phi_i(a_{opt}) < \infty$. Now, that $\mathcal{M}_R$ is proven to be Feasible, one only needs the following to prove that $\mathcal{M}_R$ has a pure equilibrium: **(1)** $u_i^r(m_i; \boldsymbol{m}_{-i})$ is continuous in $\boldsymbol{m}_{-i}$ and **(2)** quasi-concave for $m_i \geq m_i^u := \inf\{m_i \mid [\mathcal{M}_R^U(m_i; \boldsymbol{m}_{-i})]_i > 0\}$.

Continuity. By definition of $u_i^r(m_i; \boldsymbol{m}_{-i})$, Equation D.22, one only needs to consider $[\mathcal{M}_R^U(m_i; \boldsymbol{m}_{-i})]_i$ since that is the only portion affected by $\boldsymbol{m}_{-i}$. By definition of the utility returned by the mechanism $\mathcal{M}_R$, shown in Equation 6.10, no discontinuities arise for a fixed m_i and varying $\boldsymbol{m}_{-i}$. By assumptions on continuity in Assumptions 5 and 6, $\phi_i(a_i(m))$ is continuous for all m. Thus, for non-zero utility (zero utility would lead to zero reward), the marginal monetary reward function $r(\boldsymbol{m})$ found in Equation 6.13 is continuous. Therefore, each piecewise component of $[\mathcal{M}_R^U(m_i; \boldsymbol{m}_{-i})]_i$ is continuous since they are sums of continuous functions. Finally, it is shown that the piecewise functions connect with each other continuously. The accuracy-shaping function γ_i is defined such that $\gamma_i(m_i^o) = 0$ and $a_i(m_i^o) + \gamma_i(m_i^*) = a_i(\sum \boldsymbol{m})$, which finishes proof of continuity.

Quasi-Concavity. For all values of $m_i \geq m_i^u$, the mechanism $\mathcal{M}_R$ produces positive utility. By construction, the mechanism $\mathcal{M}_R$ is strictly increasing for $m_i \geq m_i^u$. The mechanism $\mathcal{M}_R$ returns varying utilities within three separate intervals. While piece-wise, these intervals are continuous and $\mathcal{M}_R$ is strictly increasing with respect to m_i in each. The first interval, consisting of the concave function $\phi_i(a_i(m_i))$, is quasi-concave by construction. The second interval consists of a linear function $r(\boldsymbol{m}) \cdot (m_i - m_i^o)$ added to a quasi-concave $\phi_i(a_i(m_i) + \gamma_i(m_i))$ function, resulting in a quasi-concave function (note that $\phi_i(\hat{a}_i(m_i))$ is concave). Finally, the third interval consists of a linear function $r(\boldsymbol{m}) \cdot (m_i - m_i^*)$ added to a concave function $\phi_i(a_i(m_i))$, which is also quasi-concave. In sum, this makes $[\mathcal{M}_R^U(m_i; \boldsymbol{m}_{-i})]_i$ a quasi-concave function. Since $-c_i m_i$

is a linear function, the utility of a participating device $u_i^r(m_i; \boldsymbol{m}_{-i})$ will also be quasi-concave function, as the sum of a linear and quasi-concave function is quasi-concave.

Existence of Pure Equilibrium with Increased Data Contribution. Since $[\mathcal{M}_R^U(\boldsymbol{m})]_i$ satisfies feasibility, continuity, and quasi-concavity requirements, $\mathcal{M}_R$ is guaranteed to have a pure Nash equilibrium by Theorem 9. Furthermore, since $\mathcal{M}_R$ performs accuracy-shaping with γ_i prescribed in Theorem 9, it is guaranteed that each device i will produce $m_i^* \geq m_i^o$ updates.

Individually Rational (IR). $\mathcal{M}_R$ is proven IR by looking at piecewise portions of Eq. 6.9:

Case 1: $m_i \leq m_i^o$ *(Free-Riding).* When $m_i \leq m_i^o$, a device would attempt to provide as much or less than the amount of contribution which is locally optimal. The hope for such strategy would be free-riding: enjoy the performance of a well-trained model as a result of federated training while providing few (or zero) data points in order to save costs. The mechanism avoids the free rider problem trivially by returning a model with an accuracy that is proportional to the amount of data contributed by the device. This is shown in Equation 6.9, as devices receive a model with accuracy $a_i(m_i)$ if $m_i \leq m_i^o$ (*i.e.,* devices are rewarded with a model equivalent to one that they could've trained themselves if they fail to contribute an adequate amount of data). In this case, devices receive the same model accuracy as they would've on their own and thus IR is satisfied in this case.

Case 2: $m_i \in (m_i^o, m_i^*]$. Via the results of Theorem 11, the accuracy of the model returned by $\mathcal{M}_R$ when $m_i \in (m_i^o, m_i^*]$ is greater than a model trained by device i on m_i local contributions. Mathematically, this is described as $a^r(m_i) = a_i(m_i) + \gamma_i(m_i) > a_i(m_i)$ for $m_i \in (m_i^o, m_i^*]$. Since ϕ_i is increasing, IR must hold as accuracy from $\mathcal{M}_R$ outstrips local training accuracy.

Case 3: $m_i \geq m_i^*$. By Theorem 11, by definition of m_i^* when $m_i = m_i^*$ then the accuracy of a returned model by $\mathcal{M}_R$ is equal to $a_i(\sum_j m_j)$. Therefore, given a fixed set of contributions from

all other devices $\boldsymbol{m_{-i}}$, device i will still attain a model with accuracy $a_i(\sum_j m_j)$ for $m_i \geq m_i^*$ (since the limits of accuracy shaping have been reached for the given contributions). Due to this, IR trivially holds as $a_i(\sum_j m_j) \geq a_i(m_i)$. □

Appendix E: Appendix to Chapter 7

E.1 Additional Experiments

Within this section, the computed PDFs and CDFs in Corollaries 4 and 5 are empirically verified. To accomplish this, V_i and λ_i are sampled and their product is computed fifty million times. Then, the PDF and CDF of the resultant products are plotted and compared with the theoretical PDF and CDF. The theoretical PDF and CDF for Corollary 4 are defined in Equations (E.20) and (E.21), while those for Corollary 5 are found in Equations (E.34) and (E.35). The results of these simulations, which validate the computed PDFs and CDFs, are shown in Figures E.1 and E.2. To ensure correctness, testing is performed on varying values of p_ϵ. As expected, the theory lines up exactly with the empirical simulations for both Corollaries as well as across varying p_ϵ.

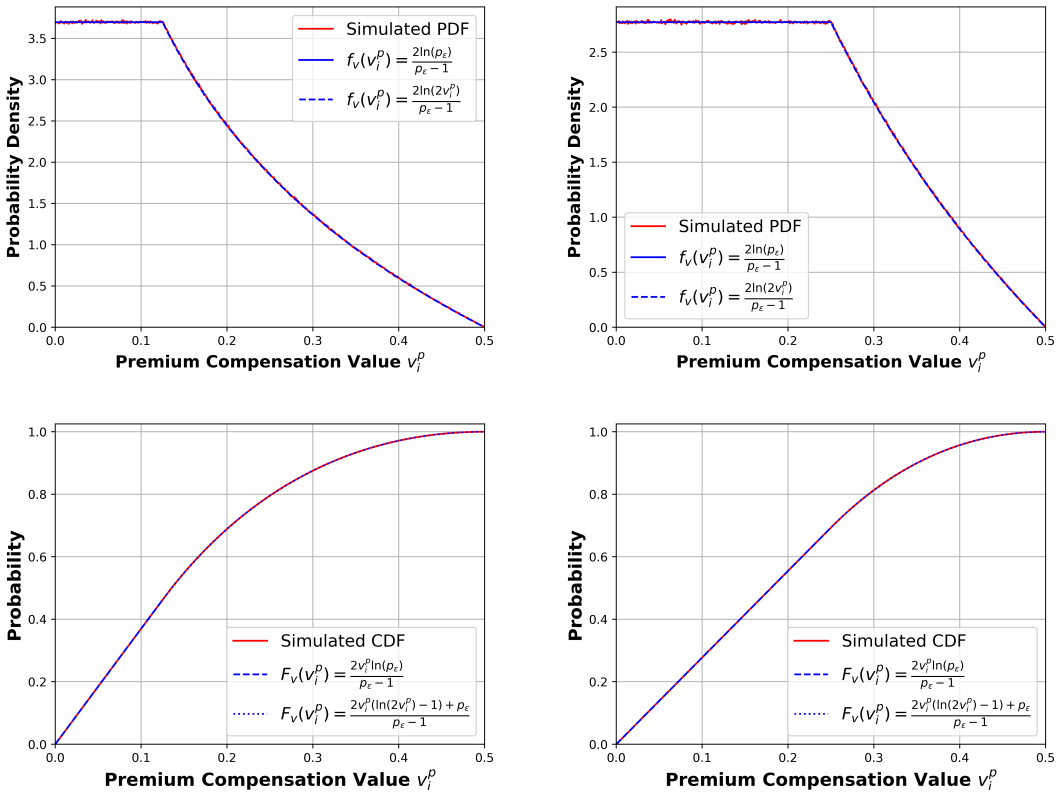


Figure E.1: **Uniform PDF and CDF Validation**. Numerical validation of the derivations for $f_v(v_i^p)$ and $F_v(v_i^p)$, where $v_i^p := V_i\lambda_i$, for V_i and λ_i coming from Uniform distributions (Corollary 4). The price of attaining ϵ is set as $p_\epsilon = 1/4$ (top row) and $p_\epsilon = 1/2$ (bottom row).

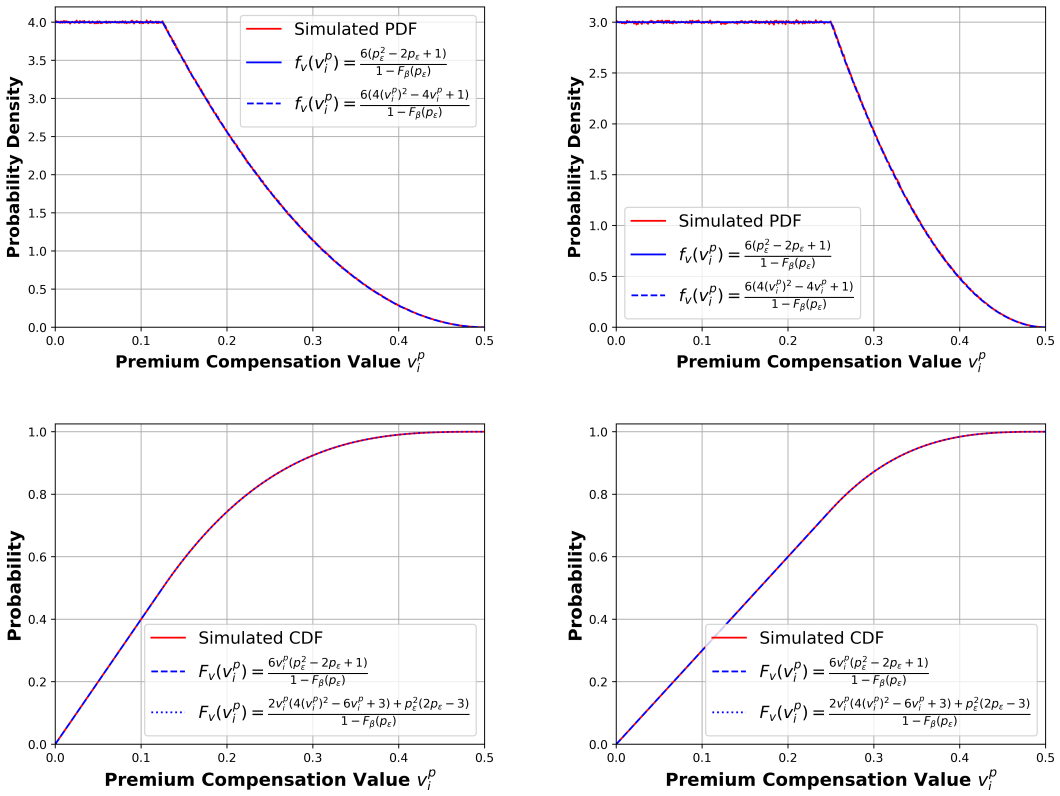


Figure E.2: **Beta PDF and CDF Validation**. Numerical validation of the derivations for $f_v(v_i^P)$ and $F_v(v_i^P)$, where $v_i^p := V_i\lambda_i$, for V_i coming from a Beta distribution and λ_i from a Uniform distributions (Corollary 5). The price of attaining ϵ is set as $p_\epsilon = 1/4$ (top row) and $p_\epsilon = 1/2$ (bottom row).

E.2 Proofs

E.2.1 Proof of Theorem 13

Theorem 13 (Restated)**.** *Under Assumption 9, agents participating in Reserve Thresholding* (7.2) *have an optimal bidding strategy and utility of,*

$$b_i^* = p_\epsilon, \quad u_i(b_i^*; \boldsymbol{b}_{-i}) = v_i^d - p_\epsilon,$$

and submit models with the following compliance level,

$$s_i^* = \begin{cases} \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no model submission)} & \text{else.} \end{cases}$$

Proof. From agent i's utility within Reserve Thresholding, Equation 7.2, it is clear that $u_i(0) = 0$. One can proceed to break the proof up into cases where agents have (1) a deployment value equal to or less than the price of compliance p_ϵ and (2) a deployment value larger than p_ϵ.

Case 1: $v_i^d \leq p_\epsilon$. From Equation 7.2, if $v_i^d \leq p_\epsilon$, an agent will not attain positive utility,

$$\max_{b_i \in (0,1]} v_i^d \cdot 1_{b_i \geq p_\epsilon} - b_i \leq \max_{b_i \in (0,1]} p_\epsilon \cdot 1_{b_i \geq p_\epsilon} - b_i = \max_{b_i \in [p_\epsilon, 1]} p_\epsilon - b_i = p_\epsilon - p_\epsilon = 0. \tag{E.1}$$

$$\arg\max_{b_i \in (0,1]} u_i(b_i) = p_\epsilon. \tag{E.2}$$

For an agent with deployment value at most equal to p_ϵ, the upper bound on attainable utility when it participates, *i.e.,* $b_i \in (0, 1]$, is zero (Equation E.1). This maximum utility is attained

when bidding $b_i = p_\epsilon$ (Equation E.2). Thus, agents have nothing to gain by participating, as they already start at zero utility $u_i(0) = 0$. As a result, agents will not submit a model, $s_i^* = M(0) = 0$.

Case 2: $v_i^d > p_\epsilon$. Similar steps to Case 1 above,

$$\max_{b_i \in (0,1]} v_i^d \cdot 1_{b_i \geq p_\epsilon} - b_i > \max_{b_i \in (0,1]} p_\epsilon \cdot 1_{b_i \geq p_\epsilon} - b_i = \max_{b_i \in [p_\epsilon, 1]} p_\epsilon - b_i = p_\epsilon - p_\epsilon = 0. \tag{E.3}$$

$$b_i^* = \arg\max_{b_i \in (0,1]} u_i(b_i) = p_\epsilon \longrightarrow u_i(b_i^*) = v_i^d - p_\epsilon > 0. \tag{E.4}$$

An agent with deployment value larger than p_ϵ has a maximal utility that is non-negative when participating (Equation E.3). Maximal utility is attained when bidding $b_i^* = p_\epsilon$ (Equation E.4). Furthermore, at this optimal bid, the corresponding compliance level is $s_i^* = M^{-1}(p_\epsilon) = \epsilon$. □

E.2.2 Proof of Theorem 14

Theorem 14 (Restated)**.** *Agents participating in* CIRCA (7.6) *follow an optimal bidding strategy,*

$$\hat{b}_i^* := p_\epsilon + v_i^p F_v(v_i^p) - \int_0^{v_i^p} F_v(z)dz > p_\epsilon,$$

where $F_v(\cdot)$ denotes the cumulative density function of the random variable corresponding to the premium reward $v_i^p = V_i \lambda_i$.

Proof. Before beginning, it should be noted that each agent i cannot alter its own valuation v_i^p for winning the all-pay auction. Each valuation is private (unknown by other agents) and predetermined: total reward V_i and partition factor λ_i are randomly selected from a given distribution $\mathcal{D}$ on $[0, 1]$ and $[0, 1/2]$ respectively by "nature". The cumulative distribution function for the auction reward $v_i^p = V_i \lambda_i$ is denoted as $F_v(\cdot)$ and the probability distribution function as $f_v(\cdot)$.

From Equation (7.6), an agent i that does not participate (*i.e.,* $b_i = 0$) receives no utility,

$$u_i(0) = 0. \tag{E.5}$$

An agent receives negative utility if its bid does not reach the price of compliance p_ϵ,

$$\max_{b_i \in (0, p_\epsilon)} u_i(b_i) < 0. \tag{E.6}$$

Consequently, rational agents will either opt not to participate (notated as the set of agents N) or participate (notated as the set of agents P) and bid at least p_ϵ. These groups are defined as,

$$N = \{i \in [n] \mid \max_{b_i \in [0,1]} u_i(b_i) \leq 0\}, \tag{E.7}$$

$$P = \{i \in [n] \mid \max_{b_i \in [0,1]} u_i(b_i) > 0\}. \tag{E.8}$$

From here, one only needs to focus on agents $i \in P$ which participate (*i.e.,* have utility to be gained by participating). As a result from Equations (E.5) and (E.6), Equation (E.8) transforms into,

$$P = \{i \in [n] \mid \max_{b_i \in [p_\epsilon, 1]} u_i(b_i) > 0\}. \tag{E.9}$$

The result of (E.9) is that participating agents bid at least p_ϵ. This is important, as every participating agent knows that all rival agents j they will possibly be compared against have $b_j \in [p_\epsilon, 1]$. Agents can dictate how much they bid, and the auction is designed to ensure that agents bid in proportion to their valuation.

Following previous literature [206, 207, 208], a *monotone increasing* bidding function $b(\cdot)$:

$[0, 1/2] \rightarrow [p_\epsilon, 1]$ that each agent follows is desired. It will be proven that each agent i's best strategy is to bid its own valuation $b(v_i^p)$ irrespective of other agent bids (Nash Equilibrium). Using a bidding function transforms agent utility,

$$u_i(b_i) = \big(v_i^d + v_i^p \cdot 1_{(\text{if } i \text{ wins auction})}\big) \cdot \underbrace{1_{(\text{if } b_i \geq p_\epsilon)}}_{\text{satisfied for agents } i \in P} - b_i,$$

$$= \mathbb{P}\big(b(b_i) > b(b_j)\big) v_i^p - b(b_i) + v_i^d, \quad b_j \sim \text{randomly sampled agent bid.} \tag{E.10}$$

Since $b(x)$ is monotone increasing up to 1, agents bidding $b = 1$ automatically win, the utility function above can be simplified as,

$$u_i(b_i) = v_i^p \mathbb{P}\big(b_i > b_j\big) - b(b_i) + v_i^d, \quad b_j \sim \text{randomly sampled agent bid},$$

$$= v_i^p F_v(b_i) - b(b_i) + v_i^d. \tag{E.11}$$

Taking the derivative and setting it equal to zero yields,

$$\frac{d}{db_i} u_i(b_i) = v_i^p f_v(b_i) - b'(b_i) = 0. \tag{E.12}$$

As agents bid in proportion to their valuation, the first-order equilibrium conditions are solved at $b_i = v_i^p$,

$$b'(v_i^p) = v_i^p f_v(v_i^p). \tag{E.13}$$

Integrating by parts, and knowing ϵ is the minimum bid ($b(0) = p_\epsilon$), reveals the optimal bidding function,

$$
\begin{aligned}
b(v_i^p) - b(0) &= \int_0^x v_i^p f_v(v_i^p) dv_i^p, \\
b(v_i^p) - p_\epsilon &= v_i^p F_v(v_i^p) - \int_0^{v_i^p} F_v(z) dz, \\
\hat{b}_i^* = b(v_i^p) &:= p_\epsilon + v_i^p F_v(v_i^p) - \int_0^{v_i^p} F_v(z) dz. \qquad \text{(E.14)}
\end{aligned}
$$

□

E.2.3 Proof of Corollary 4

Corollary 4 (Restated)**.** *Under Assumption 9, for agents having total value V_i and scaling factor λ_i both stemming from a Uniform distribution, with $v_i^d = (1-\lambda_i)V_i$, and $v_i^p = \lambda_i V_i$, their optimal bid and utility participating in* CIRCA (7.6) *are,*

$$
b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2(\ln(2v_i^p) - 1/2) + p_\epsilon^2}{8(p_\epsilon - 1)} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}, \end{cases}
$$

$$
u_i(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} \frac{2(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} + v_i^d - b_i^* & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ \frac{2(v_i^p)^2(\ln(2p_\epsilon) - 1) + p_\epsilon}{p_\epsilon - 1} + v_i^d - b_i^* & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}. \end{cases}
$$

Participating agents submit models with compliance,

$$
s_i^* := \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no submission)} & \text{else}. \end{cases}
$$

Proof. Let $v_i^p := V_i \lambda_i$, where $V_i \sim U[p_\epsilon, 1]$ and $\lambda_i \sim U[0, 1/2]$. The reason that V_i is within the interval $[p_\epsilon, 1]$, is that all participating agents must have a value of at least p_ϵ or else they would not have rationale to bid. The smallest value of V_i such that this is possible is p_ϵ, so it is the lower bound on this interval. The first goal is to find the PDF of v_i^p, $f_{v_i^p}(\cdot)$.

To begin, $f_{v_i^p}(\cdot)$ is determined by using a change of variables. For the product of two random variables $v = x_1 \cdot x_2$, let $y_1 = x_1 \cdot x_2$ and $y_2 = x_2$. Thus, inversely that $x_2 = y_2$ and $x_1 = y_1/y_2$. Since x_1 and x_2 are independent and both uniform,

$$f_{y_1,y_2}(x_1, x_2) = (\frac{1}{1 - p_\epsilon})(\frac{1}{1/2 - 0}) = \frac{2}{1 - p_\epsilon}. \tag{E.15}$$

When using the change of variables this becomes,

$$f_{y_1,y_2}(y_1, y_2) = f_{y_1,y_2}(x_1, x_2)|J| = \frac{2}{(1 - p_\epsilon)y_2}, \quad |J| = \left| \begin{pmatrix} 1/y_2 & -y_1/y_2^2 \\ 0 & 1 \end{pmatrix} \right| = 1/y_2 \tag{E.16}$$

Marginalizing out y_2 (a non-negative value) yields,

$$f_{y_1}(y_1) = \int_0^\infty \frac{2}{(1 - p_\epsilon)y_2} dy_2. \tag{E.17}$$

The bounds of integration depend upon the value of y_1. The change of variable to the (y_1, y_2) space, where $0 \leq y_1, y_2 \leq 1/2$, results in a new region of possible variable values. This region is a triangle bounded by the three vertices: $(0, 0)$, $(p_\epsilon/2, 1/2)$, and $(1/2, 1/2)$. Thus, the bounds of

marginalization depend upon the value of y_1. For $0 \leq y_1 \leq p_\epsilon/2$,

$$f_{y_1}(y_1) = \int_{y_1}^{y_1/p_\epsilon} \frac{2}{(1-p_\epsilon)y_2} dy_2 = \frac{2}{(1-p_\epsilon)} [\ln(y_2)\big|_{y_1}^{y_1/p_\epsilon}] = \frac{2\ln(p_\epsilon)}{(p_\epsilon - 1)}. \tag{E.18}$$

For $p_\epsilon \leq y_1 \leq 1/2$,

$$f_{y_1}(y_1) = \int_{y_1}^{1/2} \frac{2}{(1-p_\epsilon)y_2} dy_2 = \frac{2}{(1-p_\epsilon)} [\ln(y_2)\big|_{y_1}^{1/2}] = \frac{2\ln(2y_1)}{(p_\epsilon - 1)}. \tag{E.19}$$

Thus, as a piecewise function the PDF is formally,

$$f_{y_1}(y_1) = \begin{cases} \frac{2\ln(p_\epsilon)}{(p_\epsilon - 1)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \\ \frac{2\ln(2y_1)}{(p_\epsilon - 1)} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.20}$$

Now, the CDF is determined through integration,

$$F_{y_1}(y_1) = \int_0^{y_1} f_{y_1}(y_1) dy_1 = \begin{cases} \frac{2y_1\ln(p_\epsilon)}{(p_\epsilon - 1)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \\ \frac{2y_1(\ln(2y_1) - 1) + p_\epsilon}{(p_\epsilon - 1)} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.21}$$

One can integrate the CDF to get,

$$\int_0^{y_1} F_{y_1}(y_1) = \begin{cases} \frac{y_1^2\ln(p_\epsilon)}{(p_\epsilon - 1)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \\ \frac{4y_1^2(2\ln(2y_1) - 3) + 8y_1 p_\epsilon - p_\epsilon^2}{8(p_\epsilon - 1)} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.22}$$

Plugging all of this back into Equation (7.7) yields,

$$\begin{aligned}\hat{b}_i^* &= \begin{cases} p_\epsilon + v_i^p \frac{2v_i^p \ln(p_\epsilon)}{p_\epsilon - 1} - \frac{(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1}, \\ p_\epsilon + v_i^p \frac{2v_i^p (\ln(2v_i^p) - 1) + p_\epsilon}{(p_\epsilon - 1)} - \frac{4(v_i^p)^2 (2\ln(2v_i^p) - 3) + 8v_i^p p_\epsilon - p_\epsilon^2}{8(p_\epsilon - 1)}, \end{cases} \\ &= \begin{cases} p_\epsilon + \frac{(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} & \text{if } 0 \leq v_i^p \leq \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2 (\ln(2v_i^p) - 1/2) + p_\epsilon^2}{8(p_\epsilon - 1)} & \text{if } \frac{p_\epsilon}{2} \leq v_i^p \leq \frac{1}{2}. \end{cases} \end{aligned} \tag{E.23}$$

Since b_i cannot be larger than 1, the bidding function is capped at 1 via,

$$b_i^* := \min\{\hat{b}_i^*, 1\}. \tag{E.24}$$

The utility gained by agent i for using such a bidding function is,

$$u(b_i^*) = \begin{cases} v_i^d - b_i^* + \frac{2(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} & \text{for } 0 \leq v_i^p \leq \frac{p_\epsilon}{2}, \\ v_i^d - b_i^* + \frac{2(v_i^p)^2 (\ln(2v_i^p) - 1) + p_\epsilon}{(p_\epsilon - 1)} & \text{for } \frac{p_\epsilon}{2} \leq v_i^p \leq 1/2. \end{cases} \tag{E.25}$$

When this utility is larger than 0, the agent will participate otherwise the agent will not submit a model to the regulator. Finally, one can find the optimal compliance level by using Assumption 9,

$$s_i^* := M^{-1}\big(b_i^*\big). \tag{E.26}$$

□

E.2.4 Proof of Corollary 5

Corollary 5 (Restated)**.** *Under Assumption 9, let agents have total value V_i and scaling factor λ_i stem from Beta ($\alpha = \beta = 2$) and Uniform distributions respectively, with $v_i^d = (1-\lambda_i)V_i$ and $v_i^p = \lambda_i V_i$. Denote the CDF of the Beta distribution on $[0,1]$ as $F_\beta(x) = 3x^2 - 2x^3$. The optimal bid and utility for agents participating in* CIRCA *(7.6) are,*

$$b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{3(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2\left(6(v_i^p)^2 - 8v_i^p + 3\right) + p_\epsilon^3(3p_\epsilon - 4)}{8(1 - F_\beta(p_\epsilon))} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}, \end{cases}$$

$$u(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} v_i^d + \frac{6(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ v_i^d + \frac{v_i^p\left(8(v_i^p)^3 - 12(v_i^p)^2 + 6v_i^p + p_\epsilon^2(2p_\epsilon - 3)\right)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } \frac{p_\epsilon}{2} \le v_i^p \le 1/2. \end{cases}$$

Participating agents submit models with compliance,

$$s_i^* = \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no submission)} & \text{else}. \end{cases}$$

Proof. Similar to Corollary 4, $f_{v_i^p}(\cdot)$ is determined using a change of variables. For the product of two random variables $v = x_1 \cdot x_2$, let $y_1 = x_1 \cdot x_2$ and $y_2 = x_2$. Inversely, $x_2 = y_2$ and $x_1 = y_1/y_2$. While x_1 and x_2 are independent, x_1 comes from a Beta distribution and x_2 from a

Uniform one. The PDF and CDF of a Beta distribution, with $\alpha = \beta = 2$, on $[0, 1]$ are defined as,

$$f_\beta(x) := 6x(1-x), \tag{E.27}$$

$$F_\beta(x) := 3x^2 - 2x^3. \tag{E.28}$$

Now, the PDF over y_1, y_2 is defined as,

$$f_{y_1,y_2}(x_1, x_2) = (\frac{6x_1(1-x_1)}{1-F_\beta(p_\epsilon)})(\frac{1}{1/2-0}) = \frac{12x_1(1-x_1)}{1-F_\beta(p_\epsilon)}. \tag{E.29}$$

When using the change of variables this becomes,

$$f_{y_1,y_2}(y_1, y_2) = f_{y_1,y_2}(x_1, x_2)|J| = \frac{12y_1(1-\frac{y_1}{y_2})}{(1-F_\beta(p_\epsilon))y_2^2}, \quad |J| = \left| \begin{pmatrix} 1/y_2 & -y_1/y_2^2 \\ 0 & 1 \end{pmatrix} \right| = 1/y_2 \tag{E.30}$$

Marginalizing out y_2 (a non-negative value) yields,

$$f_{y_1}(y_1) = \frac{12y_1}{1-F_\beta(p_\epsilon)} \int_0^\infty \frac{1}{y_2^2} - \frac{y_1}{y_2^3} dy_2. \tag{E.31}$$

The bounds of integration depend upon the value of y_1. The change of variable to the (y_1, y_2) space, where $0 \leq y_1, y_2 \leq 1/2$, results in a new region of possible variable values. This region is a triangle bounded by the three vertices: $(0, 0)$, $(p_\epsilon/2, 1/2)$, and $(1/2, 1/2)$. Thus, the bounds of

marginalization depend upon the value of y_1. For $0 \leq y_1 \leq p_\epsilon/2$,

$$f_{y_1}(y_1) = \frac{12y_1}{1-F_\beta(p_\epsilon)} \int_{y_1}^{y_1/p_\epsilon} \frac{1}{y_2^2} - \frac{y_1}{y_2^3} dy_2 = \frac{12y_1}{1-F_\beta(p_\epsilon)} [-\frac{1}{y_2} + \frac{y_1}{2y_2^2}\Big|_{y_1}^{y_1/p_\epsilon}]$$
$$= \frac{12y_1}{1-F_\beta(p_\epsilon)} [-\frac{p_\epsilon}{y_1} + \frac{p_\epsilon^2}{2y_1} + \frac{1}{y_1} - \frac{1}{2y_1}] = \frac{6(p_\epsilon^2 - 2p_\epsilon + 1)}{1-F_\beta(p_\epsilon)}. \tag{E.32}$$

For $p_\epsilon \leq y_1 \leq 1/2$,

$$f_{y_1}(y_1) = \frac{12y_1}{1-F_\beta(p_\epsilon)} \int_{y_1}^{1/2} \frac{1}{y_2^2} - \frac{y_1}{y_2^3} dy_2 = \frac{12y_1}{1-F_\beta(p_\epsilon)} [-\frac{1}{y_2} + \frac{y_1}{2y_2^2}\Big|_{y_1}^{1/2}]$$
$$= \frac{12y_1}{1-F_\beta(p_\epsilon)} [-2 + 2y_1 + \frac{1}{y_1} - \frac{1}{2y_1}] = \frac{6(4y_1^2 - 4y_1 + 1)}{1-F_\beta(p_\epsilon)}. \tag{E.33}$$

Thus, as a piecewise function the PDF is formally,

$$f_{y_1}(y_1) = \begin{cases} \frac{6(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \frac{6(4y_1^2-4y_1+1)}{1-F_\beta(p_\epsilon)} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.34}$$

Now, the CDF is determined through integration,

$$F_{y_1}(y_1) = \int_0^{y_1} f_{y_1}(y_1) dy_1 = \begin{cases} \frac{6y_1(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \frac{2y_1(4y_1^2-6y_1+3)+p_\epsilon^2(2p_\epsilon-3)}{1-F_\beta(p_\epsilon)} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.35}$$

One can integrate the CDF to get,

$$\int_0^{y_1} F_{y_1}(y_1) = \begin{cases} \frac{3y_1^2(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} & \text{for } 0 \leq y_1 \leq \frac{p_\epsilon}{2}, \\ \frac{8y_1\left(2y_1^3-4y_1^2+3y_1+p_\epsilon^2(2p_\epsilon-3)\right)+p_\epsilon^3(4-3p_\epsilon)}{8(1-F_\beta(p_\epsilon))} & \text{for } \frac{p_\epsilon}{2} \leq y_1 \leq 1/2. \end{cases} \tag{E.36}$$

Plugging all of this back into Equation (7.7) yields,

$$\begin{aligned}\hat{b}_i^* &= \begin{cases} p_\epsilon + v_i^p \frac{6v_i^p(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} - \frac{3(v_i^p)^2(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)}, \\ p_\epsilon + v_i^p \frac{2v_i^p(4(v_i^p)^2-6v_i^p+3)+p_\epsilon^2(2p_\epsilon-3)}{1-F_\beta(p_\epsilon)} - \frac{8v_i^p\left(2(v_i^p)^3-4(v_i^p)^2+3v_i^p+p_\epsilon^2(2p_\epsilon-3)\right)+p_\epsilon^3(4-3p_\epsilon)}{8(1-F_\beta(p_\epsilon))}, \end{cases} \\ &= \begin{cases} p_\epsilon + \frac{3(v_i^p)^2(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2\left(6(v_i^p)^2-8v_i^p+3\right)+p_\epsilon^3(3p_\epsilon-4)}{8(1-F_\beta(p_\epsilon))} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}. \end{cases} \end{aligned} \tag{E.37}$$

Since b_i cannot be larger than 1, the bidding function is capped at 1 via,

$$b_i^* := \min\{\hat{b}_i^*, 1\}. \tag{E.38}$$

The utility gained by agent i for using such a bidding function is,

$$u(b_i^*) = \begin{cases} v_i^d - b_i^* + \frac{6(v_i^p)^2(p_\epsilon^2-2p_\epsilon+1)}{1-F_\beta(p_\epsilon)} & \text{for } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ v_i^d - b_i^* + \frac{v_i^p\left(8(v_i^p)^3-12(v_i^p)^2+6v_i^p+p_\epsilon^2(2p_\epsilon-3)\right)}{1-F_\beta(p_\epsilon)} & \text{for } \frac{p_\epsilon}{2} \le v_i^p \le 1/2. \end{cases} \tag{E.39}$$

When this utility is larger than 0, the agent will participate otherwise the agent will not submit a model to the regulator. Finally, one can find the optimal compliance level by using Assumption 9,

$$s_i^* := M^{-1}\left(b_i^*\right). \tag{E.40}$$

□

E.3 Repeating CIRCA Auctions

The current auction structure (Algorithm 11) expects agents to submit a single model trained solely for the upcoming auction. There is no expectation that the model will be reused for a future auction, or indication that the model has been submitted to a previous auction. Looking towards the future, it is desirable to design CIRCA to fit a repeatable auction structure, in which approved or rejected models may be resubmitted in subsequent auctions.

Repeated Agent Utility. Previously, in Algorithm 11, agents start the regulatory process with zero cost and value (*i.e.,* they are building their models from scratch). In repeating CIRCA auctions, agent cost and value are accumulated across all previous auction submissions. For example, if an agent trains its already-accepted model further to attain a higher compliance level s_i, its total accumulated training cost is $M(s_i)$. This agent's total value becomes the value its model gained from previous auction submissions plus any value gained from the current auction.

By allowing repeated CIRCA auctions, an agent is able to repeatedly submit its model for regulatory review. It should be noted that repeated submissions decrease the value of model deployment; once an agent earns the reward for deploying their model, subsequent deployments of the same model with improved compliance levels can be realistically expected to earn less value than the initial deployment. This loss in value for repeated submissions is characterized by an indicator function in the utility function that only allows deployment value to be obtained once, on initial acceptance of a model. While agents are allowed to win premium rewards across multiple auctions, a regulator can curb this by either limiting the number of auction submissions per agent or the number of auctions held per year. The repeated CIRCA auction utility of agent i,

who has participated in $a-1$ previous auctions, is defined as,

$$u_{i,a}(b_i) = \left(\sum_{n=1}^{a} \nu_i^n\right) - b_i, \tag{E.41}$$

where ν_i^n, the value gained at the n^{th} auction model i was submitted to, is formulated as:

$$\nu_i^n = \begin{cases} v_i^{d,n} \cdot 1_{(\text{if } \nu_i^{n-1}=0)} & \text{if } b_j^n \geq p_\epsilon^n \text{ and } b_i^n < b_j^n \text{ randomly sampled bid } b_j^n, \\ v_i^{d,n} \cdot 1_{(\text{if } \nu_i^{n-1}=0)} + v_i^{p,n} & \text{if } b_i^n \geq p_\epsilon^n \text{ and } b_i^n > b_j^n \text{ randomly sampled bid } b_j^n, \\ 0 & \text{if n} \leq 0. \end{cases} \tag{E.42}$$

The repeated CIRCA auction setup creates a unique property for models in training. If an agent intends to obtain a high compliance level, but an auction takes place mid-training, the agent is actually incentivized to submit their model early if they have a chance at winning the premium reward. Though the model may have a lower likelihood of earning the reward, there is no consequence for models failing to attain the premium reward. Gaining value is strictly beneficial to agents, and accumulated value helps offset the costs of training a model. This property only exists for the premium reward; the deployment reward can only be obtained once, thus there is no incentive to submit early to earn it.

Repeated Optimal Bidding Function. Using the same assumptions for single-auction CIRCA, namely Assumptions 8 and 9 along with private values, one can derive the bidding function for a rational agent under a repeated CIRCA auction setting. Following an equivalent setup to Theorem 14 with regards to the valuation of rewards, the cumulative distribution function for $v_i^p = V_i \lambda_i$ as $F_v(\cdot)$ and the probability distribution function as $f_v(\cdot)$ can be found.

From the definition of utility $u_{i,a}(b_i)$, an agent i that does not participate (*i.e.,* submitting $b_i = 0$) receives utility equal to ν_i^a. However, since $b_i = 0$ will never be larger than p_ϵ (by definition), it must be true that $\nu_i^a = 0$ as well, since the model will never meet the required compliance threshold. Therefore, a non-participating agent will always receive non-negative utility.

$$u_{i,a}(0) = 0. \tag{E.43}$$

Following closely to the proof of Theorem 14 in Appendix E.2, one finds that participating agents $i \in P$ (with P defined in the previous proof) will now have a utility of,

$$\begin{aligned} u_{i,a}(b_i) &= \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + v_i^p \mathbb{P}\big(b_i > b_j\big) - b(b_i), \quad b_j \sim \text{randomly sampled agent bid,} \\ &= \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + v_i^p F_v(b_i) - b(b_i). \end{aligned} \tag{E.44}$$

Taking the derivative and setting it equal to zero yields,

$$\frac{d}{db_i} u_{i,a}(b_i) = v_i^p f_v(b_i) - b'(b_i) = 0. \tag{E.45}$$

As agents bid in proportion to their valuation, the first-order conditions are solved at $b_i = v_i^p$,

$$b'(v_i^p) = v_i^p f_v(v_i^p). \tag{E.46}$$

Note, at this point in the proof the bidding function calculation is now equivalent to the calculations found in Theorem 14. One can thus follow the same steps to reveal the optimal bidding function,

$$b(v_i^p) := p_\epsilon + v_i^p F_v(v_i^p) - \int_0^{v_i^p} F_v(z)dz, \tag{E.47}$$

which is equivalent to the optimal bidding function derived in Theorem 14.

As the optimal bidding function is equivalent, calculations for the Nash Bidding Equilibrium are also equivalent to those found in Corollary 4 and Corollary 5. The optimal bid and utility participating in CIRCA (7.6) under the assumptions of Corollary 4 will thus be,

$$b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} & \text{if } 0 \leq v_i^p \leq \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2(\ln(2v_i^p) - 1/2) + p_\epsilon^2}{8(p_\epsilon - 1)} & \text{if } \frac{p_\epsilon}{2} \leq v_i^p \leq \frac{1}{2}, \end{cases}$$

$$u_{i,a}(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + \frac{2(v_i^p)^2 \ln(p_\epsilon)}{p_\epsilon - 1} - b_i^* & \text{if } 0 \leq v_i^p \leq \frac{p_\epsilon}{2}, \\ \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + \frac{2(v_i^p)^2(\ln(2p_\epsilon) - 1) + p_\epsilon}{p_\epsilon - 1} - b_i^* & \text{if } \frac{p_\epsilon}{2} \leq v_i^p \leq \frac{1}{2}. \end{cases}$$

Agents participating in CIRCA under Corollary 4 submit models with the following compliance,

$$s_i^* := \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no model submission)} & \text{else.} \end{cases}$$

The optimal bid and utility participating in CIRCA (7.6) under the assumptions of Corollary 5 will be,

$$b_i^* := \min\{\hat{b}_i^*, 1\}, \quad \hat{b}_i^* = \begin{cases} p_\epsilon + \frac{3(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} & \text{if } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ p_\epsilon + \frac{8(v_i^p)^2\left(6(v_i^p)^2 - 8v_i^p + 3\right) + p_\epsilon^3(3p_\epsilon - 4)}{8(1 - F_\beta(p_\epsilon))} & \text{if } \frac{p_\epsilon}{2} \le v_i^p \le \frac{1}{2}, \end{cases}$$

$$u_{i,a}(b_i^*; \boldsymbol{b}_{-i}) = \begin{cases} \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + \frac{6(v_i^p)^2(p_\epsilon^2 - 2p_\epsilon + 1)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } 0 \le v_i^p \le \frac{p_\epsilon}{2}, \\ \nu_i^a + v_i^d \cdot 1_{(\nu_i^a = 0)} + \frac{v_i^p\left(8(v_i^p)^3 - 12(v_i^p)^2 + 6v_i^p + p_\epsilon^2(2p_\epsilon - 3)\right)}{1 - F_\beta(p_\epsilon)} - b_i^* & \text{for } \frac{p_\epsilon}{2} \le v_i^p \le 1/2. \end{cases}$$

Agents participating in CIRCA under Corollay 5 submit models with the following compliance,

$$s_i^* = \begin{cases} M^{-1}(b_i^*) > \epsilon & \text{if } u_i(b_i^*; \boldsymbol{b}_{-i}) > 0, \\ 0 \text{ (no model submission)} & \text{else.} \end{cases}$$

E.4 AI Regulation Challenges

While Chapter 7 addresses key challenges in regulating AI compliance, several directions remain open for future exploration:

(1) Model Evaluation: Creating a realistic protocol for the regulator to evaluate submitted model compliance levels is important to ensure agents do not skirt around compliance requirements. While this problem is left for future work, one possible solution is that agents can either provide the regulator API access to test its model or provide the model weights directly to the regulator. Truthfulness can be enforced via audits and the threat of legal action.

(2) Extension to Heterogeneous Settings: Extending the proposed mechanism to heterogeneous scenarios, where evaluation data for agents and regulators differs, is a critical next step.

Real-world data distributions often vary across contexts, and understanding how these variations affect both model compliance and agent strategies will create a more robust regulatory mechanism. While explicit protocols or mathematical formulations are left as future work, a few ideas to tackle this issue are provided below. One idea could be establishing a data-sharing framework between agents and the regulator, where each participating agent must contribute part of (or all of) its data to the regulator for evaluation. If data can be anonymized, then this would be a suitable solution. Another idea could be that the regulator collects data on its own, and can compare its distribution of data versus each participating agents' data distribution. If distributions greatly differ, then the regulator could collect more data or resort to the previous data-sharing method.

Bibliography

[1] Petroc Taylor. Data growth worldwide 2010-2025, Nov 2023. URL www.statista.com/statistics/871513/worldwide-data-created/.

[2] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Communication-efficient local decentralized sgd methods. *arXiv preprint arXiv:1910.09126*, 2019.

[3] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[4] Sai Praneeth Karimireddy, Wenshuo Guo, and Michael I Jordan. Mechanisms that incentivize data sharing in federated learning. *arXiv preprint arXiv:2207.04557*, 2022.

[5] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[6] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[7] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. 2024. *URL https://openai. com/research/video-generation-models-as-world-simulators*, 3, 2024.

[8] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[9] Arjun Panesar. *Machine learning and AI for healthcare*. Springer, 2019.

[10] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[11] Fernando van der Vlist, Anne Helmond, and Fabian Ferrari. Big ai: Cloud infrastructure dependence and the industrialisation of artificial intelligence. *Big Data & Society*, 11(1): 20539517241232630, 2024.

[12] Bart Gerritsen and Imre Horváth. The upcoming and proliferation of ubiquitous technologies in products and processes. *Strojniški vestnik-Journal of Mechanical Engineering*, 56 (11):765–783, 2010.

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

[14] Raghubir Singh and Sukhpal Singh Gill. Edge ai: a survey. *Internet of Things and Cyber-Physical Systems*, 3:71–92, 2023.

[15] Panagiotis Papadimitratos and Zygmunt J Haas. Secure data transmission in mobile ad hoc networks. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 41–50, 2003.

[16] Huang Lu, Jie Li, and Mohsen Guizani. Secure and efficient data transmission for cluster-based wireless sensor networks. *IEEE transactions on parallel and distributed systems*, 25 (3):750–761, 2013.

[17] Nasir N Hurrah, Shabir A Parah, Javaid A Sheikh, Fadi Al-Turjman, and Khan Muhammad. Secure data transmission framework for confidentiality in iots. *Ad Hoc Networks*, 95:101989, 2019.

[18] Ryo Sugihara and Rajesh K Gupta. Improving the data delivery latency in sensor networks with controlled mobility. In *International Conference on Distributed Computing in Sensor Systems*, pages 386–399. Springer, 2008.

[19] Saurabh Shukla, Mohd Fadzil Hassan, Duc Chung Tran, Rehan Akbar, Irving Vitra Paputungan, and Muhammad Khalid Khan. Improving latency in internet-of-things and cloud computing for real-time data transmission: a systematic literature review (slr). *Cluster Computing*, pages 1–24, 2023.

[20] Ahmed Abd, Tarek Saadawi, Myung Lee, et al. Improving throughput and reliability in mobile wireless networks via transport layer bandwidth aggregation. *Computer Networks*, 46(5):635–649, 2004.

[21] Penglin Dai, Kai Liu, Xiao Wu, Yong Liao, Victor Chung Sing Lee, and Sang Hyuk Son. Bandwidth efficiency and service adaptiveness oriented data dissemination in heterogeneous vehicular networks. *IEEE Transactions on Vehicular Technology*, 67(7):6585–6598, 2018.

[22] Leana Golubchik, Samir Khuller, Koyel Mukherjee, and Yuan Yao. To send or not to send: Reducing the cost of data transmission. In *2013 Proceedings IEEE INFOCOM*, pages 2472–2478. IEEE, 2013.

[23] Zongjian He, Daqiang Zhang, and Junbin Liang. Cost-efficient sensory data transmission in heterogeneous software-defined vehicular networks. *IEEE Sensors Journal*, 16(20): 7342–7354, 2016.

[24] Puneet Himthani, Ghanshyam Prasad Dubey, Brij Mohan Sharma, and Ankur Taneja. Big data privacy and challenges for machine learning. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 707–713. IEEE, 2020.

[25] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[26] Directorate-General for Communication. Data act enters into force: What it means for you, Jan 2024. URL commission.europa.eu/news/data-act-enters-force-what-it-means-you-2024-01-11_en.

[27] Ben Wolford. What is gdpr, the eu's new data protection law?, Sep 2023. URL gdpr.eu/what-is-gdpr/.

[28] Herbert Woisetschläger, Alexander Erben, Bill Marino, Shiqiang Wang, Nicholas D Lane, Ruben Mayer, and Hans-Arno Jacobsen. Federated learning priorities under the european union artificial intelligence act. *arXiv preprint arXiv:2402.05968*, 2024.

[29] Herbert Woisetschläger, Simon Mertel, Christoph Krönke, Ruben Mayer, and Hans-Arno Jacobsen. Federated learning and ai regulation in the european union: Who is responsible?–an interdisciplinary analysis. *arXiv preprint arXiv:2407.08105*, 2024.

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

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

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

[33] Momina Shaheen, Muhammad Shoaib Farooq, Tariq Umer, and Byung-Seo Kim. Applications of federated learning; taxonomy, challenges, and research trends. *Electronics*, 11 (4):670, 2022.

[34] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2021.

[35] Marco Bornstein, Tahseen Rabbani, Evan Z Wang, Amrit Bedi, and Furong Huang. Swift: Rapid decentralized federated learning via wait-free model communication. In *The Eleventh International Conference on Learning Representations*, 2023.

[36] Avrim Blum, Nika Haghtalab, Richard Lanas Phillips, and Han Shao. One for one, or all for all: Equilibria and optimality of collaboration in federated learning. In *International Conference on Machine Learning*, pages 1005–1014. PMLR, 2021.

[37] Aniket Murhekar, Zhuowen Yuan, Bhaskar Ray Chaudhury, Bo Li, and Ruta Mehta. Incentives in federated learning: Equilibria, dynamics, and mechanisms for welfare maximization. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019.

[39] Marco Bornstein, Amrit Singh Bedi, Abdirisak Mohamed, and Furong Huang. Fact or fiction: Can truthful mechanisms eliminate federated free riding? *Advances in Neural Information Processing Systems*, 37, 2024.

[40] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[41] Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. Matcha: Speeding up decentralized sgd via matching decomposition sampling. In *2019 Sixth Indian Control Conference (ICC)*, pages 299–300. IEEE, 2019.

[42] Edwige Cyffers and Aurélien Bellet. Privacy amplification by decentralization. In *International Conference on Artificial Intelligence and Statistics*, pages 5334–5353. PMLR, 2022.

[43] Thorsten Wittkopp and Alexander Acker. Decentralized federated learning preserves model and data privacy. In *International Conference on Service-Oriented Computing*, pages 176–187. Springer, 2020.

[44] iphone, ipad, and ipod benchmarks. browser.geekbench.com/ios-benchmarks, 2024.

[45] Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

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

[47] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 401–416, 2020.

[48] Qi Liu, Bo Yang, Zhaojian Wang, Dafeng Zhu, Xinyi Wang, Kai Ma, and Xinping Guan. Asynchronous decentralized federated learning for collaborative fault diagnosis of pv stations. *IEEE Transactions on Network Science and Engineering*, 2022.

[49] Giorgi Nadiradze, Amirmojtaba Sabour, Peter Davies, Shigang Li, and Dan Alistarh. Asynchronous decentralized sgd with quantized and local updates. *Advances in Neural Information Processing Systems*, 34:6829–6842, 2021.

[50] Soheil Abbasloo and H Jonathan Chao. Sharpedge: An asynchronous and core-agnostic solution to guarantee bounded-delays. *CCF Transactions on Networking*, 3(1):35–50, 2020.

[51] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

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

[53] Tuncer Can Aysal, Mehmet Ercan Yildiz, Anand D Sarwate, and Anna Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal processing*, 57(7): 2748–2761, 2009.

[54] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1653–1664. IEEE, 2005.

[55] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. *Advances in Neural Information Processing Systems*, 28, 2015.

[56] Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1):267–305, 2016.

[57] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[58] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011.

[59] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. *Advances in neural information processing systems*, 24, 2011.

[60] Xuanyu Cao and Tamer Başar. Decentralized multi-agent stochastic optimization with pairwise constraints and quantized communications. *IEEE Transactions on Signal Processing*, 68:3296–3311, 2020. doi: 10.1109/TSP.2020.2997394.

[61] Amrit Singh Bedi, Alec Koppel, and Ketan Rajawat. Asynchronous online learning in multi-agent systems with proximity constraints. *IEEE Transactions on Signal and Information Processing over Networks*, 5(3):479–494, 2019.

[62] Amrit Singh Bedi, Alec Koppel, and Ketan Rajawat. Asynchronous saddle point algorithm for stochastic optimization in heterogeneous networks. *IEEE Transactions on Signal Processing*, 67(7):1742–1757, 2019.

[63] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101, 2018.

[64] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

[65] Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake Woodworth. Asynchronous sgd beats minibatch sgd under arbitrary delays. In *NeurIPS 2022-Thirty-sixth Conference on Neural Information Processing Systems*, 2022.

[66] Hao Ye, Le Liang, and Geoffrey Ye Li. Decentralized federated learning with unreliable communications. *IEEE Journal of Selected Topics in Signal Processing*, 2022.

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

[68] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

[69] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

[70] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[71] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[72] Tahseen Rabbani, Marco Bornstein, and Furong Huang. Large-scale distributed learning via private on-device lsh. In *Advances in Neural Information Processing Systems*, volume 36, pages 16153–16171, 2023.

[73] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[74] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[75] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[76] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, 27, 2014.

[77] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.

[78] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2020.

[79] Beidi Chen and Anshumali Shrivastava. Revisiting winner take all (wta) hashing for sparse datasets. *arXiv preprint arXiv:1612.01834*, 2016.

[80] William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[81] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[82] Anshumali Shrivastava and Ping Li. In defense of minhash over simhash. In *Artificial Intelligence and Statistics*, pages 886–894. PMLR, 2014.

[83] Zaifeng Pan, Feng Zhang, Hourun Li, Chenyang Zhang, Xiaoyong Du, and Dong Deng. G-slide: A gpu-based sub-linear deep learning engine via lsh sparsification. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):3015–3027, 2021.

[84] Beidi Chen and Anshumali Shrivastava. Densified winner take all (wta) hashing for sparse datasets. In *Uncertainty in artificial intelligence*, 2018.

[85] Andrew Ziegler, Eric Christiansen, David Kriegman, and Serge Belongie. Locally uniform comparison image descriptor. *Advances in Neural Information Processing Systems*, 25, 2012.

[86] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625, 1997.

[87] Noam Nisan. Pseudorandom generators for space-bounded computations. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 204–212, 1990.

[88] Lars Engebretsen, Piotr Indyk, and Ryan O'Donnell. Derandomized dimensionality reduction with applications. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, page 705–712, USA, 2002. Society for Industrial and Applied Mathematics. ISBN 089871513X.

[89] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[90] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[91] Jay Yagnik, Dennis Strelow, David A Ross, and Ruei-sung Lin. The power of comparative reasoning. In *2011 International Conference on Computer Vision*, pages 2431–2438. IEEE, 2011.

[92] Devi Parikh and Kristen Grauman. Relative attributes. In *2011 International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[93] Ryan Spring and Anshumali Shrivastava. Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 445–454, 2017.

[94] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[95] Junjie Jia and Zhipeng Lei. Personalized recommendation algorithm for mobile based on federated matrix factorization. In *Journal of Physics: Conference Series*, volume 1802, page 032021. IOP Publishing, 2021.

[96] Chengshuai Shi, Cong Shen, and Jing Yang. Federated multi-armed bandits with personalization. In *International Conference on Artificial Intelligence and Statistics*, pages 2917–2925. PMLR, 2021.

[97] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fedattack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4164–4172, 2022.

[98] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. Pipattack: Poisoning federated recommender systems formanipulating item promotion. *arXiv preprint arXiv:2110.10926*, 2021.

[99] Zareen Alamgir, Farwa K Khan, and Saira Karim. Federated recommenders: methods, challenges and future. *Cluster Computing*, 25(6):4075–4096, 2022.

[100] Minghao Yan, Nicholas Meisburger, Tharun Medini, and Anshumali Shrivastava. Distributed slide: Enabling training large neural networks on low bandwidth and simple cpu-clusters via model parallelism and sparsity. *arXiv preprint arXiv:2201.12667*, 2022.

[101] Zhaozhuo Xu, Luyang Liu, Zheng Xu, and Anshumali Shrivastava. Adaptive sparse federated learning in large output spaces via hashing. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[102] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. *Advances in neural information processing systems*, 28, 2015.

[103] Felix Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. In *International conference on machine learning*, pages 946–954. PMLR, 2014.

[104] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. Analyzing social bookmarking systems: A del. icio. us cookbook. In *Proceedings of the ECAI 2008 Mining Social Data Workshop*, pages 26–30, 2008.

[105] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

[106] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.

[107] Kush Bhatia, Kunal Dahiya, Himanshu Jain, Anshul Mittal, Yashoteja Prabhu, and Manik Varma. The extreme classification repository: Multi-label datasets and code. *URL http://manikvarma. org/downloads/XC/XMLRepository. html*, 2016.

[108] Ian En-Hsu Yen, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Sanjiv Kumar, and Pradeep Ravikumar. Loss decomposition for fast learning in large output spaces. In *International Conference on Machine Learning*, pages 5640–5649. PMLR, 2018.

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

[110] Marco Bornstein, Tahseen Rabbani, Brian Joseph Gravelle, and Furong Huang. Shrinking the size of deep extreme multi-label classification. In *Workshop on Machine Learning and Compression, NeurIPS 2024*, 2024.

[111] Capital One. Number of products on amazon. Technical report, Capital One Shopping Research, Jul 2024. URL capitaloneshopping.com/research/number-of-products-on-amazon/.

[112] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL http://manikvarma.org/downloads/XC/XMLRepository.html.

[113] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28, 2015.

[114] Tianyi Zhou, Dacheng Tao, and Xindong Wu. Compressed labeling on distilled labelsets for multi-label learning. *Machine Learning*, 88:69–126, 2012.

[115] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

[116] Arpan Dasgupta, Siddhant Katyan, Shrutimoy Das, and Pawan Kumar. Review of extreme multilabel classification. *arXiv preprint arXiv:2302.05971*, 2023.

[117] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[118] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, 75:2008, 2008.

[119] Eneldo Loza Mencía and Johannes Fürnkranz. *Efficient multilabel classification algorithms for large-scale problems in the legal domain*. Springer, 2010.

[120] Arkaitz Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*, 2012.

[121] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

[122] Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. Balanced sparsity for efficient dnn inference on gpu. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5676–5683, 2019.

[123] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.

[124] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[125] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433): 1287–1289, 2019.

[126] David Solans, Battista Biggio, and Carlos Castillo. Poisoning attacks on algorithmic fairness. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 162–177. Springer, 2020.

[127] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, Lingjuan Lyu, and Ji Liu. Data poisoning attacks on federated machine learning. *IEEE Internet of Things Journal*, 9(13):11365–11375, 2021.

[128] Yuancheng Xu, Jiarui Yao, Manli Shu, Yanchao Sun, Zichu Wu, Ning Yu, Tom Goldstein, and Furong Huang. Shadowcast: Stealthy data poisoning attacks against vision-language models. *arXiv preprint arXiv:2402.06659*, 2024.

[129] Yann Fraboni, Richard Vidal, and Marco Lorenzi. Free-rider attacks on model aggregation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1846–1854. PMLR, 2021.

[130] Jierui Lin, Min Du, and Jian Liu. Free-riders in federated learning: Attacks and defenses. *arXiv preprint arXiv:1911.12560*, 2019.

[131] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020.

[132] Hamid Mozaffari, Sunav Choudhary, and Amir Houmansadr. Fake or compromised? making sense of malicious clients in federated learning. *arXiv preprint arXiv:2403.06319*, 2024.

[133] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[134] Jinhyun So, Başak Güler, and A Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2020.

[135] Mingshu Cong, Han Yu, Xi Weng, and Siu Ming Yiu. A game-theoretic framework for incentive mechanism design in federated learning. *Federated Learning: Privacy and Incentive*, pages 205–222, 2020.

[136] Wei Yang Bryan Lim, Zehui Xiong, Chunyan Miao, Dusit Niyato, Qiang Yang, Cyril Leung, and H Vincent Poor. Hierarchical incentive mechanism design for federated machine learning in mobile networks. *IEEE Internet of Things Journal*, 7(10):9575–9588, 2020.

[137] Yuan Liu, Mengmeng Tian, Yuxin Chen, Zehui Xiong, Cyril Leung, and Chunyan Miao. A contract theory based incentive mechanism for federated learning. In *Federated and Transfer Learning*, pages 117–137. Springer, 2022.

[138] Lingjuan Lyu, Xinyi Xu, Qian Wang, and Han Yu. Collaborative fairness in federated learning. *Federated Learning: Privacy and Incentive*, pages 189–204, 2020.

[139] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *International conference on machine learning*, pages 8927–8936. PMLR, 2020.

[140] Xinyi Xu and Lingjuan Lyu. A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning. *arXiv preprint arXiv:2011.10464*, 2020.

[141] Marco Bornstein, Amrit Bedi, Anit Kumar Sahu, Furqan Khan, and Furong Huang. Realfm: A realistic mechanism to incentivize data contribution and device participation. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023.

[142] Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 1387–1395. IEEE, 2019.

[143] Jianfeng Lu, Bangqi Pan, Abegaz Mohammed Seid, Bing Li, Gangqiang Hu, and Shaohua Wan. Truthful incentive mechanism design via internalizing externalities and lp relaxation for vertical federated learning. *IEEE Transactions on Computational Social Systems*, 2022.

[144] Chenrui Wu, Yifei Zhu, Rongyu Zhang, Yun Chen, Fangxin Wang, and Shuguang Cui. Fedab: Truthful federated learning with auction-based combinatorial multi-armed bandit. *IEEE Internet of Things Journal*, 2023.

[145] Xuezhen Tu, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Yang Zhang, and Juan Li. Incentive mechanisms for federated learning: From economic and game theoretic perspective. *IEEE transactions on cognitive communications and networking*, 8(3):1566–1593, 2022.

[146] Yufeng Zhan, Peng Li, Song Guo, and Zhihao Qu. Incentive mechanism design for federated learning: Challenges and opportunities. *IEEE network*, 35(4):310–317, 2021.

[147] Yufeng Zhan, Jie Zhang, Zicong Hong, Leijie Wu, Peng Li, and Song Guo. A survey of incentive mechanism design for federated learning. *IEEE Transactions on Emerging Topics in Computing*, 10(2):1035–1044, 2021.

[148] Jiaqi Liu, Wei Wang, Deng Li, Shaohua Wan, and Hui Liu. Role of gifts in decision making: An endowment effect incentive mechanism for offloading in the iov. *IEEE Internet of Things Journal*, 6(4):6933–6951, 2019.

[149] Yufeng Zhan, Yuanqing Xia, and Jinhui Zhang. Incentive mechanism in platform-centric mobile crowdsensing: A one-to-many bargaining approach. *Computer Networks*, 132: 40–52, 2018.

[150] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal*, 7(7):6360–6368, 2020.

[151] Guoju Gao, Mingjun Xiao, Jie Wu, Liusheng Huang, and Chang Hu. Truthful incentive mechanism for nondeterministic crowdsensing with vehicles. *IEEE Transactions on Mobile Computing*, 17(12):2982–2997, 2018.

[152] Xiumin Wang, Wayes Tushar, Chau Yuen, and Xinglin Zhang. Promoting users' participation in mobile crowdsourcing: A distributed truthful incentive mechanism (dtim) approach. *IEEE Transactions on Vehicular Technology*, 69(5):5570–5582, 2020.

[153] Jia Xu, Jinxin Xiang, and Dejun Yang. Incentive mechanisms for time window dependent tasks in mobile crowdsensing. *IEEE Transactions on Wireless Communications*, 14(11): 6353–6364, 2015.

[154] Honggang Zhang, Benyuan Liu, Hengky Susanto, Guoliang Xue, and Tong Sun. Incentive mechanism for proximity-based mobile crowd service systems. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[155] Tra Huong Thi Le, Nguyen H Tran, Yan Kyaw Tun, Zhu Han, and Choong Seon Hong. Auction based incentive design for efficient federated learning in cellular wireless networks. In *2020 IEEE wireless communications and networking conference (WCNC)*, pages 1–6. IEEE, 2020.

[156] Ruiting Zhou, Jinlong Pang, Zhibo Wang, John CS Lui, and Zongpeng Li. A truthful procurement auction for incentivizing heterogeneous clients in federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 183–193. IEEE, 2021.

[157] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[158] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022.

[159] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

[160] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[161] Bo Wang, Hongtao Li, Ximeng Liu, and Yina Guo. Frad: Free-rider attacks detection mechanism for federated learning in aiot. *IEEE Internet of Things Journal*, 2023.

[162] Jianhua Wang, Xiaolin Chang, Ricardo J Rodrìguez, and Yixiang Wang. Assessing anonymous and selfish free-rider attacks in federated learning. In *2022 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2022.

[163] Jianhua Wang. Pass: Parameters audit-based secure and fair federated learning scheme against free rider. *arXiv preprint arXiv:2207.07292*, 2022.

[164] Yufeng Zhan, Peng Li, Kun Wang, Song Guo, and Yuanqing Xia. Big data analytics by crowdlearning: Architecture and mechanism design. *IEEE Network*, 34(3):143–147, 2020.

[165] Enric Junqué de Fortuny, David Martens, and Foster Provost. Predictive modeling with big data: is bigger really better? *Big data*, 1(4):215–226, 2013.

[166] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*, 2020.

[167] Xiangxin Zhu, Carl Vondrick, Deva Ramanan, and Charless C Fowlkes. Do we need more training data or better models for object detection?. In *BMVC*, volume 3. Citeseer, 2012.

[168] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[169] Mengjing Chen, Yang Liu, Weiran Shen, Yiheng Shen, Pingzhong Tang, and Qiang Yang. Mechanism design for multi-party machine learning. *arXiv preprint arXiv:2001.08996*, 2020.

[170] Rongfei Zeng, Chao Zeng, Xingwei Wang, Bo Li, and Xiaowen Chu. A comprehensive survey of incentive mechanism for federated learning. *arXiv preprint arXiv:2106.15406*, 2021.

[171] Asad Ali, Inaam Ilahi, Adnan Qayyum, Ihab Mohammed, Ala Al-Fuqaha, and Junaid Qadir. A systematic review of federated learning incentive mechanisms and associated security challenges. *Computer Science Review*, 50:100593, 2023.

[172] Xinyi Xu, Lingjuan Lyu, Xingjun Ma, Chenglin Miao, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Gradient driven rewards to guarantee fairness in collaborative machine learning. *Advances in Neural Information Processing Systems*, 34:16104–16117, 2021.

[173] Wei Yang Bryan Lim, Jianqiang Huang, Zehui Xiong, Jiawen Kang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, and Chunyan Miao. Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach. *IEEE Transactions on Intelligent Transportation Systems*, 22(8):5140–5154, 2021.

[174] Yuntao Wang, Zhou Su, Tom H Luan, Ruidong Li, and Kuan Zhang. Federated learning with fair incentives and robust aggregation for uav-aided crowdsensing. *IEEE Transactions on Network Science and Engineering*, 9(5):3179–3196, 2021.

[175] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. Towards fair and privacy-preserving federated deep models. *IEEE Transactions on Parallel and Distributed Systems*, 31(11):2524–2541, 2020.

[176] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. *Federated Learning: Privacy and Incentive*, pages 153–167, 2020.

[177] Marco Bornstein, Zora Che, Suhas Julapalli, Abdirisak Mohamed, Amrit Singh Bedi, and Furong Huang. Auction-based regulation for artificial intelligence. *arXiv preprint arXiv:2410.01871*, 2024.

[178] Jack Brewster. How i built an ai-powered, self-running propaganda machine for $105. *The Wall Street Journal*, 2024. URL www.wsj.com/politics/how-i-built-an-ai-powered-self-running-propaganda-machine-for-105-e9888705.

[179] Adi Robertson. Google apologizes for 'missing the mark' after gemini generated racially diverse nazis. *The Verge*, 2024. URL www.theverge.com/2024/2/21/24079371/google-ai-gemini-generative-inaccurate-historical.

[180] Jeremy White. See how easily a.i. chatbots can be taught to spew disinformation. *The New York Times*, 2024. URL www.nytimes.com/interactive/2024/05/19/technology/biased-ai-chatbots.html.

[181] Denny Jacob. Openai forms new committee to evaluate safety, security. *The Wall Street Journal*, 2024. URL www.wsj.com/tech/ai/openai-forms-new-committee-to-evaluate-safety-security-4a6e74bb.

[182] Deepa Seetharaman. Openai, meta and google sign on to new child exploitation safety measures. *The Wall Street Journal*, 2024. URL www.wsj.com/tech/ai/ai-developers-agree-to-new-safety-measures-to-fight-child-exploitation-2a58129c.

[183] Jeremy White. How strangers got my email address from chatgpt's model. *The New York Times*, 2023. URL www.nytimes.com/interactive/2023/12/22/technology/openai-chatgpt-privacy-exploit.html.

[184] Alexandra Bruell. New york times sues microsoft and openai, alleging copyright infringement. *The Wall Street Journal*, 2023. URL www.wsj.com/tech/ai/new-york-times-sues-microsoft-and-openai-alleging-copyright-infringement-fd85e1c4.

[185] Cade Metz, Cecilia Kang, Sheera Frenkel, Stuart A. Thompson, and Nico Grant. How tech giants cut corners to harvest data for a.i. *The New York Times*, 2024. URL www.nytimes.com/2024/04/06/technology/tech-giants-harvest-data-artificial-intelligence.html.

[186] J. Edward Moreno. Boom in a.i. prompts a test of copyright law. *The New York Times*, 2023. URL www.nytimes.com/2023/12/30/business/media/copyright-law-ai-media.html.

[187] Dipto Barman, Ziyi Guo, and Owen Conlan. The dark side of language models: Exploring the potential of llms in multimedia disinformation generation and dissemination. *Machine Learning with Applications*, 16:100545, 2024. ISSN 2666-8270. doi: doi.org/10.1016/j.mlwa.2024.100545. URL www.sciencedirect.com/science/article/pii/S2666827024000215.

[188] Terrence Neumann, Sooyong Lee, Maria De-Arteaga, Sina Fazelpour, and Matthew Lease. Diverse, but divisive: Llms can exaggerate gender differences in opinion related to harms of misinformation. *arXiv preprint arXiv:2401.16558*, 2024.

[189] Yanshen Sun, Jianfeng He, Limeng Cui, Shuo Lei, and Chang-Tien Lu. Exploring the deceptive power of llm-generated fake news: A study of real-world detection challenges. *arXiv preprint arXiv:2403.18249*, 2024.

[190] The White House. Fact sheet: President biden issues executive order on safe, secure, and trustworthy artificial intelligence, 2023.

[191] California Legislative Information. Sb-1047 safe and secure innovation for frontier artificial intelligence models act. leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202320240SB1047, 2024.

[192] U.S. Food and Drug Administration. E19 a selective approach to safety data collection in specific late-stage pre-approval or post-approval clinical trials, 2022. URL https://www.fda.gov/regulatory-information/search-fda-guidance-documents/e19-selective-approach-safety-data-collection-specific-late-stage-pre-approval-or-post-approval.

[193] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

[194] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

[195] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*, 2023.

[196] Gary D Lopez Munoz, Amanda J Minnich, Roman Lutz, Richard Lundeen, Raja Sekhar Rao Dheekonda, Nina Chikanov, Bolor-Erdene Jagdagdorj, Martin Pouliot, Shiven Chawla, Whitney Maxwell, et al. Pyrit: A framework for security risk identification and red teaming in generative ai system. *arXiv preprint arXiv:2410.02828*, 2024.

[197] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[198] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[199] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 2023.

[200] Robert N Stavins. A meaningful us cap-and-trade system to address climate change. *Harv. Envtl. L. Rev.*, 32:293, 2008.

[201] Lawrence H Goulder and Andrew R Schein. Carbon taxes versus cap and trade: a critical review. *Climate Change Economics*, 4(03):1350010, 2013.

[202] Zheyuan Liu, Guangyao Dou, Zhaoxuan Tan, Yijun Tian, and Meng Jiang. Towards safer large language models through machine unlearning. *arXiv preprint arXiv:2402.10058*, 2024.

[203] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*, 2024.

[204] Nikolaus Howe, Ian McKenzie, Oskar Hollinsworth, Michał Zajac, Tom Tseng, Aaron Tucker, Pierre-Luc Bacon, and Adam Gleave. Effects of scale on language model robustness. *arXiv preprint arXiv:2407.18213*, 2024.

[205] Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, Johannes Heidecke, and Amelia Glaese. Trading inference-time compute for adversarial robustness. https://cdn.openai.com/papers/trading-inference-time-compute-for-adversarial-robustness-20250121_1.pdf, 2025.

[206] Erwin Amann and Wolfgang Leininger. Asymmetric all-pay auctions with incomplete information: the two-player case. *Games and economic behavior*, 14(1):1–18, 1996.

[207] V. Bhaskar. Lecture 8: All pay auction, January 2018.

[208] Eva Tardos. Lecture 17: All-pay auctions, March 2017. URL www.cs.cornell.edu/courses/cs6840/2017sp/lecnotes/lec17.pdf.

[209] Patricia Gomes Rêgo de Almeida, Carlos Denner dos Santos, and Josivania Silva Farias. Artificial intelligence regulation: a framework for governance. *Ethics and Information Technology*, 23(3):505–525, 2021.

[210] Meena Jagadeesan, Michael I Jordan, and Jacob Steinhardt. Safety vs. performance: How multi-objective learning reduces barriers to market entry. *arXiv preprint arXiv:2409.03734*, 2024.

[211] Manuel J García Rodríguez, Vicente Rodríguez-Montequín, Pablo Ballesteros-Pérez, Peter ED Love, and Regis Signor. Collusion detection in public procurement auctions with machine learning algorithms. *Automation in Construction*, 133:104047, 2022.

[212] Mohammad Yaghini, Patty Liu, Franziska Boenisch, and Nicolas Papernot. Regulation games for trustworthy machine learning. *arXiv preprint arXiv:2402.03540*, 2024.

[213] Michael R Baye, Dan Kovenock, and Casper G De Vries. The all-pay auction with complete information. *Economic Theory*, 8:291–305, 1996.

[214] Dominic DiPalantino and Milan Vojnovic. Crowdsourcing and all-pay auctions. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 119–128, 2009.

[215] Ian Gemp, Thomas Anthony, Janos Kramar, Tom Eccles, Andrea Tacchetti, and Yoram Bachrach. Designing all-pay auctions using deep learning and multi-agent simulation. *Scientific Reports*, 12(1):16937, 2022.

[216] Jacob K Goeree and John L Turner. All-pay-all auctions. *University of Virginia, Mimeo*, 2000.

[217] Ron Siegel. All-pay contests. *Econometrica*, 77(1):71–92, 2009.

[218] Kimmo Karkkainen and Jungseock Joo. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1548–1558, 2021.

[219] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Multi-label learning with stronger consistency guarantees. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[220] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[221] Beomyeol Jeon, SM Ferdous, Muntasir Raihan Rahman, and Anwar Walid. Privacy-preserving decentralized aggregation for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[222] Aurélien Bellet, Anne-Marie Kermarrec, and Erick Lavoie. D-cliques: Compensating for data heterogeneity with topology in decentralized federated learning. In *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, pages 1–11. IEEE, 2022.

[223] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course, 1998.

[224] Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2014.

[225] Arjun K Gupta and Saralees Nadarajah. *Handbook of beta distribution and its applications*. CRC press, 2004.

[226] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[227] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[228] Li Hu, Hongyang Yan, Lang Li, Zijie Pan, Xiaozhang Liu, and Zulong Zhang. Mhat: An efficient model-heterogenous aggregation training scheme for federated learning. *Information Sciences*, 560:493–503, 2021.

[229] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[230] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of machine learning, 2018.

[231] Daron Acemoglu and Asu Ozdaglar. Lecture notes for course "6.207/14.15 networks", October 2009.