Re-examining Contribution Fairness in Federated Learning

by

Guilherme Campos

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2026

© Guilherme Campos 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Federated Learning (FL) allows multiple data owners to train a shared model collaboratively without exposing their local datasets. Because participation imposes real computation, communication, and data-collection costs, sustaining long-term collaboration requires reward mechanisms that satisfy contribution fairness, the principle that each client should be rewarded commensurately with its contribution to the training.

A substantial body of research pursues this goal through the distribution of contribution-based Top-K sparsified gradient rewards, such that higher-contributing clients receive denser, and, therefore, more useful, gradients than lower-contributing ones. Top-K sparsification, however, was originally devised as a gradient-compression technique intended to preserve convergence, which raises the question of whether it can actually produce the client-level model differentiation that contribution fairness demands.

Furthermore, existing frameworks often combine distinct contribution estimation algorithms, reward generation rules, and client model update methods, making it unclear which component is responsible for the observed fairness behavior.

This thesis investigates these questions by implementing two representative state-of-the-art frameworks, namely ACGSV and CFFL, and evaluating them across four benchmark datasets under distinct training-data partition settings. Building on this comparative evaluation, a series of controlled ablation studies is conducted to isolate the effects of three key components: the gradient used for reward generation, Top-K sparsification, and the retention of each client's locally accumulated gradient.

The analysis yields three main findings. First, contribution fairness is commonly evaluated using Pearson's r between clients' standalone and federated test accuracies, with standalone accuracy serving as the reference contribution value and forming a set of contribution ranks. However, Pearson's r can remain high even when most clients converge to nearly identical federated accuracies and the reference contribution ranking is not preserved. Thus, Pearson's r alone may provide an incomplete fairness assessment. To address this limitation, this thesis proposes a novel protocol that uses Kendall's τ to measure the reference contribution ranks preservation and the Gini Mean Difference to quantify differentiation among the final client models.

Second, Top-K sparsification induces significant model differentiation primarily when reward sparsity is high, corresponding to at least 70% in the studied settings. At lower sparsity levels, clients tend to converge to models with nearly identical performance. These findings indicate that sparsifying the aggregated gradient through Top-K is not, by itself, an effective and calibrated mechanism for estimated-contribution-based reward allocation.

Third, local gradient retention, whereby clients retain their locally accumulated gradients generated during model training alongside the received estimated-contribution-based reward, is commonly treated as a secondary detail in framework design because it is not governed by the FL framework's contribution estimation algorithm and the contribution-score-to-reward mapping pipeline. However, it is an under-acknowledged confounding factor that drives much of the model differentiation, preservation of the reference contribution ranking, and, consequently, the high fairness scores reported by these frameworks.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Wojciech Golab and Tahsin Reza, for the opportunity to be part of their research groups and for allowing me to grow as both a person and a researcher. I could not have undertaken this process without your guidance and support.

I also want to thank the friends I made along the way at the University of Waterloo, especially Steven Ng and Faraz Awan, for bringing so much joy to the daily routine, for all the thoughtful conversations, and for showing me that every failed experiment, stubborn bug, and looming deadline is far more survivable in good company. Thank you for turning long days into ones I actually looked forward to.

To my family and friends, who have shown me unconditional love and support: I am extremely grateful. To my parents, Carlese and Adelaine, thank you for all the sacrifices you made while raising me, for preparing me for the future, and for making sure I had opportunities in life that you never had. Thank you, Dad, for being a source of calm, for always reminding me that things are going to be fine, and for being there in the moments that mattered most. Thank you, Mom, for the sleepless nights you spent helping me study, for your prayers, your care, and your words of affirmation.

To my friends back home Rodrigo, Jose Ibrahim, Tak-Ming , Caio, Breno, Flavio, and Matheus thank you for all these years of friendship, for every call to check how I was doing, for all the laughs, and always supporting me to reach here. You are living proof that brothers don't need to share the same blood.

Finally, to my partner, Bruna, thank you for your love, for always laughing at my jokes, for all early mornings and late nights that we spent working together, for never letting me give up, for enduring through the hard moments, for listening to me talk about federated learning and sparsification countless times, and for making me cook you barbecue to you when it's -20°outside.

Thank you all.

Dedication

Dedico esta tese aos meus pais, que sempre me disseram que tudo o que a gente quer, a gente consegue; a gente só precisa tentar.

I dedicate this thesis to my parents, who always told me that anything is achievable, as long as we try.

To my partner, for all your unconditional love and support.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements v

Dedication vi

List of Figures xii

List of Tables xv

List of Abbreviations xix

1 Introduction 1

1.1 Motivation 3

1.2 Research Contributions 4

1.3 Thesis Outline 5

2 Background 7

2.1 Machine Learning 7

2.2 Federated Learning 10

2.3 Sustaining Client Participation Through Incentivization 12

2.4 Incentive-Aware FL . . . 13
2.5 Fairness-Aware FL . . . 14
2.6 Top-K Sparsification . . . 16
2.7 Federated Learning Contribution Estimation . . . 18
2.7.1 Contribution Estimation Scheme: Individual . . . 20
2.7.2 Contribution Estimation Scheme: Leave-One-Out . . . 20
2.7.3 Contribution Estimation Scheme: Shapley Value . . . 21
2.7.4 Data Utility Metric: Data Quantity . . . 22
2.7.5 Data Utility Metric: Data Diversity . . . 23
2.7.6 Data Utility Metric: Accuracy . . . 24
2.7.7 Data Utility Metric: Mutual Evaluation . . . 24
2.7.8 Data Utility Metric: Cosine Similarity . . . 25
2.8 Rank Correlation Coefficients . . . 26
2.8.1 Spearman's Rank Correlation Coefficient . . . 27
2.8.2 Kendall's Rank Correlation Coefficient . . . 28
2.9 Gini Mean Difference . . . 30

3 Related Work **31**

3.1 Frameworks Summary . . . 31
3.2 Monetary Rewards . . . 32
3.2.1 Profit Allocation for Federated Learning . . . 33
3.2.2 OpenFL: A scalable and secure decentralized FL system on the Ethereum blockchain . . . 34
3.2.3 A fairness-aware Incentive Scheme for Federated Learning . . . 34
3.2.4 CoAst: Validation-Free Contribution Assessment for Federated Learning based on Cross-Round Valuation . . . 35
3.3 Non-Monetary Rewards . . . 35
3.3.1 Collaborative Fairness in Federated Learning . . . 36

3.3.2 Towards Fair and Privacy-Preserving Federated Deep Models 36
3.3.3 Gradient-driven Rewards to Guarantee Fairness in Collaborative Machine Learning 38
3.3.4 Towards Fair Graph Federated Learning via Incentive Mechanisms . 38
3.3.5 FedDAR: Multidimensional Contribution Assessment and Adaptive Gradient Rewards 39
3.3.6 FedFAIM: A Model Performance-Based Fair Incentive Mechanism for Federated Learning 40
3.3.7 Privacy-Preserving and Hierarchically Fair FL in IoT Based on Compressed Sensing 41
3.3.8 Incentive-Aware Federated Learning with Training-Time Model Rewards 42
3.3.9 ShapleyFL: Robust Federated Learning Based on Shapley Value . . 43
3.4 Framework Selection Rationale 44
3.5 ACGSV Framework 45
3.5.1 Cosine Gradient Shapley Value Approximation 46
3.5.2 ACGSV Framework Workflow 48
3.6 CFFL framework 51
3.6.1 CFFL Framework Workflow 53
3.6.2 CFFL - Gradient Aggregation modification for class-label-skewed distribution 58
3.7 Chapter Conclusion 58

4 Experimental Analysis of contribution fairness frameworks 60
4.1 Methodology 60
4.1.1 Evaluation Metrics 61
4.1.2 Client-Level Shapley Values 63
4.1.3 Benchmark Datasets Description 63
4.1.4 Neural Architectures 69

4.1.5 Training Hyperparameters . . . 70
4.1.6 Experiment Assumptions . . . 72
4.1.7 FL Architecture Implementation . . . 73
4.2 Experiments results: CFFL framework . . . 74
4.3 Experiment Results: ACGSV Framework . . . 79
4.4 Experiments Summary . . . 85
4.5 New Contribution Fairness Evaluation Protocol: Kendall's τ and GMD . . . 87
4.6 Fairness Analysis with Shapley Values . . . 93
4.7 Chapter Conclusion . . . 99

5 Ablation Studies **101**
5.1 Design Differences between CFFL and ACGSV . . . 102
5.1.1 Framework Components that Influence Client Model State . . . 104
5.2 Ablation Studies: Methodology . . . 108
5.3 Ablation Studies: Evaluation Metrics . . . 109
5.4 Ablation Studies: Framework Variants . . . 110
5.4.1 UR-ACGSV . . . 111
5.4.2 OTOPK-CFFL . . . 112
5.4.3 DPR-CFFL . . . 113
5.4.4 DFR-ACGSV . . . 113
5.5 Ablation study results: UR-ACGSV Variant . . . 114
5.6 Ablation study results: OTOPK-CFFL Variant . . . 131
5.7 Ablation study results: DPR-CFFL Variant . . . 145
5.8 Ablation study results: DFR-ACGSV Variant . . . 158
5.9 Chapter Conclusion . . . 171

6 Conclusion and Future Directions **173**
6.1 Summary of Contributions and Findings 174
6.2 Implications 176
6.3 Limitations 177
6.4 Future Directions 177

References **179**

APPENDICES **193**

A Notation Tables **194**

B Shapley Value Properties **195**

C ACGSV Longer Training **197**

D Run-by-Run Results **202**
D.1 CFFL 202
D.2 ACGSV 204
D.3 Framework Variants Plots and Tables 206
D.3.1 Framework Variant: UR-ACGSV 206
D.3.2 Framework Variant: OTOPK-CFFL 213
D.3.3 Framework Variant: DPR-CFFL 220
D.3.4 Framework Variant: DFR-ACGSV 227

E CFFL original CLA Aggregation Experiments **234**

List of Figures

2.1 Simplified Neural Architecture 8

4.1 CFFL Convergence and Scatter Plots: MNIST 76
4.2 CFFL Convergence and Scatter Plots: FA-MNIST 77
4.3 CFFL Convergence and Scatter Plots: CIFAR10 78
4.4 CFFL Convergence and Scatter Plots: SST2 79
4.5 ACGSV Convergence and Scatter Plots: MNIST 82
4.6 ACGSV Convergence and Scatter Plots: FA-MNIST 83
4.7 ACGSV Convergence and Scatter Plots: CIFAR10 84
4.8 ACGSV Convergence and Scatter Plots: SST2 85
4.9 MNIST Shapley-value fairness analysis: ACGSV vs. CFFL 96
4.10 FA-MNIST Shapley-value fairness analysis: ACGSV vs. CFFL 97
4.11 CIFAR10 Shapley-value fairness analysis: ACGSV vs. CFFL 98
4.12 SST2 Shapley-value fairness analysis: ACGSV vs. CFFL 99

5.1 UR-ACGSV Convergence, Scatter, and Sparsity Plots: MNIST POW . . . 119
5.2 UR-ACGSV Convergence, Scatter, and Sparsity Plots: MNIST CLA 120
5.3 UR-ACGSV Convergence, Scatter, and Sparsity Plots: FA-MNIST POW . 121
5.4 UR-ACGSV Convergence, Scatter, and Sparsity Plots: FA-MNIST CLA . . 122
5.5 UR-ACGSV Convergence, Scatter, and Sparsity Plots: CIFAR10 POW . . 123
5.6 UR-ACGSV Convergence, Scatter, and Sparsity Plots: CIFAR10 CLA . . . 124

5.7 UR-ACGSV Convergence, Scatter, and Sparsity Plots: SST2 POW 125
5.8 MNIST and FA-MNIST Shapley-value fairness analysis: UR-ACGSV . . . 129
5.9 CIFAR10 and SST2 Shapley-value fairness analysis: UR-ACGSV 130
5.10 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: MNIST POW . 134
5.11 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: MNIST CLA . . 135
5.12 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: FA-MNIST POW 136
5.13 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: FA-MNIST CLA 137
5.14 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: CIFAR10 POW 138
5.15 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: CIFAR10 CLA . 139
5.16 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: SST2 POW . . . 140
5.17 MNIST and FA-MNIST Shapley-value fairness analysis: OTOPK-CFFL . . 143
5.18 CIFAR10 and SST2 Shapley-value fairness analysis: OTOPK-CFFL 144
5.19 DPR-CFFL Convergence and Scatter Plots: MNIST POW 148
5.20 DPR-CFFL Convergence and Scatter Plots: MNIST CLA 149
5.21 DPR-CFFL Convergence and Scatter Plots: FA-MNIST POW 150
5.22 DPR-CFFL Convergence and Scatter Plots: FA-MNIST POW 151
5.23 DPR-CFFL Convergence and Scatter Plots: CIFAR10 POW 152
5.24 DPR-CFFL Convergence and Scatter Plots: CIFAR10 CLA 153
5.25 DPR-CFFL Convergence and Scatter Plots: SST2 POW 154
5.26 MNIST and FA-MNIST Shapley-value fairness analysis: DPR-CFFL 156
5.27 CIFAR10 and SST2 Shapley-value fairness analysis: DPR-CFFL 157
5.28 DFR-ACGSV Convergence and Scatter Plots: MNIST POW 161
5.29 DFR-ACGSV Convergence and Scatter Plots: MNIST CLA 162
5.30 DFR-ACGSV Convergence and Scatter Plots: FA-MNIST POW 163
5.31 DFR-ACGSV Convergence and Scatter Plots: FA-MNIST POW 164
5.32 DFR-ACGSV Convergence and Scatter Plots: CIFAR10 POW 165
5.33 DFR-ACGSV Convergence and Scatter Plots: CIFAR10 CLA 166

5.34 DFR-ACGSV Convergence and Scatter Plots: SST2 POW 167
5.35 MNIST and FA-MNIST Shapley-value fairness analysis: DFR-ACGSV . . 169
5.36 CIFAR10 and SST2 Shapley-value fairness analysis: DFR-ACGSV 170

C.1 ACGSV longer training Convergence and Scatter Plots: MNIST 198
C.2 ACGSV longer training Convergence and Scatter Plots: FA-MNIST 199
C.3 ACGSV longer training Convergence and Scatter Plots: CIFAR10 200
C.4 ACGSV longer training Convergence and Scatter Plots: SST2 201

D.1 UR-ACGSV Convergence, Scatter, and Sparsity Plots: MNIST IID 209
D.2 UR-ACGSV Convergence, Scatter, and Sparsity Plots: FA-MNIST IID . . 210
D.3 UR-ACGSV Convergence, Scatter, and Sparsity Plots: CIFAR10 IID . . . 211
D.4 UR-ACGSV Convergence, Scatter, and Sparsity Plots: SST2 IID 212
D.5 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: MNIST IID . . . 216
D.6 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: FA-MNIST IID . 217
D.7 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: CIFAR10 IID . . 218
D.8 OTOPK-CFFL Convergence, Scatter, and Sparsity Plots: SST2 IID 219
D.9 DPR-CFFL Convergence and Scatter Plots: MNIST IID 222
D.10 DPR-CFFL Convergence and Scatter Plots: FA-MNIST IID 223
D.11 DPR-CFFL Convergence and Scatter Plots: CIFAR10 IID 224
D.12 DPR-CFFL Convergence and Scatter Plots: SST2 IID 225
D.13 DFR-ACGSV Convergence and Scatter Plots: MNIST IID 229
D.14 DFR-ACGSV Convergence and Scatter Plots: FA-MNIST IID 230
D.15 DFR-ACGSV Convergence and Scatter Plots: CIFAR10 IID 231
D.16 DFR-ACGSV Convergence and Scatter Plots: SST2 IID 232

E.1 Original CFFL CLA equation Convergence and Scatter Plots 235

List of Tables

2.1 Contribution scheme characteristics comparison 22

3.1 State-of-the-art federated learning frameworks comparison 32

4.1 Power-law training dataset distribution across clients. 66
4.2 Class Label Imbalance training dataset distribution across clients for MNIST and FA-MNIST datasets 67
4.3 Class Label Imbalance training dataset distribution across clients for CIFAR10 dataset. 67
4.4 Validation dataset sizes 67
4.5 Test dataset sizes 67
4.6 Simulation hyperparameters 71
4.7 CFFL Pearson correlation values and mean GMD 74
4.8 ACGSV Pearson Correlation values and mean GMD 80
4.9 CFFL and ACGSV: Pearson correlation comparison 80
4.10 CFFL and ACGSV: GMD comparison 81
4.11 CFFL Kendall Correlation values 91
4.12 ACGSV Kendall Correlation values 91
4.13 ACGSV and CFFL: Pearson, Kendall and GMD in POW partitioning . . . 93
4.14 ACGSV and CFFL: Pearson, Kendall and GMD in CLA partitioning . . . 93
4.15 ACGSV and CFFL: Pearson, Kendall and GMD in POW partitioning . . . 94
4.16 ACGSV and CFFL: Pearson, Kendall and GMD in CLA partitioning . . . 94

5.1 CFFL reward density upper bounds . . . 105
5.2 CFFL local accumulated gradient retention . . . 108
5.3 UR-ACGSV: Pearson, Kendall, GMD, and MPAD under POW partitioning 116
5.4 UR-ACGSV: Pearson, Kendall, GMD, and MPAD under CLA partitioning 116
5.5 UR-ACGSV Pearson Correlation values . . . 126
5.6 UR-ACGSV: Shapley-based Pearson, Kendall, GMD, and MPAD under POW partitioning . . . 127
5.7 UR-ACGSV: Shapley-based Pearson, Kendall, GMD, and MPAD under CLA partitioning . . . 127
5.8 OTOPK-CFFL: Pearson, Kendall, GMD, and MPAD under POW partitioning 131
5.9 OTOPK-CFFL: Pearson, Kendall, GMD, and MPAD under CLA partitioning 132
5.10 OTOPK-CFFL: Shapley-based Pearson, Kendall, GMD, and MPAD under POW partitioning . . . 142
5.11 OTOPK-CFFL: Shapley-based Pearson, Kendall, GMD, and MPAD under CLA partitioning . . . 142
5.12 DPR-CFFL: Pearson, Kendall, GMD, and MPAD under POW partitioning 146
5.13 DPR-CFFL: Pearson, Kendall, GMD, and MPAD under CLA partitioning 146
5.14 DPR-CFFL: Shapley-based Pearson, Kendall, GMD, and MPAD under POW partitioning . . . 155
5.15 DPR-CFFL: Shapley-based Pearson, Kendall, GMD, and MPAD under CLA partitioning . . . 155
5.16 DFR-ACGSV: Pearson, Kendall, GMD, and MPAD under POW partitioning 159
5.17 DFR-ACGSV: Pearson, Kendall, GMD, and MPAD under CLA partitioning 159
5.18 DFR-ACGSV: Shapley-based Pearson, Kendall, GMD, and MPAD under POW partitioning . . . 168
5.19 DFR-ACGSV: Shapley-based Pearson, Kendall, GMD, and MPAD under CLA partitioning . . . 168

A.1 Notations and definitions . . . 194

C.1 Simulation hyperparameters - ACGSV with longer training . . . 197

D.1 CFFL Shapley Pearson Correlation values 202
D.2 CFFL Shapley Kendall Correlation values 203
D.3 CFFL per-run GMD values 203
D.4 CFFL per-run MPAD values 203
D.5 ACGSV Shapley Pearson Correlation values 204
D.6 ACGSV Shapley Kendall Correlation values 204
D.7 ACGSV GMD values 205
D.8 ACGSV MPAD values 205
D.9 UR-ACGSV Kendall Correlation values 206
D.10 UR-ACGSV GMD values 206
D.11 UR-ACGSV MPAD values 207
D.12 UR-ACGSV Shapley Pearson Correlation values 207
D.13 UR-ACGSV Shapley Kendall Correlation values 208
D.14 OTOPK-CFFL Pearson Correlation values 213
D.15 OTOPK-CFFL Kendall Correlation values 213
D.16 OTOPK-CFFL GMD values 214
D.17 OTOPK-CFFL MPAD values 214
D.18 OTOPK-CFFL Shapley Pearson Correlation values 214
D.19 OTOPK-CFFL Shapley Kendall Correlation values 215
D.20 DPR-CFFL Pearson Correlation values 220
D.21 DPR-CFFL Kendall Correlation values 220
D.22 DPR-CFFL GMD values 221
D.23 DPR-CFFL MPAD values 221
D.24 DPR-CFFL Shapley Pearson Correlation values 226
D.25 DPR-CFFL Shapley Kendall Correlation values 226
D.26 DFR-ACGSV Pearson Correlation values 227
D.27 DFR-ACGSV Kendall Correlation values 227

D.28 DFR-ACGSV GMD values 228
D.29 DFR-ACGSV MPAD values 228
D.30 DFR-ACGSV Shapley Pearson Correlation values 233
D.31 DFR-ACGSV Shapley Kendall Correlation values 233

E.1 CFFL Original CLA Equation Simulation hyperparameters 234

List of Abbreviations

AI Artificial Intelligence 7

CML Collaborative Machine Learning 10

CNN Convolutional Neural Network 69, 70

FAFL Fairness-Aware Federated Learning 14, 27

FL Federated Learning 10

FLCE Federated Learning Contribution Estimation 14, 17–19

GMD Gini Mean Difference 30

IAFL Incentive-Aware Federated Learning 13, 27

IID Independent and Identically Distributed 65

LOO Leave-One-Out 20, 21

ML Machine Learning 7

MLP Multilayer Perceptron 69

non-IID non-Independent and Identically Distributed 22

SV Shapley Value 21

Chapter 1

Introduction

Federated Learning (FL) [1] is a collaborative machine learning paradigm that allows multiple data owners to participate in a shared training process without directly exchanging or exposing their local datasets. This paradigm makes collaborative training possible in settings where data cannot be centralized due to privacy, ownership, regulatory, or operational constraints.

Existing FL research often assumes that clients are willing to dedicate their computing, communication, and data resources to the learning process [2, 3, 4, 5, 6]. In practice, however, this assumption does not always hold, since participation in FL imposes non-trivial costs on clients, including local computation, communication bandwidth, energy consumption, and the collection, processing, storage, and maintenance of training data. Therefore, sustaining long-term participation in FL requires reward distribution mechanisms that incentivize clients to remain engaged in the collaborative learning [7, 8, 9].

In the context of FL, contribution fairness dictates that clients should receive rewards that are commensurate with their contributions to the training outcome [7, 10, 11]. This thesis studies contribution fairness in FL due to its practical relevance to real-world FL deployments, especially in settings where valuable training data is distributed across independent and competing organizations that may be reluctant to collaborate without a clear and fair allocation of rewards. From this perspective, contribution fairness is important because it connects each client's reward to the value it contributes to the collaborative training process. This notion of fairness has motivated significant research on federated learning contribution estimation (FLCE) algorithms, whose objective is to quantify each client's contribution to the training and use the estimated scores to determine the reward it should receive [8, 11, 12, 13, 14, 15, 16, 17].

Many of the existing studies on FLCE focus predominantly on the contribution estimation algorithm optimization, and assume that the final estimated contribution scores can be used to derive post-training monetary rewards to incentivize the clients [15, 16, 17, 18, 19]. However, the distribution of post-training monetary rewards present multiple practical limitations. First, monetary incentives require either a predefined budget or an external source of revenue; therefore, without a clear source of capitalization, clients may not join the training process [12, 13]. Second, there is no well-defined mapping between existing currencies (e.g., US dollars, Bitcoin) and the model parameters or gradients exchanged during FL training [7, 14], making it non-trivial to assign a monetary value to a specific client's model parameter or gradient. Third, clients may leave the federation before training is completed due to unreliable communication with the central server, limited local resources, or the delayed compensation between their training investment and the eventual payoff [5, 6], making the post-training monetary incentive ineffective.

In addition to these practical limitations, the existing frameworks that distribute monetary rewards [12, 13, 15, 16, 17, 18, 19] allow all clients to access the optimized global model, regardless of their estimated contributions. This design introduces a more fundamental fairness problem: low-contributing clients may receive a smaller monetary payoff after training, yet still obtain the same model as high-contributing clients. This issue becomes especially important in real-world FL applications for organizations that use machine learning models as part of their business models (e.g., products, services), where accessing a better-performing model may provide greater practical values than the monetary reward. As a result, the model itself becomes an implicit non-monetary reward, and distributing it uniformly across the federation contradicts the purpose of contribution-aware reward allocation.

Consequently, another area of research leverages each client's estimated contribution scores generated by the FLCE algorithm to derive personalized gradient rewards, where the clients with higher estimated contribution receive better gradients than clients with lower estimated contribution [7, 14, 20, 21, 22, 23, 24, 25, 26]. The overarching goal of these studies is to transform estimated contribution into an in-training reward mechanism that directly shapes the learning of each client's local model and induces client-level model differentiation, so that clients with higher estimated contributions converge to models with better predictive performance.

In this research domain, the server aggregates the gradients generated by the clients and then derives, for each client, a personalized estimated-contribution-based version of the aggregated gradient. One commonly used approach for generating these personalized reward gradients is the Top-K sparsification [27], where only the K coordinates with the largest absolute magnitude of the aggregated gradient are preserved and all remaining

coordinates are set to zero. Therefore, the value K is defined by the estimated contribution score and it is uses at every training round to determine the reward sparsity degree. When applied at every communication round, the intended effect is to make clients with higher estimated contribution receive denser rewards and converge into models that achieve better performance, while clients with lower estimated contribution obtain sparser rewards and converge into models that achieve lower performance.

Although Top-K sparsification has been used as a non-monetary reward mechanism to achieve contribution fairness in FL [14, 21, 22, 23, 24, 25, 26], it was developed as a gradient compression technique to reduce communication costs in distributed learning, alongside other gradient compression methods [27, 28, 29]. Originally, the goal of gradient compression techniques is to allow models trained with sparsified gradients to achieve the same or comparable convergence to models trained with denser gradients. Given this original purpose, the use of Top-K sparsification as a reward-generation mechanism raises an important question: how can a gradient compression technique originally designed to preserve convergence also be used to induce model convergence degradation required by contribution fairness in FL?

Additionally, existing frameworks that adopt Top-K sparsification-based rewards also differ across several framework design choices, including the FLCE algorithm, the sparsification rule, the aggregation strategy, and whether the clients' locally accumulated gradient are also used during its model update [14, 21, 22, 23, 24, 25]. As a result, when high-contributing clients obtain better-performing models than low-contributing clients, the observed model differentiation cannot be credited directly to the FLCE. It may be produced by the interaction among FLCE algorithm, the local gradient retention, and how the sparsified reward itself is being generated [7, 14, 21, 22, 23, 24, 25, 26].

1.1 Motivation

This ambiguity constitutes the first objective of this thesis: to decouple contribution estimation, Top-K Sparsification, and local gradient retention in order to determine their individual effects on contribution fairness, specifically on the reference contribution rank preservation and model differentiation after federated training. This study is important because existing frameworks often assume that improving the FLCE algorithm will produce more precise reward sparsity assignments and, consequently, stronger contribution fairness. However, this assumption is difficult to justify without first understanding whether Top-K sparsification is capable of inducing estimated-contribution-based model differentiation and reference contribution rank preservation.

Equally important to understanding the effects of Top-K sparsification is understanding the effects of clients' local gradient retention. In some frameworks [14, 21, 24], the client model is updated only with the Top-K sparsified reward returned by the server, while in others [7, 22, 23, 30], clients preserve their locally accumulated gradients computed during training and combine it with the server-generated reward. This distinction is important, because the client model update rule in these frameworks introduce an additional component that is not directly governed by the FLCE algorithm or the contribution-score-to-reward mapping pipeline. Consequently, the same FLCE and estimated-contribution-based reward-generation pipeline may produce different model differentiation and reference contribution rank preservation depending on whether locally accumulated gradients are discarded or retained.

A second motivation emerged from the study of reproduced FL frameworks and the evaluation of contribution fairness. Existing frameworks assess contribution fairness by computing Pearson's correlation coefficient between clients' final test accuracies after federated training and their respective reference contribution values [7, 14, 21, 22, 23, 24, 25, 26]. Although the Shapley Value provides a more principled measure of each client's contribution, its computation requires evaluating all possible client coalitions, which scales exponentially with the federation size. Consequently, these frameworks commonly use standalone test accuracy as a tractable proxy for each client's reference contribution value. Under this protocol, a high Pearson's r coefficient, typically $r \geq 0.7$, is interpreted as strong evidence of contribution fairness.

However, the experiments conducted in this thesis showed that a high Pearson's r correlation coefficient can still arise when most clients converge to models with similar, and sometimes identical final federated accuracies, while only a small subset of clients achieve lower performance. In such cases, Pearson's coefficient indicates a strong contribution fairness, even though the framework reward generation pipeline did not produce a strong reference contribution ranking preservation and model differentiation. This observed limitation motivates the use of metrics that evaluate these two distinct and key aspects of contribution fairness: whether the clients' reference contribution ranking is preserved and whether their final models exhibit meaningful performance differentiation.

1.2 Research Contributions

This thesis addresses the research gaps identified above through the following contributions:

1. **A critique of Pearson's r as a contribution fairness metric.** Empirical ev-

idence shows that a high correlation can be produced by a small subset of lower-performing clients while the remaining clients converge to near-identical federated accuracies, so r alone does not measure contribution fairness.

2. **An evaluation protocol based on rank preservation and model differentiation.** Kendall's τ measures whether the ordering of clients is preserved between standalone and federated performance; the Gini Mean Difference (GMD) quantifies differentiation among client models.

3. **Evidence that reward sparsity does not map linearly onto performance degradation.** Differentiation emerges only under sustained high sparsity (at least 70% in the studied setting); below that threshold, clients converge to models of nearly identical performance.

4. **Identification of locally accumulated gradient retention as a confounding factor.** High Pearson r values between standalone and federated performance can be driven primarily by retained local gradients, which lie outside the framework's contribution estimation and reward generation pipeline.

The research contributions and empirical findings presented in this thesis are primarily based on standalone test accuracy as the client reference contribution value, in order to remain consistent with the evaluation approach adopted in the existing Contribution Fairness literature.

Nevertheless, to further validate and strengthen these findings, I also compute the Shapley Values for each client using test accuracy as the coalition utility. The results obtained with the Shapley Values-based analysis corroborate the main findings of this thesis, showing that the limitations of Top-K sparsified rewards and the confounding effect of local-gradient retention remain present when client contribution is evaluated through coalition-based marginal utility.

1.3 Thesis Outline

This thesis is organized into the following chapters:

- **Chapter 2:** This chapter introduces the foundational concepts, notation, and definitions used throughout the thesis. I first present the background on machine learning

and federated learning, followed by the concepts of Incentive-Aware FL and Fairness-Aware FL. I then define contribution fairness and discuss how it is measured in related work. Next, I introduce the Top-K sparsification concept and algorithm, the main components of a Federated Learning Contribution Estimation (FLCE) algorithm, and the statistical background required to understand the rank-based correlation coefficients and the Gini Mean Difference (GMD) used later in the thesis.

- **Chapter 3:** This chapter reviews existing federated learning frameworks developed around the notion of contribution fairness and contribution-aware incentivization. I organize the literature according to the type of reward mechanism adopted by each framework, discuss the rationale used to select the frameworks studied in this thesis, and describe in details the selected frameworks.

- **Chapter 4:** This chapter presents the experimental methodology used to compare the selected frameworks and analyzes their fairness and model-differentiation outcomes. The chapter also evaluates the limitations of using Pearson's correlation coefficient as the sole metric for Contribution Fairness and motivates the use of rank-based correlation metrics together with the GMD.

- **Chapter 5:** This chapter presents ablation studies designed to isolate the effects of the framework components, including Top-K sparsification and local gradient retention. These experiments investigate how individual design choices affect model differentiation and Contribution Fairness.

- **Chapter 6:** This chapter summarizes the findings of this thesis, discusses their implications for current research, and presents the study limitations and directions for future work.

Chapter 2

Background

In this chapter, I establish the background knowledge required for the remainder of this thesis. I first introduce the general concept of machine learning, followed by federated learning and the foundational framework proposed by McMahan *et al.* [1]. Then, I discuss the need for client incentivization for sustaining long-term participation in practical federated learning deployments. Building on this motivation, I introduce Incentive-Aware Federated Learning and distinguish between the two main reward categories used to incentivize participation: monetary and non-monetary rewards. I then present Fairness-Aware Federated Learning and formalize the fairness notion adopted in this thesis. This definition provides the basis for discussing Top-K sparsification, a commonly adopted mechanism for distributing contribution-aware non-monetary rewards. I then present Federated Learning Contribution Estimation, with emphasis on the contribution estimation schemes and data utility metrics used by state-of-the-art research. Finally, I discuss correlation coefficients that are underexplored in the related literature for evaluating contribution fairness, and I introduce the Gini Mean Difference as a metric for measuring model differentiation.

2.1 Machine Learning

Machine Learning (ML) is a field of Artificial Intelligence (AI) concerned with the development of learning algorithms that construct models from data. These models encode patterns extracted from training samples and use the acquired knowledge to generalize to unseen inputs. Rather than following explicitly programmed rules, ML models improve their behavior by adjusting internal parameters during training. This learning process

enables them to perform tasks such as classification, regression, prediction, and decision-making.

For a machine learning problem, the objective is to obtain a set of model weights, denoted by $\vec{w}$ of dimension D, that minimizes the training loss on a dataset $\mathcal{D}$. This objective is formally written as

$$\min_{\vec{w} \in \mathbb{R}^D} f(\vec{w}; \mathcal{D}). \tag{1}$$

Deep learning is an area of study within ML centered on multilayer neural networks. These models process input data through a hierarchy of layers, where each layer receives the representation produced by the previous layer and transforms it into a new representation. In a deep learning model, each layer is composed of neurons, which are computational units that receive input values, compute a weighted sum of these inputs, apply an activation function, and pass the result to the next layer, in a process called forward propagation.

Neurons are typically organized into an input layer, one or more hidden layers, and an output layer. The input layer receives the data representation used by the model, the hidden layers learn intermediate representations, and the output layer produces the final prediction. During forward propagation, the input data are progressively transformed into representations that allow the model to capture complex patterns and relationships. This representation-learning process enables deep neural networks to perform tasks such as image classification, natural language processing, and regression. In a classification problem, for example, the model learns a mapping between input samples and predefined class labels. Figure 2.1 presents a simplified neural network architecture with one hidden layer.

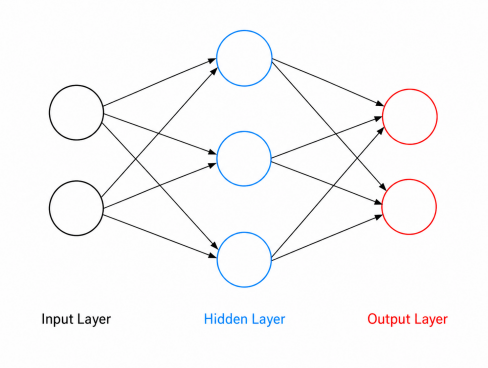


Figure 2.1: Simplified Neural Architecture. Black circles represent the input layer, blue circles represent the hidden layer, and red circles represent the output layer.

The training process of a neural network aims to minimize a loss function, as defined in Equation 1, which measures the discrepancy between the model prediction and the true target value. In a fully connected neural network, each neuron in layer l is connected to each neuron in layer $l+1$, and each connection is associated with a weight that controls the strength and direction of the signal transmitted between neurons.

Let $\mathbf{a}_l$ denote the vector of activations produced by layer l, $\mathbf{W}_l$ denote the weight matrix connecting layer l to layer $l+1$, and $\mathbf{b}_{l+1}$ denote the bias vector associated with layer $l+1$. In vector form, the pre-activation values of layer $l+1$ can be written as in Equation 2.

$$\mathbf{z}_{l+1} = \mathbf{W}_l \times \mathbf{a}_l + \mathbf{b}_{l+1} \tag{2}$$

The resulting pre-activation vector $\mathbf{z}_{(l+1)}$ is then passed through a nonlinear activation function $\sigma(\cdot)$, producing the activation vector of the next layer, as shown in Equation 3:

$$\mathbf{a}_{l+1} = \sigma\left(\mathbf{z}_{l+1}\right) \tag{3}$$

This forward propagation process is repeated across all layers until the output layer produces a prediction. At this stage, the loss function is then evaluated by comparing the predicted output with the corresponding ground-truth label. After the loss is computed, backpropagation is used to calculate the gradients of the loss with respect to each trainable parameter in the network. This process propagates the prediction error backward through the network and applies the chain rule of calculus to determine how changes in each weight and bias affect the final loss. The resulting gradients are then used by an optimization algorithm to update the model parameters in a direction that reduces the loss.

Gradient descent is one of the most widely used optimization methods for this purpose. In standard gradient descent, the model computes the gradient using the entire training dataset before performing each parameter update. Although this approach provides a complete estimate of the loss gradient, it becomes computationally expensive for large datasets. Therefore, stochastic gradient descent and mini-batch gradient descent are commonly used in practice. These methods update the model parameters using either a single training sample or a small batch of samples at a time, reducing computational cost while introducing stochasticity into the optimization process. The complete training cycle, consisting of forward propagation, loss evaluation, backpropagation, and parameter update, is repeated over multiple iterations or epochs until the model reaches a satisfactory level of convergence.

2.2 Federated Learning

Federated Learning (FL) as a privacy-oriented Collaborative Machine Learning (CML) paradigm that enables a set of clients $\mathcal{N} = \{1, \ldots, N\}$ to collaboratively train a shared model without directly sharing their local training data with the server or with other clients. In the foundational FL framework proposed by McMahan *et al.* [1], each client $i \in \mathcal{N}$ owns a private dataset $\mathcal{D}_i = \{(x_{i,1}, y_{i,1}), \ldots, (x_{i,k_i}, y_{i,k_i})\}$ of size $|\mathcal{D}_i| = k_i$. The dataset associated with the grand coalition of all participating clients is denoted by $\mathcal{D}_\mathcal{N} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_N$, with total size $|\mathcal{D}_\mathcal{N}| = \sum_{i=1}^{N} |\mathcal{D}_i|$.

The objective in FL is to learn a D-dimensional parameter vector $\vec{w} \in \mathbb{R}^D$ that minimizes the global loss function $f_g(\vec{w})$, formally written as

$$\min_{\vec{w} \in \mathbb{R}^D} f_g(\vec{w}; \mathcal{D}_\mathcal{N}). \tag{4}$$

The global loss is commonly defined as a weighted aggregation of the clients' local empirical losses:

$$f_g(\vec{w}) = \sum_{i=1}^{N} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{N}|} f_i(\vec{w}; \mathcal{D}_i) \tag{5}$$

In the federated learning training procedure introduced by McMahan *et al.* [1], training is performed over communication rounds $t \in \{0, \ldots, T\}$. The training initialization happens at $t = 0$, where the server creates an initial global model, denoted by $\vec{w}_{0,0}$, and broadcasts it to all clients $i \in \mathcal{N}$. Therefore, all clients start from the same parameter vector as the server, such that $\vec{w}_{i,0} = \vec{w}_{0,0}$.

At each communication round $t > 0$, each client starts from the most recent global model received from the server and performs E local epochs of training on its private dataset $\mathcal{D}_i$. During local training, the client optimizes its local objective by computing gradients of the loss function with respect to its model parameters and updating these parameters through an optimization algorithm such as stochastic gradient descent after each local epoch. After completing the local training procedure, client i obtains an updated local model, denoted by $\vec{w}_{i,t}$.

Once the participating clients have produced their local models, they send the models $(\vec{w}_{i,t})$ to the server. The server then aggregates the received models into a new global model by computing a weighted average of the client models based on the number of local

training samples held by each client. Assuming a full client participation, the global model is computed as

$$\vec{w}_{\mathcal{N},t} \leftarrow \sum_{i \in \mathcal{N}} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{N}|} \vec{w}_{i,t}. \tag{6}$$

The server then replaces the previous global model, denoted as $\vec{w}_{\mathcal{N},t-1}$, with the new aggregated model and broadcasts $\vec{w}_{\mathcal{N},t}$ to the clients for the next communication round. After receiving the updated global model, each client uses $\vec{w}_{\mathcal{N},t}$ as the starting point for its next local training phase. Therefore, the training process alternates between local optimization on private client datasets and global aggregation at the server. Algorithm 1 shows each step of the federated learning procedure proposed by McMahan *et al.* [1].

Algorithm 1 - Federated Learning with Federated Averaging. All clients are assumed to start from the same initial model, such that $\vec{w}_{i,0} = \vec{w}_{0,0}$. The server defines the training hyperparameters, where B is the mini-batch size, E is the number of local epochs, η is the learning rate, and T is the total number of communication rounds.

for $t = 1$ to T **do**
  Role: client $i \in \mathcal{N}$
    $\mathcal{B} \leftarrow \mathcal{D}_i / B$ ▷ Splits training dataset $\mathcal{D}_i$ into $\mathcal{B}$ mini-batches
    $\vec{w} \leftarrow \vec{w}_{i,t-1}$
    for $e = 1$ to E
      for batch $b \in \mathcal{B}$
        $\vec{w} \leftarrow \vec{w} - \eta_t \nabla f_i(\vec{w}; b)$ ▷ Client updates $\vec{w}_{i,t-1}$ locally
      end for
    end for
    $\vec{w_i}_{,t} \leftarrow \vec{w}$
    Send $\vec{w_i}_{,t}$ to Server

  Role: Server
    Receive $\vec{w_i}_{,t}$ from each client $i \in \mathcal{N}$
    $\vec{w}_{\mathcal{N},t} \leftarrow \sum_{i \in \mathcal{N}} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{N}|} \cdot \vec{w}_{i,t}$ ▷ $\vec{w}_{\mathcal{N},t}$: New global model
    Send $\vec{w}_{\mathcal{N},t}$ to each client $i \in \mathcal{N}$

  Role: client $i \in \mathcal{N}$
    Receive $\vec{w}_{\mathcal{N},t}$ from Server
    $\vec{w_i}_{,t} \leftarrow \vec{w}_{\mathcal{N},t}$ ▷ Client i updates its local model
end for

Originally, the FL framework proposed by McMahan *et al.* [1] also allowed the server to sample only a subset of clients $\mathcal{S} \subseteq \mathcal{N}$ at each round using a client participation fraction C. In that case, only the selected clients participate in local training and aggregation during round t. I omit this client sampling step for simplicity and assume full client participation.

2.3 Sustaining Client Participation Through Incentivization

Several federated learning studies [1, 2, 4, 31, 32, 33, 34] focus on the optimization of aggregation protocols, client sampling strategies, or other system-level aspects of the collaborative training while assuming that clients are willing to join the federation and remain active throughout the training process. This assumption is useful for studying the core mechanics of federated learning, but it abstracts away a central practical challenge: participation is not cost-free. In realistic deployments, clients incur non-trivial costs associated with local computation, communication bandwidth, energy consumption, and the collection, processing, storage, and maintenance of training data.

As a result, rational or self-interested clients may refuse to participate or leave the federation if the training process does not provide a clear incentive for participation. This incentive can take the form of monetary compensation, where clients are financially rewarded for contributing with their model updates [9, 12, 35], but it can also take the form of non-monetary rewards, where clients receive models, updates, or learning benefits whose quality depends on their role in the collaborative training process [14, 17, 22, 23]. Although incentives can be allocated according to different rules, they are expected to be distributed fairly across the federation. The challenge, however, is that fairness is not a universal concept, and its meaning depends on the objective of the federated learning system.

Therefore, incentive design in federated learning cannot be separated from the fairness notion adopted by the system. A reward mechanism can only be evaluated as fair after defining what fairness means in the corresponding federated learning setting. This creates a direct connection between incentive design, fairness, and contribution estimation: to allocate rewards fairly, the system must first estimate how each client contributed to the training outcome.

In the next section, I examine the two types of incentives introduced in this section and clarify which type of incentive this thesis emphasizes. This distinction is necessary before discussing fairness, because the fairness criterion adopted in this work depends on what the incentive mechanism is expected to distribute.

2.4 Incentive-Aware FL

The Incentive-Aware Federated Learning (IAFL) literature [7, 8, 10, 11, 14, 36, 37] focuses on two broad incentive categories: **monetary rewards** and **non-monetary rewards**. Below, I bring the literature definition on these two rewards types:

- **Monetary rewards**: Incentivizes participation through payments, such as currency or cryptocurrency. It typically assumes the existence of a budget, either predetermined or accumulated across training rounds, which is allocated to clients according to a reward rule. Typically, when a framework adopts this incentive type, clients continue to receive the same global model at the beginning of each round, while the incentive is provided separately in monetary form. [8, 12, 15, 18]
- **Non-monetary rewards**: Incentivizes participation by controlling the model learning in each client on the federation, rather than by providing an external monetary payment. In this case, the reward may be implemented through personalized models, differentiated gradient or selective synchronization of the global model with sampled clients. The behavior of this incentive mechanism depends directly on the fairness notion adopted by the system, and it can be used to reward stronger contributors with model updates that lead to better predictive performance. Conversely, it can be used to reduce client disparities, by helping clients that were sampled less frequently or that received lower-quality models during training and lagged behind the rest of the federation. [7, 14, 23, 30]

Although both monetary and non-monetary incentives can be used to encourage client participation, in this thesis I focus on non-monetary reward mechanisms for the reasons discussed in Chapter 1. These mechanisms are particularly relevant to real-world FL deployments in which valuable training data are held by independent, and sometimes competing, organizations that may be reluctant to collaborate unless the benefits of training are fairly allocated. In such settings, the trained model is itself a valuable resource, since organizations may use it as part of their products, services, or decision-making pipelines, meaning that access to a better-performing model can provide greater practical value than a monetary reward alone.

Before discussing the mechanisms used to implement this type of reward, it is necessary to define the fairness notion adopted in this thesis. In Section 2.5, I formalize this fairness notion and use it as the basis for analyzing related work on federated learning contribution estimation and contribution-aware incentivization. Later, in Section 2.6, I discuss the

gradient compression technique, that is commonly used to distribute personalized non-monetary rewards under this fairness definition. Throughout this thesis, I use the terms *incentive* and *reward* interchangeably. Unless otherwise stated, both terms refer to the recompense that a client receives based on its estimated contribution to the global training process.

2.5 Fairness-Aware FL

Fairness-Aware Federated Learning (FAFL) studies the fairness challenges that arise in federated learning, including, but not limited to, client selection, distribution of model performance across participants, group-level disparities, and reward allocation among contributors. FAFL defines different fairness notions for different aspects of the federated learning process, and these notions often pursue distinct, and sometimes orthogonal, objectives. Therefore, they should not be directly compared without first specifying the fairness criterion being considered [7, 10, 37]. Among the existing fairness notions, this thesis focuses on **Contribution Fairness**, as stated in the thesis' Introduction (Chapter 1), and is formally defined in Definition 1.

Definition 1 (Contribution Fairness)**.** Contribution Fairness[1] is a fairness notion grounded in cooperative game theory in which a client's incentive should be commensurate to its contribution to the federated learning training outcome. This notion is primarily concerned with the fair allocation of incentives among individual clients, rather than increasing the global FL model accuracy [7, 10, 14, 22, 23, 37, 38].

Based on this fairness definition, extensive research has focused on developing game-theory-based algorithms that can estimate how much each client contributes to the federated learning outcome, such that the estimated contribution can guide reward allocation to each client. These methods are commonly referred to as Federated Learning Contribution Estimation (FLCE) algorithm, which I discuss in more details in Section 2.7.

However, since each framework relies on its own FLCE algorithm to estimate client contributions and uses these estimates to determine the rewards assigned to each client, correlating their FL test accuracy values with the estimated contribution values would primarily indicate whether the framework's reward allocation is consistent with its own

[1]Related literature use the term "Collaborative Fairness" for the same fairness notion. Both terms are synonyms, but I adopt the term "Contribution Fairness" because it more explicitly captures the principle that client rewards should be commensurate to client contributions.

contribution estimator. For this reason, contribution fairness must be evaluated using external reference contribution values that are independent of the framework being assessed.

Definition 2 (Reference Contribution)**.** A *reference contribution* is a per-client value that quantifies a client's contribution independently of the federated learning framework under evaluation. Reference contributions serve as the ground truth against which a framework's contribution estimates and reward allocations are compared, enabling fairness benchmarking across distinct frameworks.

The related literature [10, 11, 18, 37] adopts the Shapley Values as a principled method for estimating each client's marginal contribution to collaborative training, thereby establishing reference contribution values and ranks. Furthermore, because Shapley Values average each client's marginal contribution across all possible participant coalitions, they are less sensitive to any particular coalition composition.

However, despite their theoretical appeal, Shapley Values are assumed computationally impractical in ML and FL. Computing them requires evaluating a number of client coalitions that grows exponentially with the federation size on the order of $\mathcal{O}(2^{|\mathcal{N}|})$, where $|\mathcal{N}|$ denotes the number of clients in the federation, which would require retraining or evaluating a large number of models to compute each client's reference contribution value and rank [11, 18].

Therefore, a practical alternative adopted in the literature is to use each client's final standalone test accuracy as a tractable method for establishing each's clients reference contribution, and evaluate fairness by computing the Pearson's r correlation coefficient between the final test accuracy achieved by each client in standalone training and the final test accuracy achieved by the same client after federated collaboration [7, 14, 21, 22, 25, 23, 39, 40, 41, 42].

In this context, a Pearson's r correlation coefficient value closer to 1 is interpreted as evidence of a fairer framework, while values closer to -1 indicates an unfair framework, and values closer to 0 indicate that the framework does not promote contribution fairness. This fairness metric is formally defined in Definition 3.

Definition 3 (Pearson's r Correlation Coefficient for Contribution Fairness)**.** Let x_i denote the final test accuracy achieved by client $i \in \mathcal{N}$ in standalone training, $\bar{x}$ denote the mean standalone test accuracy, y_i denote the final test accuracy achieved by the same client under federated training, and $\bar{y}$ the mean federated test accuracy. Standalone training refers to the setting in which each client trains exclusively on its local dataset $\mathcal{D}_i$, without exchanging data, gradients, model updates, or model parameters with other clients. The

Pearson's r Correlation Coefficient used to evaluate Contribution Fairness is defined as

$$r = \frac{\sum_{i \in \mathcal{N}} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i \in \mathcal{N}} (x_i - \bar{x})^2} \sqrt{\sum_{i \in \mathcal{N}} (y_i - \bar{y})^2}}, \tag{7}$$

As explained in the literature that adopts this correlation as a measure of fairness [7, 14, 21, 22, 25, 23, 39, 40, 41, 42], standalone model accuracy represents each client's individual learning capability on its own local data and establishes a performance ordering among clients. Therefore, when the same clients participate in an FL framework designed to promote contribution fairness, those with higher standalone test accuracy should also obtain higher final test accuracy after federated training. This expectation is particularly relevant in real-world FL deployments involving independent and potentially competing organizations. In such settings, all participants may benefit from collaboration, but a client capable of training a stronger model from its own data would have little incentive to join a federation if the reward mechanism allowed lower-performing clients to obtain equal or better final models without making comparable contributions.

In addition to the definition and justification provided in the literature, my interpretation is that the Pearson's r correlation coefficient has been used as a metric that captures more than whether two performance profiles vary linearly together. Since Pearson's coefficient is computed from the raw accuracy values and their deviations from their respective means, it is also influenced by the magnitude of the gaps between client accuracies and by how these gaps align across standalone and federated performance.

After defining the fairness notion and the metric used to evaluate it, I can now discuss the mechanisms commonly used to implement contribution-aware non-monetary rewards. The next section introduces Top-K sparsification, a widely used technique for distributing personalized gradients to clients with the goal of inducing contribution-aligned model differentiation.

2.6 Top-K Sparsification

Federated learning frameworks that seek to promote contribution fairness through non-monetary rewards [14, 21, 22, 23] commonly rely on the server to generate client-specific reward vectors through Top-K sparsification, which is a technique that preserves only the K coordinates with the largest absolute magnitude values of an input vector and sets all remaining coordinates to zero.

In these frameworks, in each round of training, the server first computes an aggregated gradient from the accumulated gradients uploaded by the clients, then executes its FLCE algorithm to estimate each client's contribution and uses these estimates to determine a client-specific value of K. Therefore, clients with higher estimated contribution receive larger values of K and, therefore, denser reward vectors, while clients with lower estimated contribution receive smaller values of K and sparser rewards. Algorithm 2 presents the Top-K sparsification procedure.

Algorithm 2 - Top-K Sparsification. The algorithm takes as input a gradient vector $\vec{w}$ and a value K which defines the number of largest-magnitude coordinates to preserve. If the flattened vector has dimension d, the resulting sparsity is $1 - K/d$.

function TOP-K($\vec{w}, K$)
  $\vec{f} \leftarrow \emptyset$ ▷ Initialize an empty flattened vector
  for layer $\vec{g}$ in $\vec{w}$ **do**
    $\vec{f}$.append($\vec{g}$) ▷ Append the entries of layer $\vec{g}$ to $\vec{f}$
  end for
  $d \leftarrow |\vec{f}|$ ▷ Number of entries in the flattened vector
  $K \leftarrow \min(K, d)$ ▷ Ensure that at most d entries are preserved
  $\pi \leftarrow \text{ArgsortDescending}(|\vec{f}|)$ ▷ Sort indices by decreasing absolute magnitude
  $\mathcal{I}_K \leftarrow \emptyset$ ▷ Initialize the set of selected indices
  for $m \leftarrow 1$ to K **do**
    $\mathcal{I}_K \leftarrow \mathcal{I}_K \cup \{\pi[m]\}$ ▷ Select the index of the m-th largest entry
  end for
  $\vec{\nu} \leftarrow \vec{0}$ ▷ Initialize the sparsified vector with zeros
  for each $j \in \mathcal{I}_K$ **do**
    $\vec{\nu}[j] \leftarrow \vec{f}[j]$ ▷ Preserve the selected coordinate
  end for
  $\vec{\nu} \leftarrow \text{Reshape}(\vec{\nu})$ ▷ Restore the layer-wise structure of $\vec{w}$
  return $\vec{\nu}$
end function

Since these rewards are computed by the server and distributed back to the clients, the server directly controls how much of the aggregated training signal each client receives. The intended effect is to shape each client's local model by controlling the sparsity of its reward, thus inducing model differentiation.

The use of Top-K sparsification as a reward mechanism in these federated learning frameworks is an adaptation of a technique originally developed to reduce communication

costs in decentralized machine learning. In its original setting, clients do not transmit their entire locally accumulated gradients in full, instead, each client compresses its accumulated gradient by sending only the K-largest parameters of its own locally accumulated gradient to the server, thereby reducing the number of coordinates communicated over the network. Examples of this family of methods include Top-K sparsification [27], Layer-wise Top-K [28], and Random-K [29].

In the biased compression literature, the objective is to identify sparsification rules and sparsity levels that reduce communication cost while preserving, or only slightly affecting, the convergence behavior obtained when full gradients are exchanged between client and server. Prior work in this direction has shown that gradient sparsification can remain effective even under high sparsification rates, and in some cases, sparsity levels as high as 99.9% have been reported with limited impact on model accuracy [27, 28, 43]. This property has made gradient sparsification particularly attractive for large and deep architectures, including AlexNet [44], ShuffleNetV2 [45], and ResNet [46]. However, these convergence properties were established in the context of communication-efficient training, not in the context of contribution-aware reward allocation.

In the next section, I discuss the main components of a FLCE algorithm and present the contribution estimation and data utility metrics most commonly used by state-of-the-art frameworks in related literature. This section completes the background knowledge required for the related work discussion in Chapter 3.

2.7 Federated Learning Contribution Estimation

FLCE algorithms provide the mechanism through which a federated learning system estimates each client's contribution and converts that estimate into rewards. These algorithms typically fall into two categories: **In-Training FLCE** and **Post-Training FLCE**.

1. **Post-training FLCE:** Algorithms used to estimate client contributions after the global model has been trained, making them well suited for reward mechanisms that distribute compensation after training, especially monetary rewards [15, 18, 19, 47, 48].

2. **In-training FLCE:** Algorithms used to estimate contributions during training, allowing contribution scores to influence the learning process itself. For this reason, they are commonly used in frameworks that distribute non-monetary rewards or detect Byzantine clients to prevent model poisoning. Since this thesis focuses

on contribution-aware incentivization rather than adversarial robustness, I mention Byzantine detection only as a related application and place the main emphasis on FLCE methods that support fair reward allocation [14, 22, 25, 40, 41, 49].

Considering the grand coalition of clients $\mathcal{N} = \{1, \ldots, N\}$, an FLCE algorithm aims to estimate how much each client contributes to the federated learning outcome. The output of this process is a contribution vector $\vec{\phi} = (\phi_1, \phi_2, \ldots, \phi_N)$, where ϕ_i denotes the estimated contribution of client i to the global model being trained. Following Chen *et al.* [38], I define an FLCE algorithm as the combination of two components: (i) a *data utility metric* and (ii) a *contribution estimation scheme*. I adopt their definitions to formalize these components as follows:

1. **Data Utility Metric**: Given a coalition $\mathcal{S} \subseteq \mathcal{N}$, a data utility metric $v(\cdot)$ is used to quantify the usefulness of $\mathcal{S}$ for a federated learning task. Let $2^{\mathcal{N}}$ denote the power set of $\mathcal{N}$. The utility of any coalition $\mathcal{S} \subseteq \mathcal{N}$ is then represented as $v(\mathcal{S})$, where $v(\cdot) : 2^{\mathcal{N}} \to \mathbb{R}$ and $v(\emptyset) = 0$. For example, if the utility of coalition $\mathcal{S}$ is defined in terms of the improvement or deterioration in validation accuracy, then $v(\mathcal{S})$ corresponds to the validation performance achieved by a model trained using the data contributed by clients in $\mathcal{S}$, measured relative to a baseline.

2. **Contribution Estimation Scheme**: Given a data utility metric $v(\cdot)$, a contribution estimation scheme defines a rule for assigning a contribution score ϕ_i to each client $i \in \mathcal{N}$ based on how the utility changes when client i is included in or excluded from different coalitions.

Additionally, Chen *et al.* [38] defines the high-level properties that an ideal FLCE algorithm should have:

1. **Effectiveness**: Estimated contributions should reflect each client's significance to the cooperative training outcome. In particular, contribution cannot be assessed solely by a client's individual data utility, since it does not capture the *marginal* value added when the client participates alongside other clients.

2. **Efficiency**: The method should estimate marginal contributions with sufficient fidelity, but without incurring prohibitive computational or communication overhead.

3. **Robustness**: The method should account for strategic or adversarial behavior (e.g., free-riding or poisoning attempts) and remain reliable under such conditions.

Achieving these three properties simultaneously is often a non-trivial task, and many FLCE algorithms involve trade-offs among effectiveness, efficiency, and robustness. In the following subsections, I first introduce the three contribution estimation schemes most commonly used in the literature, followed by the data utility metrics most frequently adopted for evaluating coalition utility. I acknowledge that alternative approaches exist for estimating per-client contribution values (ϕ_i) from a coalition utility function $v(\cdot)$, including learning-based methods that rely on previously trained machine learning models, regression models, or other ML-based estimators. However, these approaches fall outside the scope of this thesis.

2.7.1 Contribution Estimation Scheme: Individual

The *Individual* scheme estimates the contribution of client i solely by its standalone utility, defined as

$$\phi_i = v(\{i\}) \tag{8}$$

This scheme is computationally efficient because it avoids evaluating interactions among clients across different coalitions and permutations. However, it does not capture complementarity or redundancy between participants, and therefore may substantially misrepresent marginal contribution within a coalition [38].

2.7.2 Contribution Estimation Scheme: Leave-One-Out

The Leave-One-Out (LOO) is commonly utilized in machine learning for cross-validation, to estimate a client's contribution by measuring the decrease (or increase) in coalition utility when that client is removed from the coalition. Formally, the contribution of client i is defined as

$$\phi_i \;=\; v(\mathcal{S}) - v(\mathcal{S} \setminus \{i\}). \tag{9}$$

Intuitively, LOO treats client i as if it were the final addition to the grand coalition, since it compares the utility of the full coalition to the utility without i. As a contribution estimation scheme, LOO is substantially more effective than the *Individual* approach defined in 2.7.1, because it accounts for the client's contribution within a coalition rather than in isolation. At the same time, it is considerably more efficient than the Shapley Value and Least Core approaches, which require computing client marginal contributions across up to $2^{\mathcal{N}}$ coalitions, although it does not achieve the same level of effectiveness as those latter methods.

Depending on the fairness requirement, the LOO can raise fairness concerns, specially in settings where clients' data are redundant or strongly substitutable. For example, consider a coalition $\mathcal{S} \subseteq \mathcal{N}$ in which each client holds similar, high-utility data. Removing any single client $i \in \mathcal{S}$ may yield only a minor reduction in utility, causing LOO to assign small ϕ_i values to all clients in $\mathcal{S}$. Nevertheless, removing all clients in $\mathcal{S}$ simultaneously could lead to a substantial drop in utility, indicating that the group is collectively important even if each individual appears marginal under LOO [50, 51, 52].

2.7.3 Contribution Estimation Scheme: Shapley Value

The Shapley Value (SV) contribution scheme originates from cooperative game theory and was introduced by Lloyd Shapley [53]. Shapley values are widely regarded as a principled approach for estimating client contributions in FL because they quantify the expected marginal improvement in coalition utility attributable to a given client when the clients are added in a random order.

Formally, the Shapley value of client i, denoted by ϕ_i, is defined as

$$\phi_i \;=\; \frac{1}{N} \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \frac{1}{\binom{N-1}{|S|}} \Big[v(S \cup \{i\}) - v(S) \Big], \tag{10}$$

where $|\cdot|$ denotes set cardinality. The weighting term

$$\frac{1}{N} \cdot \frac{1}{\binom{N-1}{|S|}} \tag{11}$$

corresponds to the probability that, under a uniformly random permutation of $\mathcal{N}$, the set of clients appearing before client i is exactly S (equivalently, that client i appears in position $|S|+1$ and the preceding clients are precisely those in S).

Compared to the Individual and LOO schemes, the Shapley value is often preferred because it captures interactions among clients by averaging marginal contributions across all coalitions, which yields stronger fairness guarantees under standard axioms. Appendix B provides more details about the Shapley value's theoretical properties with respect to individual fairness.

Table 2.1 summarizes the main characteristics of each contribution scheme introduced above. The explanation and discussion of these properties are deferred to Appendix B.

Contribution Scheme	Individual	LOO	SV
Complexity	$\mathcal{O}(\lvert\mathcal{N}\rvert)$	$\mathcal{O}(\lvert\mathcal{N}\rvert)$	$\mathcal{O}(2^{\lvert\mathcal{N}\rvert})$
Null Player	✓	✓	✓
Symmetry	✓	✓	✓
Dummy Player	×	✓	✓
Additivity	×	×	✓
Efficiency	×	×	✓

Table 2.1: Contribution scheme characteristics comparison. In this table, $|\mathcal{N}|$ represents the number of clients. A detailed discussion of these properties can be found in Appendix B

2.7.4 Data Utility Metric: Data Quantity

A commonly used and conceptually simple data utility metric is *data quantity*, motivated by the intuition that larger datasets correspond to richer training data that can lead to more generalizable model parameter updates, thus higher contribution [54]. In federated learning, however, the server cannot directly observe clients' raw data. Instead, clients may report their number of local training samples to the server when sharing their local models, as in the original federated learning framework proposed by McMahan *et al.* [1]. Therefore, data quantity can be defined as

$$v(i) \triangleq \frac{|\mathcal{D}_i|}{|\mathcal{D}_{\mathcal{S}}|} \tag{12}$$

where $|\mathcal{D}_i|$ defines the number of training samples on client i training dataset, and $|\mathcal{D}_{\mathcal{S}}|$ defines the number of training samples in the coalition $\mathcal{S} \subseteq \mathcal{N}$.

However, in practical FL scenario, client data often follow non-Independent and Identically Distributed (non-IID) across participants. Therefore, dataset size alone is generally insufficient to characterize utility, thus making the "more samples implies more contribution" assumption inappropriate. That's because considering only the number of samples (i) ignores data quality (e.g., mislabeled or uninformative samples), (ii) ignores redundancy (e.g., duplicated samples can inflate reported size without adding learning signal), and (iii) disregards interaction effects, where a client's value depends on how its data complements other clients' data [38, 55].

In some works, the data-quantity utility notion is extended beyond raw sample counts to better capture the "amount" of useful information contributed by each client. For

example, in CFFL [23], when clients exhibit class-number imbalance (i.e., different clients contain different numbers of label classes), the server uses the number of classes available at client $i \in \mathcal{N}$ as an additional proxy for utility and employs it to weight the client's model update during aggregation. This data utility metric is defined as

$$v(i) \triangleq \frac{|\text{class}_i|}{\max |\text{class}_{\mathcal{S}}|} \tag{13}$$

where the $|\text{class}_i|$ denotes the number of distinct classes present in the training dataset of participant $i \in \mathcal{N}$, and $\max |\text{class}_{\mathcal{S}}|$ denotes the maximum number of classes present in $\mathcal{S} \subseteq \mathcal{N}$.

2.7.5 Data Utility Metric: Data Diversity

The intuition that larger datasets often provide richer training signal is partly rooted in the assumption that more samples imply greater *data diversity*. However, dataset size alone is an imperfect proxy for diversity, since clients may hold highly redundant samples. Motivated by this limitation, Xu *et al.* [55] formalize data diversity by modeling each sample as a point in a d-dimensional feature space. Under this view, diversity reflects how widely the samples are spread in the feature space. For a coalition $\mathcal{S}$, Xu *et al.* [55] define diversity as

$$v(\mathcal{S}) \triangleq \sqrt{\det(X_{\mathcal{S}}^T X_{\mathcal{S}})} \tag{14}$$

where $X_{\mathcal{S}} \in \mathbb{R}^{m \times d}$ is the feature matrix formed from the samples contributed by coalition $\mathcal{S}$, m is the number of records in the coalition dataset, and d is the number of features [55].

Similarly, Pan *et al.* [21] propose a diversity-based quantity proxy for graph-FL. They measured the graph diversity of client $i \in \mathcal{N}$ as the ratio between the number of unique motif categories present in the client's local graph data and the number of unique motif categories observed in the union of all clients' data. Since graph-based federated learning is outside the scope of this thesis, I refer the reader to [21] for further details.

Overall, *data quantity* and *data diversity* are self-reported statistics sent by clients and can be useful utility proxies in settings where clients are assumed to be honest and non-adversarial [1, 12, 21, 38, 55].

2.7.6 Data Utility Metric: Accuracy

Accuracy is a widely used data utility metric in FLCE for quantifying a client's marginal contribution to a coalition $\mathcal{S} \subseteq \mathcal{N}$ due to its interpretability in the context of machine learning. This data utility metric requires a validation dataset on which models can be evaluated in order to measure the model's utility. Therefore, when implemented in FLCE and Incentive-Aware FL, it assumes the availability of a validation dataset $\mathcal{D}_{\text{val}}$, typically maintained by the central server, which the clients agree to be used to evaluate their candidate models or gradients.

The data utility of a coalition $\mathcal{S}$ is defined as the predictive performance of the model $M_{\mathcal{S}}$, which is obtained when training using only the clients in $\mathcal{S}$ [18, 23, 38, 47]. It can be defined as

$$v(\mathcal{S}) \triangleq \text{Acc}(\vec{w}_{\mathcal{S}}; \mathcal{D}_{\text{val}}) \quad \text{where} \quad \text{Acc}(\vec{w}_{\mathcal{S}}; \mathcal{D}_{\text{val}}) \triangleq \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(x,y)\in\mathcal{D}_{\text{val}}} \mathbb{I}[y = \hat{y}_{\vec{w}_{\mathcal{S}}}(x)] \tag{15}$$

where $\hat{y}_{\vec{w}_{\mathcal{S}}}(x)$ denotes the prediction of the model $\vec{w}_{\mathcal{S}}$, obtained from training the model on the training data of coalition $\mathcal{S}$, on input x and $\mathbb{I}[\cdot]$ is the indicator function. In its original form, in order to calculate the utility of a given coalition $\mathcal{S}$, it is necessary to repeatedly retrain models across many coalitions (and permutations) $\mathcal{S}$, which quickly becomes computationally prohibitive outside controlled simulations. Song *et al.* [18], Lyu *et al.* [23], and Sun *et al.* [49] proposed methods to leverage this data utility metric without having to retrain models across many coalitions. Their approximations are further discussed in Section 3.5 of this thesis.

2.7.7 Data Utility Metric: Mutual Evaluation

The *mutual evaluation* data utility metric is a variant of the accuracy data utility metric defined in subsection 2.7.6. However, instead of relying on a single server-held reference dataset, clients use their own validation datasets to evaluate other clients' submitted models (or model updates) and report a score to the federation (e.g., accuracy value, accuracy loss, model rating). These peer-generated scores are then shared across the federation and are aggregated into a single value that's used to estimate the contribution of that participant [12, 22]. Mutual evaluation can be formally defined as

$$v(i) \triangleq \frac{1}{N-1} \sum_{j\in\mathcal{N}\setminus\{i\}} \text{Acc}(\vec{w}_i; \mathcal{D}_{\text{val},j}) \,, \tag{16}$$

$$\mathrm{Acc}(\vec{w}_i; \mathcal{D}_{\mathrm{val},j}) \triangleq \frac{1}{|\mathcal{D}_{\mathrm{val},j}|} \sum_{(x,y)\in\mathcal{D}_{\mathrm{val},j}} \mathbb{I}[y = \hat{y}_{\vec{w}_i}(x)]\,. \tag{17}$$

where $\hat{y}_{\vec{w}_i}(x)$ denotes the prediction produced by model $\vec{w}_i$ (submitted by client i) for input x, and $\mathbb{I}[\cdot]$ is the indicator function. Unlike the coalition-level utilities, mutual evaluation defines a *client-level* utility $v(i)$ by averaging the model's predictive performance across multiple evaluators' validation datasets.

Mutual evaluation is particularly common in decentralized and blockchain-based FL systems, where the absence of a trusted central coordinator motivates *distributed* verification contribution estimation. In contrast, mutual evaluation is less common in standard hub-and-spoke (centralized) FL deployments because it requires distributing one client's model/update to other clients for evaluation, which increases communication overhead and can introduce additional privacy and security considerations (e.g., exposing updates/models to peers). While such a mechanism can still be implemented in centralized settings via server-mediated relaying, it departs from the typical client-server interaction pattern assumed by most centralized FL frameworks.

2.7.8 Data Utility Metric: Cosine Similarity

The *cosine similarity* utility metric was introduced by Xu *et al.* [14] and has gained attention because it relies only on transmitted model update vectors, rather than on additional client-reported statistics. In its original formulation, this utility metric is computed using the client's L2-normalized model parameter update vector, formally denoted by $\vec{u}_{i,t}$ and defined in Equation 18.

$$\vec{u}_{i,t} = \frac{\Delta\vec{w}_{i,t}}{||\Delta\vec{w}_{i,t}||} \tag{18}$$

where $\Delta\vec{w}_{i,t}$ denotes the client's model parameter update vector and $||\cdot||$ denotes the L2 norm of a vector. Equation 19 defines the client model parameter update vector as

$$\Delta\vec{w}_{i,t} = \vec{w}_{i,t} - \vec{w}_{i,t-1} \tag{19}$$

In Xu *et al.* [14], the communication contract between clients and the server undergoes a minor modification. In the original federated learning framework proposed by McMahan *et al.* [1], clients and the server exchange model parameter vectors $\vec{w}_{i,t}$ and $\vec{w}_{\mathcal{N},t}$. In contrast, Xu *et al.* require each client to transmit the model parameter update vector $\Delta\vec{w}_{i,t}$. It is also possible to preserve the original communication protocol defined by McMahan *et al.* [1], but in this case, the server must reconstruct $\Delta\vec{w}_{i,t}$ by other means, such as by storing

a copy of each client's previous model state. This requirement may become a bottleneck as the number of clients or the model size increases. Once $\vec{u}_{i,t}$ is obtained, the data utility metric is computed as the cosine similarity between the client's normalized update vector $\vec{u}_{i,t}$ and a reference vector $\vec{u}_{\mathcal{S},t}$, as shown in Equation 20.

$$v(i) \triangleq \frac{\langle \vec{u}_{i,t}, \vec{u}_{\mathcal{S},t} \rangle}{||\vec{u}_{i,t}|| \cdot ||\vec{u}_{\mathcal{S},t}||} \tag{20}$$

This utility metric can be combined with any previously introduced contribution estimation schemes. However, Xu *et al.* [14] derive, in their Theorem 1, an error bound showing that the Cosine Gradient Shapley Value can be approximated by directly computing the cosine similarity between $\vec{u}_{i,t}$ and $\vec{u}_{\mathcal{N},t}$, where $\vec{u}_{\mathcal{N},t}$ denotes the vector obtained by aggregating all client normalized update vectors $\vec{u}_{i,t}$. This approximation is discussed in more details in Section 3.3.3, when I discuss the frameworks selected for the research.

2.8 Rank Correlation Coefficients

The purpose of this section is to provide the statistical background required for the fairness analysis conducted later in this thesis. I do not use this section to establish which rank-based coefficient is superior, instead, I introduce Spearman's ρ and Kendall's τ as rank-based association metrics that are also aligned with contribution-based fairness analysis. In Chapter 4, I use the conducted experiments and the definitions presented here to establish what I understand to be a better correlation coefficient to measure fairness.

As discussed in Section 2.5, several federated learning frameworks [7, 14, 21, 22, 23, 25, 30, 41] evaluate contribution fairness per Definition 3, that is, computing the Pearson's r Correlation Coefficient between the final test accuracy achieved by each client in standalone training and the final test accuracy achieved by the same client after federated collaboration.

The Pearson's r Correlation Coefficient is a convenient metric because it reduces the relationship between standalone and federated client performance to a single scalar value that captures the linear association between the two sets of accuracies. As discussed in Section 2.5, since Pearson's r Correlation Coefficient is computed by calculating the client accuracies values deviation from their respective means, my interpretation is that its use in this context as also being used to capture the model differentiation across clients. However, there is a potential implementation problem that was not explicitly considered when this

metric was adopted for contribution fairness evaluation: Pearson's r Correlation Coefficient is known to be sensitive to outliers [56, 57, 58].

Spearman's ρ Rank Correlation Coefficient [59] and Kendall's τ Fank Correlation Coefficient [60, 61] address this limitation by measuring association through ranks rather than raw numerical magnitudes. As a result, they are less sensitive to extreme observations and are more appropriate when the central question is whether the relative ordering of observations is preserved. Despite that, the related work reviewed in this thesis does not use Spearman's ρ or Kendall's τ to evaluate fairness in FAFL or IAFL frameworks.

However, because Spearman's ρ and Kendall's τ uses ranks rather than raw numerical values, they discard magnitude information. This makes them appropriate for evaluating whether the contribution ordering is preserved, but insufficient for measuring the numerical dispersion of client model performances. To capture this complementary aspect, I use the Gini Mean Difference, defined in Section 2.9, to quantify the pairwise performance differentiation between clients' final test accuracies.

2.8.1 Spearman's Rank Correlation Coefficient

Spearman's ρ Rank Correlation Coefficient [59], commonly denoted by the greek letter ρ, is a non-parametric measure of monotonic association between two variables. Unlike Pearson's r, which is computed using the original numerical values, Spearman's ρ is computed after replacing each variable by its rank. Therefore, Spearman's ρ evaluates whether larger values of one variable tend to correspond to larger values of the other variable, without requiring the relationship between them to be linear.

Let $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a set of n paired observations. Spearman's coefficient is computed by replacing x_i and y_i with their ranks $R(x_i)$ and $R(y_i)$, and then applying Pearson's correlation coefficient to the rank-transformed variables:

$$\rho = \frac{\sum_{i=1}^{n} \left(R(x_i) - \overline{R_X}\right)\left(R(y_i) - \overline{R_Y}\right)}{\sqrt{\sum_{i=1}^{n} \left(R(x_i) - \overline{R_X}\right)^2} \sqrt{\sum_{i=1}^{n} \left(R(y_i) - \overline{R_Y}\right)^2}} \tag{21}$$

where $R(x_i)$ and $R(y_i)$ denote the ranks of x_i and y_i, respectively, and $\overline{R_X}$ and $\overline{R_Y}$ denote the corresponding mean ranks.

Spearman's ρ takes on values in the interval $[-1, 1]$, where a value of $\rho = 1$ indicates a perfect increasing monotonic relationship, meaning that both variables induce the same ordering of observations, and a value of $\rho = -1$ indicates a perfect decreasing monotonic

relationship, meaning that one variable induces the exact reverse ordering of the other. A value close to zero indicates weak monotonic association, although it does not imply statistical independence.

2.8.2 Kendall's Rank Correlation Coefficient

Kendall's τ Rank Correlation Coefficient [60, 61], commonly denoted by Kendall's τ, is a non-parametric measure of ordinal association between two variables. Like Spearman's coefficient, Kendall's τ evaluates agreement between orderings rather than linear association between raw numerical values. However, Kendall's coefficient does this through a direct pairwise comparison of observations, which makes it an appropriate metric to measure the fairness criterion considered in this thesis.

Let

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \tag{22}$$

be a set of n paired observations of two variables X and Y. For any two distinct observations (x_i, y_i) and (x_j, y_j), with $i < j$, the pair is said to be concordant if both variables order the two observations in the same direction. Formally, the pair is concordant if

$$(x_i - x_j)(y_i - y_j) > 0. \tag{23}$$

This occurs when $x_i > x_j$ and $y_i > y_j$, or when $x_i < x_j$ and $y_i < y_j$.

The pair is discordant if the ordering is reversed between the two variables:

$$(x_i - x_j)(y_i - y_j) < 0. \tag{24}$$

This occurs when $x_i > x_j$ and $y_i < y_j$, or when $x_i < x_j$ and $y_i > y_j$. Pairs for which $x_i = x_j$, $y_i = y_j$, or both, are tied pairs.

The original Kendall coefficient, denoted by τ_a, proposed in [60] is computed as

$$\tau_a = \frac{C - D}{\binom{n}{2}} = \frac{2(C - D)}{n(n-1)}. \tag{25}$$

where C denotes the number of concordant pairs, D denotes the number of discordant pairs, and $\binom{n}{2}$ denotes the total pairs of observations.

The coefficient τ_a takes on values in the interval $[-1, 1]$. A value of 1 indicates perfect agreement between the two rankings, meaning that every pair of observations is ordered

consistently in both variables. A value of -1 indicates perfect disagreement, meaning that every pair is ordered in the opposite direction. A value close to 0 indicates weak ordinal association.

Nevertheless, τ_a does not adjust its computation when there are ties, because ties are not counted as concordant nor discordant, but they still remain the denominator. This limitation was addresses in Kendall's τ_b, proposed in [61], which adjusts the normalization term to account for ties in either variable. Let n_0 denote the total number of observation pairs:

$$n_0 = \binom{n}{2} = \frac{n(n-1)}{2}. \tag{26}$$

Let n_X denote the number of pairs tied with respect to variable X, and let n_Y denote the number of pairs tied with respect to variable Y. These quantities can be computed as

$$n_X = \sum_{g=1}^{G_X} \frac{t_g(t_g-1)}{2}, \tag{27}$$

and

$$n_Y = \sum_{h=1}^{G_Y} \frac{u_h(u_h-1)}{2}, \tag{28}$$

where t_g is the number of observations in the g-th tie group of variable X, and u_h is the number of observations in the h-th tie group of variable Y. A tie group refers to a set of observations that share the same value in a given variable. For instance, if three data points obtain the same value for a given variable, these points form one tie group of size $u_h = 3$. This tie group contributes $\binom{3}{2} = 3$ tied pairs to either n_X or n_Y, depending on whether the tie occurs in variable X or variable Y. The indices g and h simply enumerate the tie groups observed in X and Y, respectively.

Kendall's τ_b is then defined as

$$\tau_b = \frac{C - D}{\sqrt{(n_0 - n_X)(n_0 - n_Y)}}. \tag{29}$$

Equivalently, Kendall's τ_b measures the excess of concordant pairs over discordant pairs, normalized by the number of comparable pairs after accounting for ties. Like τ_a, Kendall's τ_b takes values in the interval $[-1, 1]$, where 1 indicates perfect rank agreement, -1 indicates perfect rank disagreement, and values close to 0 indicate weak ordinal association.

2.9 Gini Mean Difference

The Gini Mean Difference (GMD) [62] measures the average pairwise difference between observations. In this thesis, I use it to quantify the average pairwise difference between clients' final test accuracies, thereby measuring the degree of performance differentiation across client models. Given a set of clients $\mathcal{N}$ and the final test accuracy x_i achieved by each client $i \in \mathcal{N}$, the Gini Mean Difference is defined as

$$\text{GMD} = \frac{2}{|\mathcal{N}|(|\mathcal{N}|-1)} \sum_{i \in \mathcal{N}} \sum_{\substack{j \in \mathcal{N} \\ j<i}} |x_i - x_j|. \tag{30}$$

A GMD value close to zero indicates that clients converge to nearly identical model performances, whereas larger values indicate stronger differentiation among client models. However, in the context of contribution fairness, a larger GMD cannot determine fairness on its own, because model differentiation is only desirable when the differentiated performances remain aligned with the clients' contribution-dependent ordering.

For this reason, GMD must be interpreted together with a rank-based correlation metric. A high positive rank-based correlation combined with a high GMD indicates contribution-aligned differentiation: the expected client ordering is preserved, and the resulting client models are meaningfully separated. In contrast, a high GMD combined with a strong negative rank-based correlation indicates harmful differentiation, because the framework separates client performances while reversing the expected contribution-based ordering.

In this thesis, I interpret a GMD below 1 percentage point as negligible, because such differences indicate that the final client models are practically similar in performance. This value should be understood as a study-specific practical threshold.

Chapter 3

Related Work

In this chapter, I analyze state-of-the-art federated learning frameworks that incorporate contribution fairness as one of their main design objectives. I first summarize their key characteristics, including their contribution estimation strategies, incentive mechanisms, reward distribution approach, and reward method. I then separate the discussion between frameworks that use monetary rewards for client incentivization and frameworks that distribute non-monetary rewards through personalized model- or gradient-based mechanisms. Each subsection is titled according to the corresponding article or framework and summarizes its contribution estimation method, incentive mechanism, and approach to implementing contribution fairness. Finally, I examine in greater detail the two frameworks selected for experimental comparison in this thesis.

3.1 Frameworks Summary

Table 3.1 summarizes the main design characteristics of state-of-the-art federated learning frameworks that treat contribution fairness as a core objective. The table highlights the design space of contribution-aware incentivization along four axes: *when* contribution is estimated, given by the stage at which the FLCE is executed (in-training, post-training, or externally supplied); *whether* the resulting incentive is monetary or non-monetary; *when* the corresponding reward is distributed to clients; and *how* that reward is delivered, captured by the reward method.

The majority of the surveyed frameworks fall within the non-monetary reward FL frameworks type, which is the focus of this work. Instead of compensating clients with

currency, these methods reward contribution through preferential access to the model itself, most commonly via Top-K Sparsification, where the number of gradient components a client receives is scaled to its assessed contribution. This framework type quantifies fairness through the Pearson's r Correlation Coefficient of Definition 3. It is precisely the combination of non-monetary, in-training rewards delivered through Top-K Sparsification and evaluated with a Pearson-based fairness criterion that is most relevant to the methods developed in the remainder of this work.

Framework	**FLCE execution**	**Incentive**	**Reward Distribution**	**Reward Method**
Song *et al.* [18]	Post-Training	Monetary	Post-Training	Currency
OpenFL [12]	In-Training	Monetary	In-Training	Crypto Currency
FLI [13]	External	Monetary	In-Training	Currency
CoAst [15]	Post-Training	Monetary	Post-Training	Currency
CFFL [23]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
FPPDL [22]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
ACGSV [14]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
Pan *et al.*[21]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
FedDAR [25]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
FedFAIM [24]*	In-Training	Non-Monetary	In-Training	Top-K Sparsification
Huang *et al.* [41]*	In-Training	Non-Monetary	In-Training	Gradient-Reconstruction
Wu *et al.*[7]*	External	Non-Monetary	In-Training	Gradient-Selection
ShapleyFL[49]	In-Training	Non-Monetary	In-Training	Client-Sampling

Table 3.1: Summary of contribution estimation and incentive designs in state-of-the-art federated learning frameworks. Frameworks marked with an asterisk (*) adopt the Pearson Correlation Coefficient in Definition 3 to evaluate contribution fairness. Frameworks with "External" FLCE execution assume that the contribution estimate is supplied as an input to the system.

3.2 Monetary Rewards

In this section, I discuss frameworks in which monetary compensation is the primary mechanism for incentivizing client participation. These frameworks consider client incentivization as a financial reward allocation problem, so the estimated contribution is used

as a proxy to determine how a budget (either pre existing or collected through a specific framework design) should be allocated to each client.

3.2.1 Profit Allocation for Federated Learning

Song *et al.* [18] proposed an FLCE algorithm that evaluates client contribution using accuracy as the data utility metric (Subsection 2.7.6), and the Shapley Value as the contribution scheme (Subsection 2.7.3) to estimate per-client contribution. In principle, computing exact per-client Shapley Values requires retraining and evaluating models for many coalitions and permutations $\mathcal{S} \subseteq \mathcal{N}$, which is computationally prohibitive in a federated setting. To avoid repeated training, the framework follows the standard FL protocol in which, after local training, clients send their locally updated models $\vec{w}_{i,t}$ to the server. Because all selected clients start each round from the same global model, the server can derive each client's corresponding accumulated gradient through Equation 31.

$$\Delta\vec{w}_{i,t} = \vec{w}_{i,t} - \vec{w}_{i,t-1} \tag{31}$$

Then, for any coalition $\mathcal{S} \subseteq \mathcal{N}$, the server aggregates the accumulated gradients into a coalition gradient $\Delta\vec{w}_{\mathcal{S},t}$. By applying $\Delta\vec{w}_{\mathcal{S},t}$ to a reference model (i.e. the global model of the previous round $\vec{w}_{0,t-1}$), the server approximately reconstructs a coalition-specific model $\tilde{w}_{\mathcal{S},t}$, which is evaluated on a shared validation dataset $\mathcal{D}_{\text{val}}$ to compute coalition utilities. Using these utilities, Song *et al.* compute each client's Shapley-style contribution (ϕ_i), termed the *Contribution Index (CI)*, as

$$\phi_i = C \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}} \frac{v(\mathcal{S} \cup \{i\}) - v(\mathcal{S})}{\binom{N-1}{|\mathcal{S}|}} \tag{32}$$

where C is a constant factor. Song *et al.* [18] propose two algorithms for estimating the contribution of each client to the model. The OR algorithm is intended for post-training contribution estimation because it computes a Shapley value ϕ_i value for every client $i \in \mathcal{N}$ after the training has been completed, and the MR algorithm, is designed for in-training contribution estimation because the server computes the Shapley vale ϕ_i at every round of training. Although the framework proposed by Song *et al.* [18] does not explicitly implement in-training or post-training monetary reward, its main claim is to use the per-client contribution index ϕ_i as the basis for deriving a monetary incentive and promote contribution fairness.

3.2.2 OpenFL: A scalable and secure decentralized FL system on the Ethereum blockchain

Wahrstätter *et al.* [12] propose OpenFL, a blockchain-based federated learning framework that integrates contribution estimation through a collateral-backed reputation mechanism. OpenFL operates under a Proof-of-Stake consensus protocol, where clients must lock collateral before participating in training. After each communication round, the framework estimates contribution using the mutual-evaluation utility metric described in Section 2.7.7. In this procedure, clients exchange models off-chain, evaluate each other's models on their local test datasets, and submit discrete ratings on-chain, such as -1 for bad, 0 for neutral, and 1 for good. These peer evaluations are aggregated into a reputation score for each participant, and models that receive sufficiently positive evaluations are included in the FedAvg aggregation step shown in Algorithm 1.

OpenFL also makes participation in the mutual-evaluation phase a protocol requirement. Clients that fail to submit evaluations, or whose models receive consistently poor peer ratings, are penalized through reputation reduction and collateral slashing, and may ultimately be excluded from the federation. Therefore, OpenFL seeks to implement contribution fairness by linking client reputation, aggregation eligibility, and monetary incentivization and penalization.

3.2.3 A fairness-aware Incentive Scheme for Federated Learning

Yu *et al.* [13] propose FLI, an incentive-aware framework for settings in which a limited monetary budget must be distributed across clients to encourage participation. In this scenario, the available budget is insufficient to fully compensate all clients in proportion to their contributions, so unpaid rewards accumulate over time as *regret*. At every round of participation, clients must also report their training-related costs, which is taken in account by the framework when computing *regret*. FLI assumes that the clients contribution value is provided *as an input to the mechanism* and focuses on the training-related cost that is reported by the clients are used for regret computation only. Therefore, it abstracts away the computation required to obtain the utility values and focuses instead on the incentive layer, and how the generated revenue is allocated over time to reduce regret and retain participants in the federation under budget constraints.

3.2.4 CoAst: Validation-Free Contribution Assessment for Federated Learning based on Cross-Round Valuation

Wu *et al.* [15] propose CoAst, a validation-free federated learning contribution assessment framework that estimates client contribution without requiring a shared validation dataset. CoAst introduces a cross-round valuation strategy that compares a client's local model at round t with the direction of the global parameter updates accumulated over several subsequent rounds. This cross-round design reduces the effect of single-round stochastic noise and allows the estimator to capture whether a local update remains useful for the later global optimization trajectory.

To further emphasize the most relevant parameters, CoAst applies a parameter pruning procedure inspired by ternary weight quantization. This procedure removes small parameter updates and keeps only the sign of the most important updates, so that contribution assessment focuses on update direction rather than raw magnitude. CoAst then computes the round-level contribution of each client using a signed cosine-similarity-style metric (Subsection 2.7.8), which measures whether the client's local model direction agrees with the future global update direction. The final contribution score is obtained by summing these round-level contribution estimates across the training process.

Although CoAst is motivated by potential applications in fair reward allocation and incentive design, the framework itself performs contribution evaluation only, which can be further used to derive a post-training monetary rewards.

3.3 Non-Monetary Rewards

Along this section I discuss frameworks in which non-monetary rewards are adopted as the mechanism for incentivizing client participation. These frameworks use contribution estimation to determine how the aggregated gradient should be modified to induce differences in model performance and achieve contribution fairness based on Definition 1, measured as in Definition 3.

In this section, I discuss frameworks that adopt non-monetary rewards as the mechanism for incentivizing client participation. These frameworks use contribution estimation to determine how the aggregated gradient should be transformed into client-specific rewards, with the goal of inducing differences in client model performance. Under this design, higher-contributing clients are expected to receive more informative rewards than lower-contributing clients, promoting contribution fairness as defined in Definition 1 and evaluated according to Definition 3.

3.3.1 Collaborative Fairness in Federated Learning

Lyu *et al.* [23] propose CFFL (Collaborative Fair Federated Learning), a framework that estimates client contribution by reconstructing each client model at the server and evaluating its validation accuracy on a publicly available dataset.

CFFL starts from a common global model broadcast to all clients and adopts a gradient-based communication workflow instead of exchanging full model weights. In each round, client i sends its locally accumulated gradient $\Delta\vec{w}_{i,t}$ to the server, and since all clients start from the same initial model and the server receives their accumulated gradients, the server can track and reconstruct each client's model state for validation-based contribution estimation.

The estimated contribution scores are then used to generate client-specific non-monetary rewards from the aggregated model update $\Delta\vec{w}_{\mathcal{N},t}$. These rewards are produced through the Top-K sparsification procedure described in Algorithm 2, yielding a client-specific reward vector $\vec{\nu}_{i,t}$. The server sends to each client its respective reward, and statistics about the sample and class distribution across the federation, which by the framework's design and assumption, clients can use this statistics to apply distinct model update rules. For the data-quantity-skewed scenario, each client updates its local model as shown in Equation 33:

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{D}_i}{\max_{j\in\mathcal{N}} \mathcal{D}_j} \cdot \Delta\vec{w}_{i,t} \tag{33}$$

where $\mathcal{D}_i$ represents the number of samples in client's $i \in \mathcal{N}$ local dataset, while in the class-label-skewed scenario, each client updates its local model using as shown in Equation 34:

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{C}_i}{\max_{j\in\mathcal{N}} \mathcal{C}_j} \cdot \Delta\vec{w}_{i,t} \tag{34}$$

where $\mathcal{C}_i$ represents the number of distinct classes in client's $i \in \mathcal{N}$ local dataset.

The server updates its reconstructed version of the client model, denoted by $\tilde{\vec{w}}_{i,t}$ to track each client's model state across communication rounds.

CFFL is one of the frameworks selected for experimental comparison in this thesis. I discuss its full workflow and implementation details in Section 3.6.

3.3.2 Towards Fair and Privacy-Preserving Federated Deep Models

Lyu *et al.* [22] propose FPPDL (Fair and Privacy-Preserving Deep Learning), a decentralized blockchain-based collaborative learning framework that pursues collaborative fairness

by assigning participants different final model versions according to their relative contributions. FPPDL assesses client contribution based on a local credibility mutual-evaluation mechanism, as described in Subsection 2.7.7, together with a point-based exchange protocol and a three-layer encryption scheme for enhanced privacy.

Before collaborative training begins, each participant trains both a standalone model and a DPGAN model using its private local dataset. The DPGAN is used to generate differentially private synthetic samples that can be shared with other participants without directly disclosing the original data. During the initial phase, participant i sends a subset of its generated samples to the other participants, who label these samples using their own standalone models and return the predicted labels. Participant i then compares these returned labels with the majority-vote labels and uses the agreement rate to initialize a local credibility score for each peer. This phase also initializes each participant's sharing level and point balance, which later constrain how many gradients it can request or provide during collaborative training.

During collaborative training, FPPDL replaces server-side aggregation with a decentralized earn-and-pay exchange of gradient information. Each participant spends points to request gradients from peers that are credible from its own local perspective, while participants that provide useful information earn additional points. When participant j responds to a request from participant i, it sends a requester-specific Top-K sparsified $\vec{\nu}_{i,t}$ of its local accumulated gradient, in which K depends on the requester's available points, participant j's credibility from participant i's perspective, and participant j's sharing level. Therefore, participants do not receive the same update information; instead, they integrate different subsets of peer gradients into their own local models.

Participants that generate useful synthetic samples, provide informative gradients, and comply with the protocol are expected to receive more useful gradient information and converge to stronger local models. Conversely, low-contribution participants receive fewer useful updates and may be excluded from the credible participant set if their credibility falls below the agreed threshold. In this sense, FPPDL implements a non-monetary reward mechanism in which the reward is not an external payment, but differentiated access to model-improving gradient information.

In FPPDL, each client updates its model as shown in Equation 35

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} \tag{35}$$

where $\Delta\vec{w}_{i,t}$ denotes client $i \in \mathcal{N}$ locally accumulated gradients, and $\vec{\nu}_{i,t}$ is the Top-K sparsified updates received by client $i \in \mathcal{N}$.

3.3.3 Gradient-driven Rewards to Guarantee Fairness in Collaborative Machine Learning

Xu *et al.* [14] propose the ACGSV[1] framework, which estimates client contribution using the cosine similarity data utility metric defined in Subsection 2.7.8.

Similar to CFFL, ACGSV adopts a gradient-based communication workflow in which clients send their accumulated gradients to the server, and the server uses these gradients to estimate contribution and generate client-specific rewards using the Top-K sparsification technique discussed in Section 2.6. Before computing contribution scores, the server applies L2-normalization to the accumulated gradients. This normalization prevents gradient magnitude from dominating the contribution estimation process and makes the estimator focus primarily on the directional alignment between client gradients and the aggregated gradient.

ACGSV pursues contribution fairness by sending each client a distinct reward vector derived from the aggregated model update vector $\Delta\vec{w}_{\mathcal{N},t}$. This reward is generated through the Top-K sparsification procedure described in Algorithm 2, producing a client-specific reward vector $\vec{\nu}_{i,t}$. However, unlike CFFL, ACGSV does not update the client model using both the locally accumulated update and the received reward. Instead, each client updates its local model using only the reward vector distributed by the server, as shown in Equation 36:

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \vec{\nu}_{i,t}. \tag{36}$$

ACGSV is the second framework selected for experimental comparison in this thesis. I discuss its full workflow and implementation details in Section 3.5.

3.3.4 Towards Fair Graph Federated Learning via Incentive Mechanisms

Pan *et al.* [21] propose an FLCE framework tailored to graph-based federated learning. Similar to Xu *et al.*[14], Pan *et al.*[21] estimate client contribution estimation use the cosine similarity between a client's model update and the aggregated update and by incorporating a *motif-prototype* utility component that quantifies structural diversity in the client's local

[1]Existing literature that compares its work with Xu *et al.* [14] often refers to the framework as CGSV. In this thesis, I avoid this notation because the contribution estimator does not compute the exact Cosine Gradient Shapley Value. I therefore use ACGSV to emphasize its approximate nature.

graph data by identifying the unique motif categories present in the dataset. This motif-based signal can be interpreted as a graph dataset analogue of data diversity/quantity metrics. At each round, clients whose updates exhibit higher cosine similarity and whose local data contain richer motif prototypes receive higher contribution scores. In particular, the importance coefficient $r_{i,t}$ is updated as

$$r_{i,t} = \alpha_1 \cdot r_{i,t-1} + (1 - \alpha_1) \cdot (\psi_{i,t} + \alpha_2 \cdot d_i) \tag{37}$$

where d_i measures the graph diversity of client i, computed as $d_i = \frac{k_i}{K}$, with k_i denoting the number of unique motifs in the client's local graph data and K denoting the total number of unique motif categories present across all participating clients. The hyperparameters α_1 and α_2 control, respectively, the moving-average update rate and the weight assigned to the graph-diversity term. As a result, the client importance coefficient remains structurally coupled to the diversity of its local data. Their reward design and client model update design is identical to the one implemented by Xu *et al.* [14] with ACGSV, which is shown in Equation 36. Pan *et al.* [21] also apply Top-K sparsification. While their formulation is not directly transferable to conventional FL because it is specially tailored for Graph Federated Learning which has a different data modality, the same underlying principle remains.

3.3.5 FedDAR: Multidimensional Contribution Assessment and Adaptive Gradient Rewards

Lei *et al.* [25] propose FedDAR, a collaborative fairness framework designed for heterogeneous federated learning. Unlike CFFL and ACGSV, FedDAR does not estimate client contribution from a single signal, such as validation accuracy or cosine similarity. Instead, it uses a Multifactor Evaluation Score (MES), which combines three client-level factors: historical reputation, local data size, and class label diversity. The MES of client i at round t is computed as

$$S_{i,t} = w_{\text{rep}} \frac{r_{i,t-1}}{\max_j r_{j,t-1}} + w_{\text{data}} \frac{n_i}{\max_j n_j} + w_{\text{div}} \frac{d_i}{\max_j d_j}, \tag{38}$$

where $r_{i,t-1}$ denotes the reputation of client i from the previous round, n_i denotes its local data size, and d_i denotes its class label diversity in the clients local dataset. The coefficients w_{rep}, w_{data}, and w_{div} control the relative importance assigned to reputation, data volume, and class diversity, respectively.

Similar to CFFL and ACGSV, FedDAR adopts a gradient-based communication workflow. Clients send their accumulated gradients to the server, and the server returns client-specific rewards generated through the Top-K sparsification procedure described in Algorithm 2. FedDAR uses a contribution-dependent sparsification rule to determine how many coordinates of the aggregated gradient are preserved for each client. Higher-scoring clients receive denser reward vectors, while lower-scoring clients receive sparser ones. As in ACGSV, each client updates its local model using only the reward vector received from the server, following the update rule shown in Equation 36.

3.3.6 FedFAIM: A Model Performance-Based Fair Incentive Mechanism for Federated Learning

Shi *et al.* [24] propose FedFAIM framework, which estimates clients contribution by combining client marginal utility on the validation dataset and gradient alignment through the cosine similarity approach proposed by Xu *et al.* [14].

Similarly to CFFL and ACGSV, FedAIM adopts a gradient-based communication workflow in which clients send their accumulated gradients to the server, and the server uses these gradients to estimate contribution and generate client-specific rewards using Top-K sparsification technique. Before any contribution is estimated, the server performs a quality-detection step. For each client $i \in \mathcal{N}$ at round t, the server constructs two provisional global models: one aggregating all clients and one aggregating every client except i. The server then computes the marginal loss $\delta_{i,t} = \ell_{-i,t} - \ell_t$, i.e., the increase in validation-set loss caused by removing client i from the aggregation. A client is admitted only if $\delta_{i,t} \geq \delta$ for a loss threshold δ, and the passing gradients are combined with quality-aware weights $r_{i,t}$ obtained from a softmax over the marginal losses rather than from dataset sizes as in FedAvg [1].

For the passing clients, FedAIM estimates the per-round contribution by calculating the client's local gradient projection onto the aggregated gradient of the passing coalition $\mathcal{S}$, as show in Equation 39

$$\phi_{i,t} = r_{i,t} \cdot \|\vec{u}_{i,t}\| \cos\big(\vec{u}_{i,t}, \vec{u}_{\mathcal{S},t}\big) \tag{39}$$

where $\phi_{i,t}$ is client $i \in \mathcal{S}$ estimated contribution, $r_{i,t}$ is the weight used during the aggregation phase, $\vec{u}_{i,t}$ is the L2-normalized gradient $\vec{w}_{i,t}$ sent by client $i \in \mathcal{S}$, and the $\vec{u}_{\mathcal{S},t}$ is the gradient obtained from the aggregation of the $\vec{u}_{i,t} \in \mathcal{S}$, and $\mathcal{S}$ denotes the coalition of the gradients that passed by the initial validation phase.

These per-round contributions are accumulated into a reputation score $c_{i,t}$, as shown in Equation 40

$$c_{i,t} = \max\big(0, \sum_{\tau=1}^{t} \phi_{i,\tau}\big) \tag{40}$$

After computing the reputation score $c_{i,t}$, these values are sum-normalized as $z_{i,t} = c_{i,t} / \max_i c_{i,t}$, and passed through a Gompertz function $q_{i,t}$ that acts as a quality-detection history function, whose input grows with the number of passed detections n_i^{pass} and shrinks with the number of failed ones n_i^{fail}. The importance coefficient is the product of these two terms,

$$r_{i,t} = q_{i,t} \cdot z_{i,t} \tag{41}$$

The importance coefficient $r_{i,t}$ is used to compute the K_i, which denote the number of coordinates returned to client i through the Top-K sparsification of the global gradient $\vec{u}_{\mathcal{N},t}$, which is formally defined in Equation 42

$$K_{i,t} = \frac{r_{i,t}}{\max_i r_{i,t}} |\vec{u}_{\mathcal{N},t}| \tag{42}$$

where $|\vec{u}_{\mathcal{N},t}|$ is the dimensionality of the aggregated gradient, so the highest-reputation client recovers the entire aggregated model and the others receive proportionally sparser versions. The Top-K sparsified gradients, denote as $\vec{\nu}_{i,t}$ are sent to the clients, which update their local models as in the ACGSV framework, formally defined in Equation 36.

3.3.7 Privacy-Preserving and Hierarchically Fair FL in IoT Based on Compressed Sensing

Huang *et al.* [41] propose a privacy-preserving oriented federated learning framework. Similar to ACGSV, the framework estimates client contribution using a gradient-alignment metric based on cosine similarity, as defined in Subsection 2.7.8. However, unlike CFFL and ACGSV, this framework places stronger emphasis on secure gradient transmission. For this reason, clients do not upload dense accumulated gradients directly to the server. Instead, each client first computes its local gradient and applies Random Gradient Selection (RGS), which randomly preserves only a subset of gradient coordinates and removes the remaining ones. This step can be interpreted as a Random-K-style sparsification mechanism, since the retained coordinates are selected randomly rather than according to their magnitude, although due to some distinct nuances it is not strictly the same mechanism. Nevertheless, the resulting sparse gradient is then encrypted using the compressed-sensing-based encryption scheme and uploaded to the aggregation server.

During aggregation, the server combines the encrypted sparse gradients without directly observing their plaintext values. The encrypted representations are then used to recover both the sparse local gradients and the aggregated sparse global gradient. Importantly, this reconstruction does not recover the original dense client gradients before RGS; it only recovers the sparse gradients after random coordinate selection. These sparse gradients are then used to compute the contribution by measuring the alignment of the client's sparse gradient and the aggregated gradient.

The framework pursues contribution fairness through hierarchical allocation. Instead of sending the same global update to all participants, the grading server assigns each participant a reconstruction level according to its contribution score. Higher-contribution participants receive a higher-quality reconstruction of the aggregated sparse global gradient, while lower-contribution participants receive a lower-quality reconstruction. The hierarchical reward mechanism is analogous to Top-K sparsified reward allocation in the sense that both mechanisms control the quality of the gradient information returned to each participant. In Top-K-based methods, the server directly controls the number of aggregated-gradient coordinates sent to each client. In Huang *et al.* [41], the grading server instead controls the reconstruction quality of the aggregated sparse global gradient through participant-specific level-determination matrices. A higher-contribution participant is assigned a larger reconstruction level, allowing it to recover a more accurate version of the aggregated gradient, while a lower-contribution participant receives a lower reconstruction level and therefore a less accurate model update.

Each participant then updates its local model using the reconstructed global gradient assigned to it:

$$\vec{w}_{i,t} = \vec{w}_{i,t} - \eta_g \vec{g}_{i,t}. \tag{43}$$

where $\vec{g}_{i,t}$ denotes the reward allocated to client i at round t, and η_g is a learning rate hyperparameter that is used to control the step-size of the received reward. This equation follows the same structure as the local model update rule in Equation 36, since the client updates its model using the reward distributed by the server rather than a full shared global update.

3.3.8 Incentive-Aware Federated Learning with Training-Time Model Rewards

Wu *et al.* [7] propose a framework focused on non-monetary training-time model rewards rather than on designing a new FLCE algorithm, and it assumes that each client has a per-round estimated contribution value $p_{i,t}$ that is an input to the framework.

Unlike previous frameworks discussed in this section, which commonly rely on Top-K sparsification to differentiate client rewards at the parameter level, Wu *et al.* [7] differentiate rewards at the client-update level. For each client i, IAFL computes a reward rate $\vec{\nu}_{i,t}$ from the client's estimated contribution value. This reward rate determines how many accumulated gradients $\Delta\vec{w}_{i,t}$ from other clients can be included in client i's personalized model update. Specifically, the server samples a subset $S'_{i,t}$ from the clients other than i, then forms $S_{i,t} = \{i\} \cup S'_{i,t}$, ensuring that client i's own update is always included, which is mathematically equivalent to Equation 35. The reward sent to client i is then obtained by averaging the full local model updates from the selected set $S_{i,t}$. Therefore, higher-contribution clients receive model updates aggregated from a larger proportion of the federation, while lower-contribution clients receive updates aggregated from fewer clients.

Although Wu *et al.* [7] design IAFL so that the contribution estimate can be supplied externally, their main experiments use each client's standalone accuracy as a fixed proxy for contribution. They also evaluate alternative contribution definitions, including participation rate and using the same FLCE algorithm proposed in Xu *et al.*[14] and described in 2.7.8.

3.3.9 ShapleyFL: Robust Federated Learning Based on Shapley Value

Sun *et al.* [49] introduce ShapleyFL, a Shapley-value-based federated learning framework designed to improve robustness by assigning greater training influence to clients with higher estimated contributions. ShapleyFL is related to the framework proposed by Song *et al.* [18], since both rely on Shapley-value-based contribution estimation. However, while Song *et al.* primarily use Shapley values for client valuation and reward allocation, ShapleyFL incorporates contribution estimates directly into the training process through adaptive aggregation and client sampling.

The first algorithm proposed in ShapleyFL is AFedSV, in which clients are sampled uniformly at each communication round. After receiving the local model updates from the selected clients, the server estimates their surrogate federated Shapley values and normalizes these values to obtain aggregation weights. Therefore, unlike FedAvg, where aggregation weights are typically determined by local dataset sizes, AFedSV dynamically adjusts the aggregation rule according to each client's estimated contribution to the global training process. In this sense, contribution estimation directly affects how much influence each selected client has on the global model update.

To improve efficiency, Sun *et al.* [49] further propose AFedSV+, which extends AFedSV

with a client-importance sampling strategy. In AFedSV+, clients with higher contribution-based weights are sampled with higher probability in subsequent rounds, while the aggregation update is corrected using the corresponding sampling probabilities. Although ShapleyFL is primarily presented as a robustness-oriented framework, AFedSV+ also creates a contribution-aware participation mechanism: clients estimated to be more useful to the training process are selected more frequently and therefore receive the updated global model more often. Under the contribution fairness perspective adopted in this thesis, this sampling mechanism can be interpreted as a non-monetary incentive, because client benefit is mediated through contribution-aware access to the collaborative training process rather than through explicit monetary payment, however, contribution fairness is not measured in ShapleyFL, instead the main metric of success is achieving a higher global model accuracy.

3.4 Framework Selection Rationale

As discussed in Section 1.1, the objective of this thesis is not to reproduce every contribution-aware FL framework, but to study the mechanisms through which non-monetary rewards are claimed to produce contribution fairness. In particular, this thesis focuses on a recurring design pattern in this literature: server-side contribution estimation followed by client-specific reward generation through Top-K sparsification. This design pattern is central to the thesis because it is the point at which an estimated contribution score is converted into a personalized model update. Therefore, studying this design makes it possible to examine whether model differentiation and contribution rank preservation is caused by Top-K sparsification itself, by the gradient that's being sparsified, by the locally accumulated gradient retention, or by the interaction among these components.

Reproducing a larger set of frameworks would provide broader coverage of FLCE algorithms and reward designs, but it would not necessarily improve the causal analysis targeted in this thesis. Each additional framework introduces its own set of hyperparameters, implementation assumptions, and auxiliary mechanisms, which can make it harder to isolate the effects of the core reward-generation process. For this reason, the framework selection prioritizes mechanistic representativeness rather than breadth: the selected frameworks must implement the core design pattern studied in this thesis while differing in the specific design choices needed to support ablation analysis.

Among the surveyed frameworks, ACGSV [14] and CFFL [23] satisfy these criteria because they instantiate the same core mechanism studied in this thesis while differing in design choices that make the ablation analysis possible. Both are representative FLCE-based non-monetary reward frameworks: they estimate client contributions during training

and convert those estimates into client-specific Top-K sparsified rewards. However, they differ along three dimensions that are central to this thesis.

First, CFFL clients retain a portion of their locally accumulated gradients computed during training and combine them with the server-generated reward, whereas ACGSV clients do not retain these gradients. The portion of the locally accumulated gradient retained by client $i \in \mathcal{N}$ depends on its local dataset properties. In particular, clients with more samples and more classes in their local datasets retain a smaller portion of their accumulated gradient, whereas clients with fewer samples and classes in their local datasets retain a larger portion.

Second, CFFL defines $K_{i,t}$, which is the number of largest parameters kept in the client's reward, using both the estimated contribution and local dataset properties, such as the number of samples or distinct classes, which imposes an upper bound on reward density and produces stratified sparsity levels. In contrast, ACGSV derives $K_{i,t}$ from a non-linear transformation of the contribution score, without enforcing the same dataset-dependent stratification.

Third, CFFL applies Top-K sparsification to the unnormalized aggregated gradient obtained from a FedAvg-like aggregation rule, whereas ACGSV applies Top-K sparsification to an L2-normalized aggregated gradient. As a result, CFFL preserves the original scale of the aggregated client updates before sparsification, while ACGSV sparsifies a gradient whose norm has already been constrained by normalization. This distinction is important because the effective reward received by each client depends not only on how many coordinates are preserved, but also on the magnitude carried by those preserved coordinates.

These contrasts in design make the two frameworks suitable for the ablation studies conducted in Chapter 5 that were developed to cover the research gap and determine which aspect is the main driver of a high contribution fairness, measured according to Definition 3.

In the next sections, I describe ACGSV and CFFL in greater detail. In Chapter 4, I present the experimental methodology used to compare these two frameworks and analyze the results of that comparison.

3.5 ACGSV Framework

Xu *et al.* [14] propose ACGSV as a contribution- and incentive-aware federated learning framework that estimates client contribution without relying on a public validation dataset, and converts the resulting contribution estimates into non-monetary rewards that promote

contribution fairness, as defined in Definition 1 and measured according to Definition 3. In ACGSV, client contribution is measured through the cosine similarity between the client's accumulated gradient and the aggregated gradient of the federation. Formally, for client i at round t, this metric compares the direction of the client gradient $\Delta\vec{w}_{i,t}$ with the direction of the aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$. A cosine similarity closer to 1 indicates that the client gradient has a greater angular alignment from the aggregated gradient and therefore supports the global optimization direction. Values closer to 0 indicates that the client gradient is approximately orthogonal to the aggregated gradient, while negative values indicates that the client gradient points against the aggregated direction. Under this definition, higher cosine similarity corresponds to a higher estimated contribution, whereas lower or negative cosine similarity corresponds to a weaker or detrimental contribution, respectively.

This gradient-direction-based formulation differs from validation-accuracy-based FLCE approaches [49, 23, 22, 18, 12, 30], in which contribution is measured through the performance of a model on a validation dataset. Despite this difference, cosine-similarity-based contribution estimation has been adopted or extended by several later frameworks that treat public validation data as unavailable, impractical, or undesirable [16, 15, 38, 24, 25].

In the following subsection, I quote Xu *et al.* [14] theorem that approximates the Cosine Gradient Shapley Value to keep contribution estimation computationally feasible and grounded in the Shapley Value contribution scheme defined in Subsection 2.7.3. I present only the components of the theorem that are necessary for the discussion in this thesis, for the complete theorem statement and proof I refer to the original paper.

3.5.1 Cosine Gradient Shapley Value Approximation

Based on the discussion of contribution estimation schemes and data utility metrics in Section 2.7, cosine similarity is a data utility metric that can be combined with different contribution estimation schemes. When combined with the Shapley Value contribution scheme defined in Subsection 2.7.3, it yields the Cosine Gradient Shapley Value [14]. For client i at communication round t, the Cosine Gradient Shapley Value $\phi_{i,t}$ is defined as

$$\phi_{i,t} = \frac{1}{N} \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}} \frac{1}{\binom{N-1}{|\mathcal{S}|}} \left[\frac{\langle \Delta\vec{w}_{\mathcal{S}\cup\{i\},t},\, \Delta\vec{w}_{\mathcal{N},t} \rangle}{\|\Delta\vec{w}_{\mathcal{S}\cup\{i\},t}\|\ \|\Delta\vec{w}_{\mathcal{N},t}\|} - \frac{\langle \Delta\vec{w}_{\mathcal{S},t},\, \Delta\vec{w}_{\mathcal{N},t} \rangle}{\|\Delta\vec{w}_{\mathcal{S},t}\|\ \|\Delta\vec{w}_{\mathcal{N},t}\|} \right]. \tag{44}$$

Here, $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of clients, $\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}$ denotes a coalition that excludes client i, and $|\mathcal{S}|$ denotes its cardinality. The vector $\Delta\vec{w}_{\mathcal{S},t}$ denotes the aggregated

accumulated gradient produced by coalition $\mathcal{S}$ at round t, while $\Delta\vec{w}_{\mathcal{S}\cup\{i\},t}$ denotes the corresponding aggregated accumulated gradient after adding client i to the coalition. The vector $\Delta\vec{w}_{\mathcal{N},t}$ denotes the aggregated accumulated gradient of the grand coalition. The operator $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product, and $\|\cdot\|$ denotes the Euclidean norm. For the empty coalition, I adopt the standard convention that its utility is zero.

Computing Equation 44 exactly is computationally prohibitive because the marginal contribution of each client must be evaluated over all coalitions $\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}$. For a single client, this requires summing over 2^{N-1} coalitions. Across all clients, the exact Shapley computation involves $N2^{N-1}$ marginal-contribution terms, although the number of distinct coalition utilities can be reduced to 2^N if they are computed once and reused. In either case, the computation scales exponentially with the number of clients, making the exact Cosine Gradient Shapley Value impractical for in-training contribution estimation.

Xu *et al.* [14] address this limitation by replacing the exact coalition-based computation with a per-client approximation. Before computing this approximation, each accumulated gradient $\Delta\vec{w}_{i,t}$ is L2-normalized as

$$\vec{u}_{i,t} \leftarrow \Gamma \cdot \frac{\Delta\vec{w}_{i,t}}{\|\Delta\vec{w}_{i,t}\|}, \tag{45}$$

where $\vec{u}_{i,t}$ denotes the normalized accumulated gradient of client i at round t, and Γ is a scaling hyperparameter. This normalization controls the scale of the accumulated gradients before contribution estimation, making the approximation depend on directional agreement rather than gradient magnitude.

Let $\vec{u}_{\mathcal{N},t}$ denote the normalized aggregated gradient of the federation. Instead of computing the average marginal contribution of client i over all possible coalitions, Xu *et al.* approximate the Cosine Gradient Shapley Value by computing the cosine similarity between $\vec{u}_{i,t}$ and $\vec{u}_{\mathcal{N},t}$:

$$\psi_{i,t} = \cos(\vec{u}_{i,t}, \vec{u}_{\mathcal{N},t}) = \frac{\langle\vec{u}_{i,t}, \vec{u}_{\mathcal{N},t}\rangle}{\|\vec{u}_{i,t}\| \, \|\vec{u}_{\mathcal{N},t}\|}. \tag{46}$$

This approximation replaces the Shapley-value coalition computation with an individual contribution score based on cosine similarity. While Equation 44 measures how much client i changes the coalition utility when added to every possible coalition, Equation 46 estimates contribution directly from the directional relationship between the client's normalized accumulated gradient and the normalized aggregated gradient. Therefore, the approximation avoids coalition enumeration while preserving the same underlying data utility metric: cosine similarity with the aggregated training direction.

The approximation error is bounded by Theorem 1 in Xu *et al.* [14], restated below using the notation adopted in this thesis.

Theorem 1 (Approximation Error)**.** Suppose that $\|\vec{u}_{i,t}\| = \Gamma$ and that

$$|\langle \vec{u}_{i,t}, \vec{u}_{\mathcal{N},t} \rangle| = \frac{1}{I}$$

for all $i \in \mathcal{N}$. Let $\phi_{i,t}$ denote the exact Cosine Gradient Shapley Value and let $\psi_{i,t}$ denote the cosine-similarity approximation in Equation 46. Then, the approximation error is bounded by Equation 47

$$\phi_{i,t} - \psi_{i,t} \leq \Gamma^2 I. \tag{47}$$

In this bound, Γ controls the norm of the normalized client gradients, while I is determined by the inverse magnitude of the inner product between $\vec{u}_{i,t}$ and $\vec{u}_{\mathcal{N},t}$. The approximation error depends on both the normalization scale and the directional relationship between the client gradient and the aggregated gradient. As stated by Xu *et al.* [14], Γ cannot be reduced arbitrarily small value, because doing so may violate the condition $|\langle \vec{u}_{i,t}, \vec{u}_{\mathcal{N},t} \rangle| = 1/I$. The approximation also does not hold when $\vec{u}_{i,t}$ is orthogonal, or nearly orthogonal, to $\vec{u}_{\mathcal{N},t}$, or when the normalized gradient is close to the zero vector.

3.5.2 ACGSV Framework Workflow

The theoretical background presented above provides the foundation for understanding the ACGSV workflow. It explains why each communication round can be interpreted as a cooperative game, why accumulated client gradients are normalized before contribution estimation, how the FLCE algorithm assigns contribution scores, and how these scores are translated into client-specific non-monetary rewards. Algorithm 3 summarizes the ACGSV workflow proposed by Xu *et al.* [14].

At the initialization round $t = 0$, the server initializes the global model $\vec{w}_{0,0}$ and broadcasts it to all clients $i \in \mathcal{N}$. For each subsequent round $t > 0$, each client performs E local training epochs (lines 3 to 10) and sends its accumulated gradient $\Delta\vec{w}_{i,t}$ to the server (line 13). After receiving the accumulated gradients from the participating clients (line 15), the server applies the L2-normalization procedure described in Subsection 3.5.1, producing the normalized gradient $\vec{u}_{i,t}$ for each client (lines 16 to 18).

The server then aggregates the normalized gradients to obtain the normalized aggregated gradient of the federation, denoted by $\vec{u}_{\mathcal{N},t}$ (line 19). The aggregated gradient $\vec{u}_{\mathcal{N},t}$ is then used as the reference direction for contribution estimation, and for each client i, the server computes the round-level contribution score $\psi_{i,t}$ as the cosine similarity between the client's normalized accumulated gradient $\vec{u}_{i,t}$ and the normalized aggregated gradient $\vec{u}_{\mathcal{N},t}$ (lines 20 to 22).

The server then updates the client's importance coefficient by applying an exponential moving average to the round-level contribution score and normalizing the resulting values across all clients (line 22). This update can be written as

$$\tilde{r}_{i,t} = \alpha \cdot r_{i,t-1} + (1-\alpha) \cdot \psi_{i,t}, \qquad r_{i,t} = \frac{\tilde{r}_{i,t}}{\sum_{j \in \mathcal{N}} \tilde{r}_{j,t}}, \tag{48}$$

where α, which is a training hyperparameter, controls the contribution given to the previous importance coefficient $r_{i,t-1}$, $\psi_{i,t}$ denotes the contribution score computed at round t, $\tilde{r}_{i,t}$ denotes the unnormalized smoothed contribution score, and $r_{i,t}$ denotes the normalized importance coefficient of client i. After normalization, the importance coefficients satisfy $\sum_{i \in \mathcal{N}} r_{i,t} = 1$, so they can be interpreted as the cumulative relative contribution estimates used by the server to determine reward quality.

Once the importance coefficients have been computed, the server derives a client-specific reward vector $\vec{\nu}_{i,t}$ from the aggregated gradient (lines 25 to 28). This reward is generated through the Top-K sparsification procedure, which was previously explained in Section 2.6 and shown in Algorithm 2. Finally, the server sends the sparsified reward vector $\vec{\nu}_{i,t}$ to each client $i \in \mathcal{N}$ (line 29). Each client applies the received reward to its previous local model $\vec{w}_{i,t-1}$ to obtain its updated model for the next communication round (line 30 to 32).

The reward mechanism design also includes the parameter β, referred to by Xu *et al.* [14] as the altruism degree. This parameter controls how strongly the sparsification level differs across clients. Larger values of β reduce the gap between high- and low-contribution clients by assigning less sparse rewards to clients with smaller importance coefficients. Therefore, β regulates the trade-off between strict contribution-proportional rewards and a more egalitarian reward allocation.

Algorithm 3 - **ACGSV Framework workflow**. All clients start from the same initial model, such that $\vec{w}_{i,0} = \vec{w}_{0,0}$. The server defines the training hyperparameters, where B is the mini-batch size, E is the number of local epochs, η is the learning rate, T is the total number of communication rounds, Γ is the normalization scalar, and β is the altruism degree, and α is the EMA smoothing factor. The **Role** client refers to the data owners that participate in FL training, and **Role** server is the trusted entity that estimates contribution and generate contribution-based rewards.

for $t = 1$ to T **do**
  Role: client $i \in \mathcal{N}$
    $\mathcal{B} \leftarrow \mathcal{D}_i / B$ ▷ Splits training dataset $\mathcal{D}_i$ into $\mathcal{B}$ mini-batches
    $\vec{w} \leftarrow \vec{w}_{i,t-1}$
    for $e = 1$ to E **do**
      for batch $b \in \mathcal{B}$ **do**
        $\vec{w} \leftarrow \vec{w} - \eta_t \nabla f_i(\vec{w}; b)$ ▷ Client updates $\vec{w}_{i,t-1}$ locally
      end for
    end for
    $\vec{w_i}_{,t} \leftarrow \vec{w}$
    $\Delta\vec{w}_{i,t} \leftarrow \vec{w}_{i,t} - \vec{w}_{i,t-1}$
    $\vec{w_i}_{,t} \leftarrow \vec{w}_{i,t-1}$ ▷ Client i resets its local model to $\vec{w}_{i,t-1}$
    Send $\Delta\vec{w_i}_{,t}$ to Server

  Role: Server
    Receive $\Delta\vec{w_i}_{,t}$ from each client $i \in \mathcal{N}$
    for i in $\mathcal{N}$ **do**
      $\vec{u}_{i,t} \leftarrow \Gamma \cdot \dfrac{\Delta\vec{w}_{i,t}}{||\Delta\vec{w}_{i,t}||}$ ▷ Applies L2 normalization to $\Delta\vec{w}_{i,t}$
    end for
    $\vec{u}_{\mathcal{N},t} \leftarrow \sum_{i \in \mathcal{N}_a} r_{i,t-1} \cdot \vec{u}_{i,t}$ ▷ FedAvg with $r_{i,t-1}$

    for i in $\mathcal{N}$ **do**
      $\psi_{i,t} \leftarrow \dfrac{\langle \vec{u}_{i,t}, \vec{u}_{\mathcal{N},t} \rangle}{||\vec{u}_{i,t}||_2 \cdot ||\vec{u}_{\mathcal{N},t}||_2}$ ▷ $(\psi_{i,t})$: cosine similarity of client i
      $\tilde{r}_{i,t} = \alpha \cdot r_{i,t-1} + (1 - \alpha) \cdot \psi_{i,t}$ ▷ $(r_{i,t})$: importance coefficient of client i
    end for
    $r_{i,t} = \dfrac{\tilde{r}_{i,t}}{\sum_{j \in \mathcal{N}} \tilde{r}_{j,t}}$ ▷ Importance Coefficient normalization

    for i in $\mathcal{N}$ **do**
      $K_{i,t} \leftarrow \left\lfloor |\vec{u}_{\mathcal{N},t}| \cdot \dfrac{\tanh(\beta \cdot r_{i,t})}{\max_{j \in \mathcal{N}} (\tanh(\beta \cdot r_{j,t}))} \right\rfloor$
      $\vec{\nu}_{i,t} = \text{Top-K}(\vec{u}_{\mathcal{N},t}, K_{i,t})$ ▷ Applies Top-K Sparsification algorithm
    end for
    Send $\vec{\nu}_{i,t}$ to each client $i \in \mathcal{N}$

  Role: client $i \in \mathcal{N}$
    Receive $\vec{\nu}_{i,t}$ from Server
    $\vec{w}_{i,t} \leftarrow \vec{w}_{i,t-1} + \vec{\nu}_{i,t}$ ▷ Client's i gradient $\Delta\vec{w_i}_{,t}$
end for

3.6 CFFL framework

Lyu *et al.* [23] propose CFFL as a contribution- and incentive-aware federated learning framework that estimates client contribution using a public validation dataset and converts the resulting contribution estimates into non-monetary rewards that promote contribution fairness, as defined in Definition 1 and measured according to Definition 3. In CFFL, the server maintains a reconstructed model state for each client $i \in \mathcal{N}$ and estimates contribution by evaluating this reconstructed model on a validation dataset. Formally, for client i at round t, let

$$v_{i,t}^{\text{acc}} = \text{Acc}\left(\tilde{w}_{i,t}; \mathcal{D}_{\text{val}}\right) \tag{49}$$

denote the validation accuracy of the reconstructed client model $\tilde{w}_{i,t}$ on the public validation dataset $\mathcal{D}_{\text{val}}$, where $v_{i,t}^{\text{acc}} \in [0, 1]$. A value of $v_{i,t}^{\text{acc}} = 1$ corresponds to 100% validation accuracy, whereas $v_{i,t}^{\text{acc}} = 0$ corresponds to 0% validation accuracy. Under this definition, a higher validation accuracy corresponds to a higher estimated contribution, while a lower validation accuracy corresponds to a lower estimated contribution.

CFFL depends on two assumptions that are central to its aggregation, contribution estimation, and reward generation workflow:

- **Validation dataset availability:** CFFL assumes that the server has access to a validation dataset that can be used to evaluate each reconstructed client model. This assumption is necessary because CFFL estimates contribution through validation accuracy, therefore, without a validation dataset, the server cannot compute the utility values required to derive client-specific rewards.
- **Honest reporting of dataset statistics:** CFFL assumes that each client truthfully reports the number of local training samples and the number of distinct classes in its dataset. The server uses these statistics to characterize the federation as either data-quantity-skewed or class-label-skewed. This characterization determines both the aggregation rule used to compute the global update and the rule used to assign the Top-K sparsification level for each client. The federation dataset statistics are also shared across the federation.

Lyu *et al.* [23] design CFFL so that the server uses client-reported dataset statistics to determine both the gradient aggregation rule and the Top-K sparsification rule. However, Lyu *et al.* [23] do not specify a mechanism for validating these self-reported statistics, nor does it define a precedence rule for cases in which data-quantity skew and class-label

skew are both present. As a result, CFFL treats these statistics as trusted inputs to the server-side workflow.

Although these are a strong assumptions in a realistic deployment, it is not unusual in federated learning research. Several frameworks rely on client-reported quantities, such as the number of local samples, local loss, local accuracy, or other client-specific statistics, to support aggregation weighting, client sampling, contribution estimation, reward allocation, or participant filtering [17, 18, 22, 12, 7, 42, 25, 30]. Since Byzantine participation, strategic misreporting, and collusion are outside the scope of this thesis, I treat these reported statistics as valid inputs when replicating and evaluating CFFL. Here, I focus only on the assumptions required to describe the CFFL workflow and its framework-specific requirements, while the full set of experimental assumptions is presented in Chapter 4 Methodology (Section 4.1).

Furthermore, if the federation is characterized as data-quantity-skewed, the server aggregates client updates using sample-size weights:

$$\Delta\vec{w}_{\mathcal{N},t} = \sum_{i\in\mathcal{N}} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{N}|} \cdot \Delta\vec{w}_{i,t} \tag{50}$$

if the federation is characterized as class-label-skewed, the server instead weights each client update according to the number of distinct classes available in that client's local dataset:

$$\Delta\vec{w}_{\mathcal{N},t} = \sum_{i\in\mathcal{N}} \frac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}} |\mathcal{C}_j|} \cdot \Delta\vec{w}_{i,t} \tag{51}$$

where $|\mathcal{D}_i|$ denotes the number of samples in the local dataset of client i, $|\mathcal{D}_\mathcal{N}|$ denotes the total number of samples across the federation, and $|\mathcal{C}_i|$ denotes the number of distinct classes present in the local dataset of client i.

The same distinction also determines how the server determines the client-specific K_i value for the Top-K sparsification. Let $r_{i,t}$ denote the cumulative contribution estimated for client i until round t, and let $|\Delta\vec{w}_{\mathcal{N},t}|$ denote the dimension of the aggregated update used to generate the reward. In the data-quantity-skewed setting, CFFL computes

$$K_{i,t} = \frac{r_{i,t}}{\max_{j\in\mathcal{N}} r_{j,t}} \cdot \frac{|\mathcal{D}_i|}{\max_{j\in\mathcal{N}} |\mathcal{D}_j|} \cdot |\Delta\vec{w}_{\mathcal{N},t}|. \tag{52}$$

In the class-label-skewed setting, CFFL replaces the sample-size factor with a class-diversity factor:

$$K_{i,t} = \frac{r_{i,t}}{\max_{j\in\mathcal{N}} r_{j,t}} \cdot \frac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}} |\mathcal{C}_j|} \cdot |\Delta\vec{w}_{\mathcal{N},t}|. \tag{53}$$

Thus, the sparsification level depends on two factors: the cumulative estimated contribution of the client, $r_{i,t}$, and the dataset statistic used to characterize the federation, either $|\mathcal{D}_i|$ or $|\mathcal{C}_i|$. Since both factors are normalized by their maximum value across clients, each factor is upper-bounded by 1. Therefore, the dataset statistic acts as a client-specific upper bound on the density of the Top-K reward, while the contribution ratio further scales this bound according to the client's estimated contribution.

For instance, in a class-label-skewed setting, if client i has $|\mathcal{C}_i| = 2$ and the maximum number of classes held by any client is $\max_{j\in\mathcal{N}} |\mathcal{C}_j| = 10$, then the class-diversity factor is 0.2. As a result, even if client i has the largest contribution score, such that $r_{i,t}/\max_{j\in\mathcal{N}} r_{j,t} = 1$, its reward can preserve at most the top 20% largest-magnitude coordinates of the aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$. If its contribution ratio is smaller than 1, the resulting reward is even sparser.

Additionally, in CFFL, the server does not send only the Top-K sparsified reward to each client. It also provides the federation dataset distribution statistics, so that the clients can determine how much of the locally accumulated gradient should be preserved. Therefore, the portion of the locally accumulated gradient retained by client $i \in \mathcal{N}$ depends on its local dataset properties. In the data-quantity-skewed setting, each client updates its local model as shown in Equation 54:

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{D}_i}{\max_{j\in\mathcal{N}} \mathcal{D}_j} \cdot \Delta\vec{w}_{i,t} \tag{54}$$

where $\mathcal{D}_i$ represents the number of samples in client's $i \in \mathcal{N}$ local dataset, $\vec{w}_{i,t-1}$ it's the client model at the beginning of round t, $\Delta\vec{w}_{i,t}$ is the client's locally computed gradient, and $\vec{\nu}_{i,t}$ it the Top-K sparsified reward received by the client. In the scenario where the dataset follows a class-label-skewed scenario, each client updates its local model using as shown in Equation 55:

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{C}_i}{\max_{j\in\mathcal{N}} \mathcal{C}_j} \cdot \Delta\vec{w}_{i,t} \tag{55}$$

where $\mathcal{C}_i$ represents the number of distinct classes in client's $i \in \mathcal{N}$ local dataset.

3.6.1 CFFL Framework Workflow

The previous section presented the background knowledge required to understand the CFFL workflow. It explains how the server defines the aggregation rules based on the data distribution across the federation, how the FLCE algorithm assigns contribution scores,

how both the data distribution and the FLCE algorithm are translated into client-specific non-monetary rewards through Top-K sparsification, and how the clients update their models. Algorithm 4 and Algorithm 5 show the workflows of the CFFL framework for the data-quantity-skewed and class-label-skewed data partitioning scenarios, respectively. Since both algorithms follow the same overall structure, they use the same line references whenever the workflow is shared. The differences between them are limited to the distribution-dependent steps described above.

As with the ACGSV framework, at the initialization round $t = 0$, the server initializes the global model $\vec{w}_{0,0}$ and broadcasts it to all clients $i \in \mathcal{N}$. For each subsequent round $t > 0$, each client performs E local training epochs and sends its accumulated gradient $\Delta\vec{w}_{i,t}$ to the server, together with its local dataset statistics (lines 2 to 13). After receiving the accumulated gradients and local dataset statistics from the participating clients (line 15), the server aggregates them using the FedAvg-like rule determined by the data distribution across the federation (line 16). If the federation is characterized as data-quantity-skewed, the server applies Equation 50; if it is characterized as class-label-skewed, the server applies Equation 51.

After aggregation, the server reconstructs each client model state, denoted by $\tilde{w}_{i,t}$ (line 18). This reconstruction is possible because all clients start from the same initial model $\vec{w}_{0,0}$ and because the server keeps track of each client's model state throughout training. The server then evaluates each reconstructed model $\tilde{w}_{i,t}$ on the validation dataset $\mathcal{D}_{\text{val}}$, producing the validation accuracy $v_{i,t}^{\text{acc}} \in [0, 1]$ (lines 17 to 20). This validation accuracy is used as the round-level utility of client i: higher validation accuracy indicates a larger estimated contribution, while lower validation accuracy indicates a smaller estimated contribution.

CFFL then transforms the validation accuracy through a hyperbolic sine function and incorporates the result into a moving-average update (line 21 to 23). Unlike the ACGSV, which treated the moving average smoothing factor α as a framework hyperparameter, CFFL keeps it fixed as 0.5. The importance coefficient update is determined as

$$\tilde{r}_{i,t} = 0.5\, r_{i,t-1} + 0.5 \sinh\left(\theta \frac{v_{i,t}^{\text{acc}}}{\sum_{j\in\mathcal{N}} v_{j,t}^{\text{acc}}}\right), \qquad r_{i,t} = \frac{\tilde{r}_{i,t}}{\sum_{j\in\mathcal{N}} \tilde{r}_{j,t}}. \tag{56}$$

where θ is a framework-specific hyperparameter[2] that controls the nonlinear scaling applied to the validation accuracy $v_{i,t}^{\text{acc}}$, $\tilde{r}_{i,t}$ denotes the unnormalized smoothed contribution

[2]In the original CFFL formulation, the nonlinear scaling hyperparameter is denoted by α. In this thesis, I denote this hyperparameter by θ to avoid confusion with the smoothing factor α used in the moving-average update.

score, and $r_{i,t}$ denotes the normalized importance coefficient of client i at round t. After normalization, the coefficients satisfy $\sum_{i\in\mathcal{N}} r_{i,t} = 1$, allowing the server to interpret $r_{i,t}$ as the client's cumulative contribution estimate to the training outcome and it is used to determine the client-specific Top-K reward.

After $r_{i,t}$ has been computed and considering the data distribution in federation, the server uses either Equation 52 or 53 to derive the value K_i for each client $i \in \mathcal{N}$ and uses it to apply the Top-K sparsification algorithm, as defined in Algorithm 2, on the aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$ (lines 25 to 28). The CFFL server then sends to each client its Top-K sparsified gradient reward $\vec{\nu}_{i,t}$, and the grand coalition data number of total samples $|\mathcal{D}_{\mathcal{N}}|$ and total number of distinct classes $|\mathcal{C}_{\mathcal{N}}|$ (line 30). Upon receiving $\vec{\nu}_{i,t}$, $|\mathcal{D}_{\mathcal{N}}|$, and $|\mathcal{C}_{\mathcal{N}}|$ each client use the dataset statistics to determine the portion of their locally accumulated gradients that should be preserved, and the portion of the locally accumulated gradient retained by client $i \in \mathcal{N}$ depends on its local dataset properties. and defined in Equations 54 and 55

Algorithm 4 - CFFL Framework Workflow - Quantity Skew. All clients start from the same initial model, such that $\vec{w}_{i,0} = \vec{w}_{0,0}$. The server defines the training hyperparameters, where B is the mini-batch size, E is the number of local epochs, η is the learning rate, T is the total number of communication rounds, and θ is the punishment factor. The **Role** client refers to the data owners that participate in FL training, and **Role** server is the trusted entity that estimates contribution and generate contribution-based rewards. Here, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ denote the number of samples and distinct classes in client i's local dataset, while $|\mathcal{D}_\mathcal{N}|$ and $|\mathcal{C}_\mathcal{N}|$ denote the corresponding totals across the federation.

for $t = 1$ to T **do**
  Role: client $i \in \mathcal{N}$
    $\mathcal{B} \leftarrow \mathcal{D}_i / B$ ▷ Splits training dataset $\mathcal{D}_i$ into $\mathcal{B}$ mini-batches
    $\vec{w} \leftarrow \vec{w}_{i,t-1}$
    for $e = 1$ to E **do**
      for batch $b \in \mathcal{B}$ **do**
        $\vec{w} \leftarrow \vec{w} - \eta_t \nabla f_i(\vec{w}; b)$ ▷ Client updates $\vec{w}_{i,t-1}$ locally
      end for
    end for
    $\vec{w_{i,t}} \leftarrow \vec{w}$
    $\Delta\vec{w}_{i,t} \leftarrow \vec{w}_{i,t} - \vec{w}_{i,t-1}$
    $\vec{w_{i,t}} \leftarrow \vec{w}_{i,t-1}$ ▷ Client i resets its local model to $\vec{w}_{i,t-1}$
    Send $\Delta\vec{w_{i,t}}$, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ to Server

  Role: Server
    Receive $\Delta\vec{w_{i,t}}$, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ from each client $i \in \mathcal{N}$
    $\Delta\vec{w}_{\mathcal{N},t} \leftarrow \sum_{i \in \mathcal{N}_a} \frac{|\mathcal{D}_i|}{|\mathcal{D}_\mathcal{N}|} \cdot \Delta\vec{w}_{i,t}$ ▷ FedAvg on gradients
    for i in $\mathcal{N}$ **do**
      $\tilde{w}_{i,t} \leftarrow \tilde{w}_{i,t-1} + \Delta\vec{w}_{i,t}$ ▷ Server reconstructs client i model
      $v_{i,t}^{\text{acc}} \leftarrow \text{Acc}(\tilde{w}_{i,t}; \mathcal{D}_{\text{val}})$ ▷ Acc$(\cdot)$: Measures reconstructed model accuracy
    end for
    for i in $\mathcal{N}$ **do**
      $\tilde{r}_{i,t} = 0.5\, r_{i,t-1} + 0.5 \sinh\left(\theta \frac{v_{i,t}^{\text{acc}}}{\sum_{j \in \mathcal{N}} v_{j,t}^{\text{acc}}}\right)$
    end for
      $r_{i,t} = \frac{\tilde{r}_{i,t}}{\sum_{j \in \mathcal{N}} \tilde{r}_{j,t}}$ ▷ Normalize client i contribution
    for i in $\mathcal{N}$ **do**
      $K_{i,t} = \frac{r_{i,t}}{\max_{j \in \mathcal{N}}(r_{j,t})} \cdot \frac{|\mathcal{D}_i|}{\max_{j \in \mathcal{N}}(|\mathcal{D}_j|)} \cdot |\Delta\vec{w}_{\mathcal{N},t}|$
      $\vec{\nu}_{i,t} = \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_{i,t})$
      $\tilde{w}_{i,t} \leftarrow \tilde{w}_{i,t} + \vec{\nu}_{i,t}$ ▷ Updates the reconstructed model
    end for
    Send $\vec{\nu}_{i,t}$, $\max_{j \in \mathcal{N}}(|\mathcal{D}_j|)$ and $\max_{j \in \mathcal{N}}(|\mathcal{C}_\mathcal{N}|)$ to each client $i \in \mathcal{N}$

  Role: client $i \in \mathcal{N}$
    Receive $\vec{\nu}_{i,t}$, $|\mathcal{D}_\mathcal{N}|$ and $|\mathcal{C}_\mathcal{N}|$ from Server
    $\vec{w}_{i,t} \leftarrow \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \Delta\vec{w}_{i,t} \cdot \frac{|\mathcal{D}_i|}{\max_{j \in \mathcal{N}}(|\mathcal{D}_j|)}$
end for

Algorithm 5 - CFFL Framework Workflow - Class Label Skew. All clients start from the same initial model, such that $\vec{w}_{i,0} = \vec{w}_{0,0}$. The server defines the training hyperparameters, where B is the mini-batch size, E is the number of local epochs, η is the learning rate, T is the total number of communication rounds, and θ is the punishment factor. The **Role** client refers to the data owners that participate in FL training, and **Role** server is the trusted entity that estimates contribution and generate contribution-based rewards. Here, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ denote the number of samples and distinct classes in client i's local dataset, while $|\mathcal{D}_\mathcal{N}|$ and $|\mathcal{C}_\mathcal{N}|$ denote the corresponding totals across the federation.

for $t = 1$ to T **do**
  Role: client $i \in \mathcal{N}$
    $\mathcal{B} \leftarrow \mathcal{D}_i / B$ ▷ Splits training dataset $\mathcal{D}_i$ into $\mathcal{B}$ mini-batches
    $\vec{w} \leftarrow \vec{w}_{i,t-1}$
    for $e = 1$ to E **do**
      for batch $b \in \mathcal{B}$ **do**
        $\vec{w} \leftarrow \vec{w} - \eta_t \nabla f_i(\vec{w}; b)$ ▷ Client updates $\vec{w}_{i,t-1}$ locally
      end for
    end for
    $\vec{w_i}_{,t} \leftarrow \vec{w}$
    $\Delta\vec{w}_{i,t} \leftarrow \vec{w}_{i,t} - \vec{w}_{i,t-1}$
    $\vec{w_i}_{,t} \leftarrow \vec{w}_{i,t-1}$ ▷ Client i resets its local model to $\vec{w}_{i,t-1}$
    Send $\Delta\vec{w_i}_{,t}$, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ to Server

  Role: Server
    Receive $\Delta\vec{w_i}_{,t}$, $|\mathcal{D}_i|$ and $|\mathcal{C}_i|$ from each client $i \in \mathcal{N}$
    $\Delta\vec{w}_{\mathcal{N},t} = \sum_{i\in\mathcal{N}} \Delta\vec{w}_{i,t} \cdot \dfrac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}}(|\mathcal{C}_j|)}$ ▷ FedAvg on gradients
    for i in $\mathcal{N}$ **do**
      $\tilde{w}_{i,t} \leftarrow \tilde{w}_{i,t-1} + \Delta\vec{w}_{i,t}$ ▷ Server reconstructs client i model
      $v_{i,t}^{\text{acc}} \leftarrow \text{Acc}(\tilde{w}_{i,t}; \mathcal{D}_{\text{val}})$ ▷ Acc$(\cdot)$: Measures reconstructed model accuracy
    end for
    for i in $\mathcal{N}$ **do**
      $\tilde{r}_{i,t} = 0.5\, r_{i,t-1} + 0.5 \sinh\left(\theta \dfrac{v_{i,t}^{\text{acc}}}{\sum_{j\in\mathcal{N}} v_{j,t}^{\text{acc}}}\right)$
    end for
      $r_{i,t} = \dfrac{\tilde{r}_{i,t}}{\sum_{j\in\mathcal{N}} \tilde{r}_{j,t}}$ ▷ Normalize client i contribution
    for i in $\mathcal{N}$ **do**
      $K_i = \dfrac{r_{i,t}}{\max_{j\in\mathcal{N}}(r_{j,t})} \cdot \dfrac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}}(|\mathcal{C}_j|)} \cdot |\Delta\vec{w}_{\mathcal{N},t}|$
      $\vec{\nu}_{i,t} = \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_i)$
      $\tilde{w}_{i,t} \leftarrow \tilde{w}_{i,t} + \vec{\nu}_{i,t}$ ▷ Updates the reconstructed model
    end for
    Send $\vec{\nu}_{i,t}$, $\max_{j\in\mathcal{N}}(|\mathcal{D}_j|)$ and $\max_{j\in\mathcal{N}}(|\mathcal{C}_\mathcal{N}|)$ to each client $i \in \mathcal{N}$

  Role: client $i \in \mathcal{N}$
    Receive $\vec{\nu}_{i,t}$, $|\mathcal{D}_\mathcal{N}|$ and $|\mathcal{C}_\mathcal{N}|$ from Server
    $\vec{w}_{i,t} \leftarrow \vec{w}_{i,t-1} + \Delta\vec{w_i}_{,t} + \vec{\nu}_{i,t} - \Delta\vec{w}_{i,t} \cdot \dfrac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}}(|\mathcal{C}_j|)}$
end for

3.6.2 CFFL - Gradient Aggregation modification for class-label-skewed distribution

In the experiments conducted with CFFL under class-label-skewed data partitioning, the clients exhibited unstable learning behavior. After analyzing the framework components and comparing the implementation with the MNIST results reported by Lyu *et al.* [23], I found that this instability was associated with the aggregation rule shown in Equation 51. In particular, the max-normalization term based on the number of local classes produced highly imbalanced aggregation weights across clients. Several hyperparameter adjustments were tested, including changes to the learning rate, number of local epochs, and batch size, but these changes were not sufficient to consistently stabilize training. Moreover, further hyperparameter tuning would introduce additional confounding factors into the analysis of the reults.

Since the objective of this thesis is not to maximize the absolute accuracy of CFFL, but to isolate how Top-K sparsification, the choice of sparsified gradient, and the client-model update rule affect model differentiation and contribution fairness, I modified the class-label-skewed aggregation rule as shown in Equation 57.

$$\Delta\vec{w}_{\mathcal{N},t} = \sum_{i\in\mathcal{N}} \Delta\vec{w}_{i,t} \cdot \frac{|\mathcal{C}_i|}{\sum_{j=1}^{\mathcal{N}}(|\mathcal{C}_j|)} \tag{57}$$

This modification replaces the max-normalized class count weighting with a sum-normalized class count weighting, which stabilizes the aggregation while preserving the dependency on the clients' local class coverage. Without this modification, the resulting training instability would compromise the interpretation of the results obtained from the experiments in Chapter 4. Therefore, this modification should be interpreted as a stabilization step that enables a controlled comparison of the mechanisms studied in this thesis, rather than as an exact reproduction of the original CFFL class-label-skewed setting. Nevertheless, for completeness, the results obtained using the original aggregation rule are reported in Appendix E.

3.7 Chapter Conclusion

This chapter reviewed state-of-the-art FL frameworks that incorporate contribution fairness as a central design objective. The surveyed frameworks were organized according to

their contribution estimation strategy, incentive type, reward distribution stage, and reward method. This comparison showed that contribution-aware FL can be broadly divided into frameworks that use monetary compensation and frameworks that use non-monetary rewards. While monetary frameworks use contribution estimates to allocate financial incentives, non-monetary frameworks use contribution estimates to control the quality of the model or gradient information returned to each client.

The non-monetary frameworks are the most relevant to this thesis because they directly connect contribution estimation to in-training rewards that influence the client's model convergence. From the surveyed literature, several frameworks translate contribution scores into client-specific Top-K sparsified rewards, where higher-contributing clients receive denser and potentially more informative updates than lower-contributing clients. These same frameworks also adopt the Pearson's Correlation Coefficient between the clients' standalone test accuracy and the clients' federated learning test accuracy is used as the key metric of success to measure contribution fairness.

Based on the motivation presented in Chapter 1 of the thesis, ACGSV and CFFL were selected as the main frameworks for experimental analysis. These frameworks are suitable for this thesis because they share the same general objective of producing contribution-based non-monetary rewards, while differing in design choices that are central to the research questions.

These differences make ACGSV and CFFL useful case studies for analyzing the mechanisms behind model differentiation and contribution fairness. Overall, this chapter provides the literature and framework foundation for the experimental analysis developed in the remainder of the thesis.

The next chapter build on this foundation by implementing ACGSV and CFFL, evaluating their Contribution Fairness behavior, and comparing the framework results with respect to their client-level model differentiation and contribution fairness, measured as in Definition 3.

Chapter 4

Experimental Analysis of contribution fairness frameworks

Building on the descriptions of ACGSV and CFFL presented in Chapter 3, Sections 3.5 and 3.6, this chapter provides a comprehensive experimental comparison of both frameworks. I first define the experimental methodology, including the benchmarking datasets, the data partitioning strategies, and the model architectures used in the experiments. I then present and analyze the results obtained with CFFL and ACGSV under the defined experimental settings.

Finally, I summarize the main findings and discuss why the joint analysis of Kendall's τ rank correlation coefficient, combined with the Gini Mean Difference (GMD), provides a more complete evaluation of contribution fairness than Pearson's r correlation coefficient alone. To maintain consistency with existing work, I first use each client's standalone test accuracy as the reference contribution value. Then, to further strengthen the claim, I compute each client's Shapley Values using test accuracy as the coalition utility and examining whether the proposed evaluation protocol yields consistent conclusions under this coalition-based measure of contribution.

4.1 Methodology

ACGSV [14] and CFFL [23] share the same objective: achieve contribution fairness, such that clients receive rewards that are commensurate to their contributions to the training (Definition 1). Furthermore, in both frameworks, estimated contribution values generated

by their respective FLCE algorithms is then used to derive as a contribution-dependent Top-K sparsified version of the aggregated gradient, where clients with higher estimated contribution receive denser rewards and clients with lower estimated contribution receive sparser rewards. Moreover, both frameworks originally evaluated contribution fairness through the Pearson's r correlation coefficient (Definition 3).

Therefore, ACGSV and CFFL provide a suitable basis for comparing how contribution estimation, Top-K sparsification in reward generation, and locally accumulated gradient affect contribution fairness through model differentiation and contribution rank preservation. The goal of these experiments is methodological rather than accuracy-driven, the intention is not to claim that a framework is superior or inferior than each other, but rather to evaluate both frameworks under the same experimental settings to compare their fairness outcomes.

4.1.1 Evaluation Metrics

Traditional federated learning studies commonly use global model test accuracy as the main performance metric [1, 5, 6, 31, 32]. However, in FL contribution fairness studies, frameworks are evaluated through top-contributor test accuracy, the mean test accuracy across clients, and the Pearson's r correlation coefficient between each clients' standalone test accuracy and their respective FL test accuracy values [14, 21, 22, 23, 24, 63]. By contrast, the objective of this thesis is not to determine which framework achieves the highest accuracy, but rather to analyze contribution fairness. Therefore, I evaluate the frameworks using the following metrics:

1. **Pearson's r Correlation Coefficient:** To remain consistent with the evaluation protocol adopted in related work, I compute Pearson's r between the final test accuracy achieved by each client in Standalone mode and the final test accuracy achieved by the same client after federated training. Pearson's r takes values in the interval $[-1, 1]$, where values closer to 1 indicate a stronger fairness, while values closer to -1 indicate an inverse relationship, and values closer to 0 are interpreted as evidence no contribution fairness. Instead, clients converge toward similar final performances regardless of their standalone performance.

2. **Gini Mean Difference (GMD):** I use the GMD, defined in Section 2.9, to measure the average pairwise difference between the final test accuracies obtained by clients with unit in percentage points. This metric is used to quantify the extent to which the framework produces differentiated client model performances and its unit

is given in percentage points. A GMD value of zero indicates that client models converge to identical performance, whereas larger values indicate stronger performance differentiation across clients.

3. **Kendall's τ Rank Correlation Coefficient:** I use the Kendall's τ to evaluate whether the FL framework is capable of making clients converge into models whose test performance has rank agreement with the reference contribution ranks. I compute Kendall's τ between the final test accuracy achieved by each client in Standalone mode and the final test accuracy achieved by the same client after federated training. Kendall's τ takes values in the interval $[-1, 1]$, where values closer to 1 indicate a stronger reference contribution rank preservation, while values closer to -1 indicate an inverse relationship, and values closer to 0 indicate weaker agreement with the reference contribution ranks, and interpreted as evidence no contribution fairness.

Each experiment is conducted with a federation of 10 clients, which is one of the federation sizes used in the original experiments of Xu *et al.* [14] and Lyu *et al.* [23]. Because correlation coefficients computed over small client populations can be sensitive to sampling variability [64, 65, 58], I repeat each experimental setting over five independent runs to increase the statistical significance of the result obtained. For each dataset and partitioning strategy described later in this section, I use the fixed partitioning seeds 1111, 2222, 3333, 4444, and 5555 to control the data shuffling and client partitioning procedure, allowing each run to evaluate a different client data allocation, but enabling cross-framework comparison of the obtained results.

To isolate the effect of data partitioning from the effect of model initialization, I use a fixed initialization seed 1000 for the global model in all runs. Therefore, variation across runs are primarily attributed to the client data shuffling before partitioning and the stochastic nature of ML training rather than to differences in the initial model parameters.

For reporting framework-level fairness results across the five runs, I average Pearson's r Correlation Coefficients using Fisher's z-transformation, which is necessary because this correlation coefficient is bounded and non-additive, so directly averaging raw correlation values can produce biased summaries [64, 65, 58]. For each Pearson's r correlation coefficient value obtained in the run, I compute

$$z = \frac{1}{2} \ln \left(\frac{1 + r}{1 - r} \right) \tag{53}$$

average the resulting z-values arithmetically, generating $\bar{z}$ and then apply the inverse transformation on the average value

$$\bar{r} = \tanh(\bar{z}) \tag{54}$$

to obtain the reported mean correlation. The GMD is averaged arithmetically across runs.

In the following section I describe the datasets that were used for benchmarking and define how the data was split across the federation of clients.

4.1.2 Client-Level Shapley Values

Although Shapley Value computation is generally considered impractical in large federations due to its exponential complexity, it remains feasible in the experimental setting used in this thesis. All experiments are conducted with 10 clients, which requires evaluating $2^{10} = 1024$ possible client coalitions for each dataset, partitioning strategy, and experimental run. Therefore, I compute the Shapley Value of each client using the formulation introduced in Section 2.7.3, with test accuracy as the coalition utility metric (Section 2.7.6).

The Shapley Value provides a set of coalition-based contribution values for each client, which is computed by averaging each client's marginal effect across all possible coalitions. This makes it a stronger contribution reference than standalone test accuracy, since it accounts for how each client's data affects performance in combination with the data of every possible coalition. For this reason, I use the Shapley Values as an additional validation step for the fairness analysis conducted in this chapter.

The main analysis is first presented using standalone test accuracy as the client reference contribution value, in order to remain consistent with the evaluation protocol adopted in the existing contribution fairness literature. I then repeat the correlation-based analysis using the Shapley Values as the reference contribution value. This complementary analysis evaluates whether the conclusions obtained with standalone test accuracy remain valid when contribution is measured through coalition-based marginal utility. As shown at the end of this chapter, the Shapley-value-based results corroborate the main findings: the limitations of Pearson's r correlation coefficient to measure fairness, the advantages of evaluating contribution fairness through Kendall's τ_b and GMD.

4.1.3 Benchmark Datasets Description

For the experiments conducted in this thesis, I used four datasets that are commonly adopted for benchmarking in the federated learning literature: MNIST [66], Fashion-MNIST (FA-MNIST) [67], CIFAR [68], and Stanford Sentiment Treebank 2 (SST2) [69]. A brief description of each dataset is provided below.

- **MNIST:** MNIST consists of 70,000 grayscale images of handwritten digits, each with resolution 28×28 pixels, extracted from two NIST databases. The official split contains 60,000 training images and 10,000 test images. The dataset has 10 classes, corresponding to digits from 0 to 9, with approximately 7,000 samples per class. Due to its simple visual structure, centered digits, and low intra-class variability, MNIST is generally considered one of the easiest benchmark datasets for image classification.

- **FA-MNIST:** Fashion-MNIST (FA-MNIST) consists of 70,000 grayscale images of Zalando fashion products, each with resolution 28×28 pixels. The official split contains 60,000 training images and 10,000 test images. The dataset has 10 classes, each representing a different clothing category, with 7,000 images per class. Although it has the same image format and dataset size as MNIST, Fashion-MNIST is more challenging because the visual differences between clothing categories are more subtle than those between handwritten digits.

- **CIFAR10:** CIFAR10 consists of 60,000 colour images with resolution 32×32 pixels, divided into 10 object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset contains 50,000 training images and 10,000 test images, with 6,000 samples per class. Unlike MNIST and Fashion-MNIST, CIFAR10 is a natural image dataset, meaning that the objects appear with substantial variation in pose, scale, lighting conditions, background composition, and viewpoint. In addition, several classes exhibit strong visual similarity, such as cat versus dog and automobile versus truck, which increases the difficulty of the classification task. Its low image resolution further limits the amount of discriminative detail available to the model, making CIFAR10 a substantially more challenging benchmark than MNIST and Fashion-MNIST.

- **SST2:** The Stanford Sentiment Treebank 2 (SST2) is a sentence-level natural language processing dataset for binary sentiment classification. Each sample is a sentence labeled as expressing either positive or negative sentiment, and sentence lengths vary across samples. The dataset contains 67,349 training samples, 872 validation samples, and 1,821 unlabeled test samples in its standard form. Since the official test labels are not publicly available, the test split was not used in the experiments. Instead, a subset of the training data was separated to construct a labeled test set.

For the simulations conducted throughout this thesis, three dataset partitioning strategies were used to distribute the training samples of each dataset across clients: Independent and Identically Distributed (IID), Power-Law Distribution (POW), and Class-Label Imbalance Distribution (CLA). Since one of the evaluated frameworks requires the availability

of a validation dataset, the same train-validation-test treatment was applied to all frameworks and ablation studies. This ensures that the comparisons reported in this thesis are not confounded by differences in the data available to each framework. For each dataset-partitioning combination, as mentioned in Subsection 4.1.1, five independent runs were conducted using fixed seeds 1111, 2222, 3333, 4444, and 5555. The same set of five seeds was reused for all dataset-partitioning combinations. In each run, the corresponding seed was used to shuffle the available training samples before partitioning, producing a different assignment of samples to clients across runs.

For MNIST, FA-MNIST, and CIFAR, the predefined training and test splits were preserved. For each run, the training split of each dataset was first shuffled using the corresponding seed. Then, 10% of the samples in the training split were reserved to create a validation set, while the remaining 90% were distributed across clients according to the selected partitioning strategy. The predefined test splits of MNIST, FA-MNIST, and CIFAR were used only for client model evaluation and were not involved in client training, validation, or partitioning. For the SST2 dataset, however, the predefined test split is unlabeled. Therefore, it was necessary to construct a test set from the existing labeled training split. The original training split was first shuffled and divided into an 80/20 train-test split, with 20% of the samples reserved for testing. To form the SST2 validation set, the existing SST2 validation split, which contains 872 labeled samples, was combined with an additional 4,128 samples from the remaining training data to construct a validation set of approximately 5,000 samples. All client models were evaluated on the entire common test split, which is a common practice in Incentive-Aware FL studies because it allows the distinct models produced for each client to be evaluated under the same testing conditions, thereby supporting cross-client and cross-framework comparisons.

The remaining training samples were then partitioned across clients using the same IID, POW, and CLA strategies adopted by related work [14, 23, 22, 47, 49, 18], which are described below.

- **Independent and Identically Distributed (IID):** In this setting, the training data is evenly partitioned across clients. The training dataset is first shuffled using fixed seeds for reproducibility and then evenly partitioned so that each client receives the same number of samples and the same proportion of samples for each class, while ensuring no overlapping samples across clients.
- **Power-Law Distribution (POW):** In this setting, the training data is distributed across clients according to a power-law distribution to create data quantity skew, where clients retain samples from all classes but they differ in the total number of

samples assigned for each client. The partitioning procedure was performed to ensure that there are no training samples overlapping across clients. Table 4.1 contains the number of samples that each client has for each dataset. The parameter α for the power-law distribution equation was set to 1.65, following the configuration used by Xu *et al.* [14] in their experiments and provided in their GitHub repository.

- **Class Label Imbalance (CLA):** In this setting, all clients are assigned the same number of local training samples, but the number of distinct classes represented in each local dataset varies across clients. Therefore, the imbalance is created through class-label diversity rather than through local dataset size. Following the configuration used in related work [14, 23, 22], each client receives 600 training samples for MNIST and Fashion-MNIST, and 3000 training samples for CIFAR10. To illustrate the CLA partitioning strategy, Client 9 receives 600 samples drawn from a single class, while client 0 receives 600 samples distributed approximately evenly across all 10 classes. The remaining clients are assigned datasets between these two extremes, with progressively larger class diversity. Thus, all clients have the same dataset size, but clients differ in how many labels are represented in their local data. Table 4.2 presents the number of samples per class assigned to each client for the MNIST and FA-MNIST datasets, while 4.3 presents the number of samples per class assigned to each client on CIFAR10. Unlike the IID and POW partitioning strategies, the CLA setting does not require the client datasets to be mutually disjoint, therefore, distinct clients may have overlapping samples. This partitioning strategy was applied to all datasets except SST2.

Client ID	MNIST	FA-MNIST	CIFAR10	SST2
Client 0	10158	10158	8465	9361
Client 1	9105	9105	7588	8388
Client 2	8047	8047	6706	7413
Client 3	6988	6988	5824	6438
Client 4	5930	5930	4942	5463
Client 5	4871	4871	4059	4488
Client 6	3813	3813	3177	3513
Client 7	2754	2754	2295	2538
Client 8	1696	1696	1413	1563
Client 9	638	638	531	587

Table 4.1: Power-law training dataset distribution across clients.

Client ID	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Client 0	60	60	60	60	60	60	60	60	60	60
Client 1	66	66	66	66	66	66	66	66	72	0
Client 2	75	75	75	75	75	75	75	75	0	0
Client 3	85	85	85	85	85	85	90	0	0	0
Client 4	100	100	100	100	100	100	0	0	0	0
Client 5	120	120	120	120	120	0	0	0	0	0
Client 6	150	150	150	150	0	0	0	0	0	0
Client 7	200	200	200	0	0	0	0	0	0	0
Client 8	300	300	0	0	0	0	0	0	0	0
Client 9	600	0	0	0	0	0	0	0	0	0

Table 4.2: Class Label Imbalance training dataset distribution across clients for MNIST and FA-MNIST datasets

Client ID	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Client 0	300	300	300	300	300	300	300	300	300	300
Client 1	333	333	333	333	333	333	333	333	336	0
Client 2	375	375	375	375	375	375	375	375	0	0
Client 3	428	428	428	428	428	428	432	0	0	0
Client 4	500	500	500	500	500	500	0	0	0	0
Client 5	600	600	600	600	600	0	0	0	0	0
Client 6	750	750	750	750	0	0	0	0	0	0
Client 7	1000	1000	1000	0	0	0	0	0	0	0
Client 8	1500	1500	0	0	0	0	0	0	0	0
Client 9	3000	0	0	0	0	0	0	0	0	0

Table 4.3: Class Label Imbalance training dataset distribution across clients for CIFAR10 dataset.

Dataset	Number of Samples
MNIST	6000
FA-MNIST	6000
CIFAR10	5000
SST2	5000

Table 4.4: Number of samples in the validation dataset.

Dataset	Number of Samples
MNIST	10000
FA-MNIST	10000
CIFAR10	10000
SST2	13469

Table 4.5: Number of samples in the test dataset.

The choice of dataset partitioning strategies was designed to evaluate the frameworks under different levels and sources of client heterogeneity.

In the IID setting, all clients receive the same total number of samples and the same number of samples per class, therefore, the local datasets are not expected to induce a systematic performance ranking among clients. When training in standalone mode, clients converge to models with similar, and in some cases identical, test accuracies. Any remaining differences are expected to come primarily from the stochastic nature of training rather than from deterministic differences in the local data distributions. After federated training, the same behavior is expected: clients should obtain similar final test accuracies, leading to low GMD values, being ideally 0. In this setting, correlation-based fairness metrics may be close to zero, unstable across runs, or even undefined when all clients obtain identical accuracies.

In the POW setting, clients receive different numbers of local training samples. Since no data augmentation is applied before or during training, clients with larger local datasets are expected to achieve higher test accuracy in standalone mode, while clients with fewer samples are expected to achieve lower test accuracy. Based on the existing literature on contribution fairness [20, 7, 14, 21, 22, 23, 24, 25, 26], after federated training it is expected that clients that performed better in standalone mode also achieve higher federated test accuracy than clients that had inferior performance in standalone mode. Therefore, when this condition is met, the Pearson's r Correlation Coefficient between these 2 sets of measure should yield a high and positive value.

The same reasoning applies to the CLA setting, but the source of heterogeneity is class-label diversity rather than dataset size. In this setting, all clients receive the same number of samples, but they do not receive the same coverage of the target task. Clients whose local datasets contain more distinct classes are expected to obtain higher standalone test accuracy because their training data provides supervised signal for a larger portion of the global label space. As a result, these clients learn a broader set of class-discriminative features and decision boundaries that are more aligned with the global test distribution, which encompasses all the classes of each dataset. In contrast, clients with fewer local classes are trained on a narrower classification problem: they receive no direct supervision for the missing labels and therefore misclassify test samples from classes that were absent during local training. Therefore, the expected contribution ordering is induced by class coverage rather than by data quantity. Since low-diversity clients face a stronger mismatch between their local training distribution and the global test distribution, the CLA setting is expected to produce larger performance gaps than the POW setting after federated training.

4.1.4 Neural Architectures

In this subsection, I define the model architectures that were used for each dataset classification task. For each benchmark dataset described in Section 4.1.3, a distinct model architecture was selected based on the experimental setup reported in the corresponding related work or, when available, its public code repository.

- **MNIST:** The model is a Multilayer Perceptron (MLP) of three fully connected layers. The input layer receives a flattened 28×28 image as a 784-dimensional vector. The first hidden layer applies a linear transformation from 784 to 200 dimensions, and the second hidden layer applies a second linear transformation from 200 to 200 dimensions. Both hidden layers are followed by Rectified Linear Unit (ReLu) activations, which introduce nonlinearity into the model, and dropout regularization with rate 0.2, which helps reduce overfitting by randomly deactivating a fraction of the hidden units during training. The final output layer maps the 200-dimensional hidden representation to 10 output units, corresponding to the 10 target classes. This is the same model utilized in McMahan *et al.* [1]. This model has a total of $199,210$ trainable parameters.

- **FA-MNIST:** The model is a Convolutional Neural Network (CNN) composed of 2 convolution layers followed by two fully connected layers. The first convolution layer uses 16 filter of size 3×3 with padding 1, and the second convolution layer uses 32 filters of the same size. Each convolution layer is followed by a ReLu activation and a 2×2 max-pooling operation. After the second pooling layer, the resulting feature maps are flattened and passed to a fully connected layer with 128 hidden units, followed by a final fully connected output layer that maps this representation to 10 output units, corresponding to the 10 target classes. This is the same model utilized in Nguyen *et al.* [31]. This model has a total of $206,922$ trainable parameters.

- **CIFAR10:** The model is a CNN composed of six convolution layers followed by two fully connected layers. The convolutional layers are organized into 3 blocks, each containing 2 convolutional layers with 3×3kernels and padding 1. The first block uses 32 filters in both convolutional layers, the second block uses 64 filters, and the third block uses 128 filters. Each convolutional layer is followed by a ReLu activation. At the end of each block, a 2×2 max-pooling operation is applied, followed by dropout rates of 0.3, 0.5, and 0.5, respectively. After the third pooling layer, the resulting feature maps are flattened and passed to a fully connected layer with 128 hidden units and ReLu activation, followed by an additional dropout layer with rate 0.5.

The final fully connected output layer then maps this representation to 10 output units, corresponding to the 10 target classes. This is a modified version of the CNN utilized in Xu *et al.* [14]. This model has a total of $550,570$ trainable parameters.

- **SST2:** The model is a CNN-based text classifier composed of an embedding layer, 3 parallel convolutional layers, and a final fully connected output layer. The embedding layer maps each input token to a 300-dimensional dense vector, using a vocabulary of 20,000 tokens. The convolutional stage applies 128 filters for each of 3 kernel sizes, namely 3, 4, and 5, allowing the model to capture local semantic patterns corresponding to different n-gram lengths. Each convolution is followed by a ReLU activation and a max-pooling operation over the temporal dimension. The pooled feature vectors from all 3 convolutional branches are then concatenated, passed through a dropout layer with rate 0.5, and fed to a final fully connected output layer with 2 units, corresponding to the 2 target classes of the SST2 task. This is the same model utilized in Xu *et al.* [14] for a similar task. This model has a total of $6,461,954$ model parameters.

For the image classification tasks on MNIST, Fashion-MNIST, and CIFAR10, I used the SGD optimizer, with momentum $= 0.9$. For the SST2 task, I used the Adam optimizer with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rates for both optimizers are reported in Subsection 4.1.5.

4.1.5 Training Hyperparameters

In this subsection, I present the hyperparameters used for standalone training and for the FL simulations conducted with ACGSV and CFFL, which summarized in Table 4.6. Although ACGSV and CFFL include framework-specific hyperparameters, I report them in the same table and identify in the caption which parameters apply to each framework. I also provide a brief explanation of how each hyperparameter affects the corresponding framework, however, I refer the reader to Sections 3.5 and 3.6 for a more detailed review of how these parameters are used in each workflow. Unless explicitly stated otherwise, all remaining hyperparameters are shared across both frameworks.

Dataset	η	γ	B	E	T	α	β	Γ	θ
MNIST	0.015	0.977	32	5	30	0.95	1.0	0.5	5
FA-MNIST	0.005	0.977	64	5	40	0.95	1.0	0.5	5
CIFAR10	0.02	0.977	64	5	100	0.95	1.0	0.5	5
SST2	$1e^{-4}$	0.977	256	5	50	0.95	1.0	1.0	5

Table 4.6: Training hyperparameters used in the simulations with the ACGSV and CFFL frameworks. These values were applied to the IID, POW, and CLA partitioning scenarios. The parameters α, β, and Γ are exclusive to ACGSV, while θ is exclusive to CFFL.

In Table 4.6, η denotes the learning rate used by the optimizer. The parameter γ denotes the exponential learning-rate decay factor applied during FL training, so that the learning rate is reduced across communication rounds. The parameter B denotes the mini-batch size, which determines how many training samples are processed before each optimizer update. The parameter E denotes the number of local epochs executed by each client, corresponding to the number of full passes over the client's local dataset during each communication round. Finally, T denotes the total number of FL communication rounds.

ACGSV has three framework-specific hyperparameters: α, β, and Γ. The parameter α controls the smoothing factor used in the moving-average update of the client importance coefficient. The parameter β is the altruism degree and takes values in the range $[1, \infty)$. Larger values of β reduce the sparsity imposed on clients with smaller importance coefficients, thereby increasing the density of their reward update vector $\vec{\nu}_{i,t}$. Therefore, increasing β introduces a trade-off between strict contribution-proportional reward allocation and a more egalitarian distribution of rewards, since lower-contribution clients receive less sparse updates. The parameter Γ is the scaling factor used during the L2-normalization of each client's accumulated gradient before the server computes the cosine-similarity-based contribution score.

CFFL has one framework-specific hyperparameter, denoted by θ, which corresponds to the punishment factor in the original formulation. Since CFFL uses validation accuracy as its data utility metric, θ controls the nonlinear amplification applied to the validation-accuracy-based utility values. Larger values of θ increase the separation between clients with different validation accuracies, which makes the resulting contribution estimates more sensitive to differences in reconstructed model performance.

For standalone training, I used the same values of η, B, and γ reported in Table 4.6. Unlike FL training, which combines communication rounds with local epochs, standalone training is defined only in terms of epochs. Since there is no standard conversion rule that

maps FL communication rounds and local epochs into an equivalent number of standalone training epochs, I followed the convention adopted in related work: each client is trained in standalone mode for the same number of epochs as the number of FL communication rounds [14, 23, 24, 10, 21, 22].

The hyperparameter values used in these experiments were not selected arbitrarily. I adopted the values reported in the original experiments of Xu *et al.* [14] and Lyu *et al.* [23] for the corresponding datasets. Since ACGSV was not evaluated on Fashion-MNIST in the original paper, I used the hyperparameters reported by related work that adopted the same model architecture for this dataset. Similarly, among the datasets considered in this thesis, CFFL was originally evaluated only on MNIST. For the remaining datasets, I used the corresponding ACGSV hyperparameter values as the default experimental configuration. I also kept the framework-specific hyperparameters fixed across experiments to avoid introducing tuning bias and to ensure that the comparison focuses on the behavior of the frameworks rather than on dataset-specific hyperparameter optimization.

4.1.6 Experiment Assumptions

The experiments conducted in this section are based on the following general assumptions:

1. **Client data sensitivity:** Each client's local dataset is assumed to be sensitive and must not be exposed to any other node in the federation, including the server and other clients. As in standard federated learning, clients report how many samples exist in their local datasets.

2. **Static local datasets:** The local dataset held by each client is assumed to remain unchanged throughout the federated training process.

3. **Cross-silo federated learning:** The federation is assumed to consist of a limited number of clients, such that client sampling is not required. All clients are assumed to have stable and reliable network connections to the server, as well as sufficient computational resources to perform local model training. Once a client joins the federation, it is assumed to dedicate its available computational resources to the learning task.

4. **Non-Byzantine clients:** Clients are assumed to behave honestly during training. In particular, they do not attempt to poison the training process or the global model, do not free-ride on the contributions of other clients, and do not attempt to infer or recover private information from other clients' data.

For the CFFL, there are 2 additional assumptions:

1. **Validation dataset availability:** CFFL assumes that the server has access to a validation dataset that can be used to evaluate each reconstructed client model. This assumption is necessary because CFFL estimates contribution through validation accuracy, therefore, without a validation dataset, the server cannot compute the utility values required to derive client-specific rewards.

2. **Honest reporting of dataset statistics:** CFFL assumes that each client truthfully reports the number of local training samples and the number of distinct classes in its dataset. The server uses these statistics to characterize the federation as either data-quantity-skewed or class-label-skewed. The federation dataset statistics are also shared across the federation.

Since the experimental scenarios do not combine data-quantity skew and class-label skew simultaneously, the CFFL implementation treats each partitioning strategy as a distinct case. Therefore, when the federation is data-quantity-skewed, the implementation applies the equations associated with quantity imbalance, whereas when the federation is class-label-skewed, it applies the equations associated with class-label imbalance.

4.1.7 FL Architecture Implementation

The experiments conducted in section of the thesis were executed on a server equipped with an AMD Ryzen 9 7950X processor, featuring 16 cores, and 64 GB of main memory. The federated learning setting was implemented using `mpi4py` 4.1.0 [70], in which each FL node (i.e., client or server) was represented by an independent MPI process identified by a unique rank and through CPU pinning, each process was bound to a distinct physical core, allowing clients to conduct training in parallel while reducing computational resource contention. `PyTorch` 2.8.0 [71] was used for local model training on the clients and for tensor-related operations on the server.

For inter-process communication, tensors were converted to NumPy arrays using NumPy 2.4.1, since `mpi4py` does not support inter-process communication of tensors, and is optimized for NumPy array-based communication. To perform dataset partitioning, the `flwr-datasets` 0.6.0 library [72] was used, as it provides several built-in dataset partitioning methods, including, but not limited to the IID partitioner, power-law partitioner and the class label imbalance partitioner that were used in the experiments.

4.2 Experiments results: CFFL framework

The experiments conducted with CFFL show that the framework produced the expected fairness outcomes for the IID, POW, and CLA data partitioning scenarios that were previously discussed in Subsection 4.1.3.

Table 4.7 reports the contribution fairness values obtained by CFFL, measured as the Pearson's r correlation between clients' final standalone test accuracies and their final federated test accuracies. The table includes both the per-run, denoted as r_i, where i identifies each of the five runs, and the average Pearson's r correlation coefficient values, denoted as "Mean r" and computed according to the procedure described in Subsection 4.1.1. Averages are computed only over valid runs and when the correlation is undefined, the corresponding coefficient is reported as **-**.

Runs	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.966	0.989	-0.201	0.986	0.961	0.400	0.952	0.915	0.241	0.977
r_2	-	0.974	0.987	-0.346	0.966	0.974	0.204	0.974	0.857	0.385	0.964
r_3	-	0.961	0.987	-0.433	0.979	0.989	0.134	0.939	0.917	-0.114	0.957
r_4	-	0.981	0.986	-0.326	0.949	0.964	-0.298	0.955	0.887	-0.258	0.940
r_5	-	0.953	0.985	0.480	0.949	0.991	-0.348	0.846	0.923	0.313	0.964
Mean r	-	0.969	0.987	-0.167	0.970	0.980	0.019	0.944	0.902	0.119	0.962
Mean GMD	0.00	1.26	33.4	0.02	1.77	29.4	0.01	8.31	12.9	0.00	5.00

Table 4.7: Per-run Pearson's r correlation coefficient obtained with the CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

In the IID setting, clients converge to models with identical or nearly identical test accuracies. Consequently, Pearson's r correlation is either undefined, reported as **-**, or close to zero with variation across runs generated by the stochastic nature of the training process. Since there is no clear standalone performance structure to preserve, Pearson's r correlation is not expected to be consistently high, it may become undefined when one of the accuracy distributions has no variance, or close to zero when client accuracies remain nearly uniform.

In the POW setting, clients with more local samples achieve higher standalone test accuracy, and after training with CFFL they also obtain higher federated test accuracy than clients with fewer samples. In the CLA setting, the same pattern appears with respect to class-label diversity: clients with more local classes perform better in standalone mode and continue to perform better after federated training.

Figures 4.1, 4.2, 4.3, and 4.4 present the results obtained with the CFFL framework across the evaluated datasets and partitioning strategies, and each figure contains two charts. The charts in the left column are line plots showing each client's mean FL test accuracy over the five runs, the x-axis represents the FL training round, the y-axis represents the test accuracy.

In the scatter plots, the x-axis represents the final test accuracy in standalone mode, while the y-axis represents the final test accuracy after federated training with CFFL, each marker represents one client in one experimental run, and the marker shape identifies the client ID. In the scatter plots, the identity line was included as a visual reference to indicate whether each client obtained an accuracy gain or loss after federated training relative to its standalone baseline. The legend uses colored circles to distinguish the experimental runs. These symbols are not the same as the circular marker used to identify a specific client.

Points above the line correspond to clients that obtained test accuracy gain, while points below the line correspond to clients that had test accuracy loss after federated training. This line is included only to contextualize the client-level accuracy changes, while the main focus of the analysis remains the evaluation of contribution fairness. Nevertheless, this identity line might not always be entirely visible, such as in the CIFAR10 POW data partitioning case in Figure 4.3.

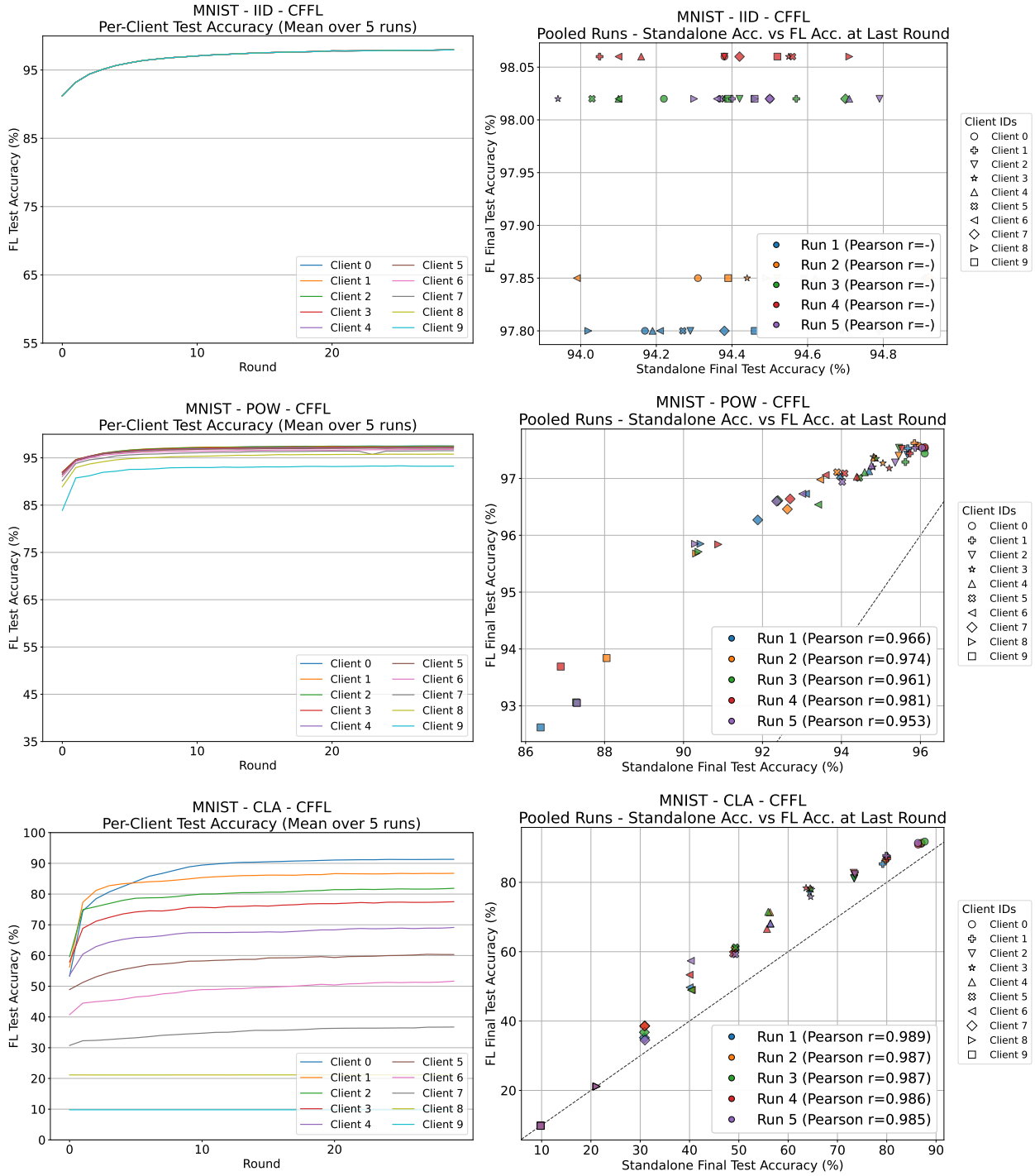


Figure 4.1: Per-client test accuracy results for the CFFL framework on MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

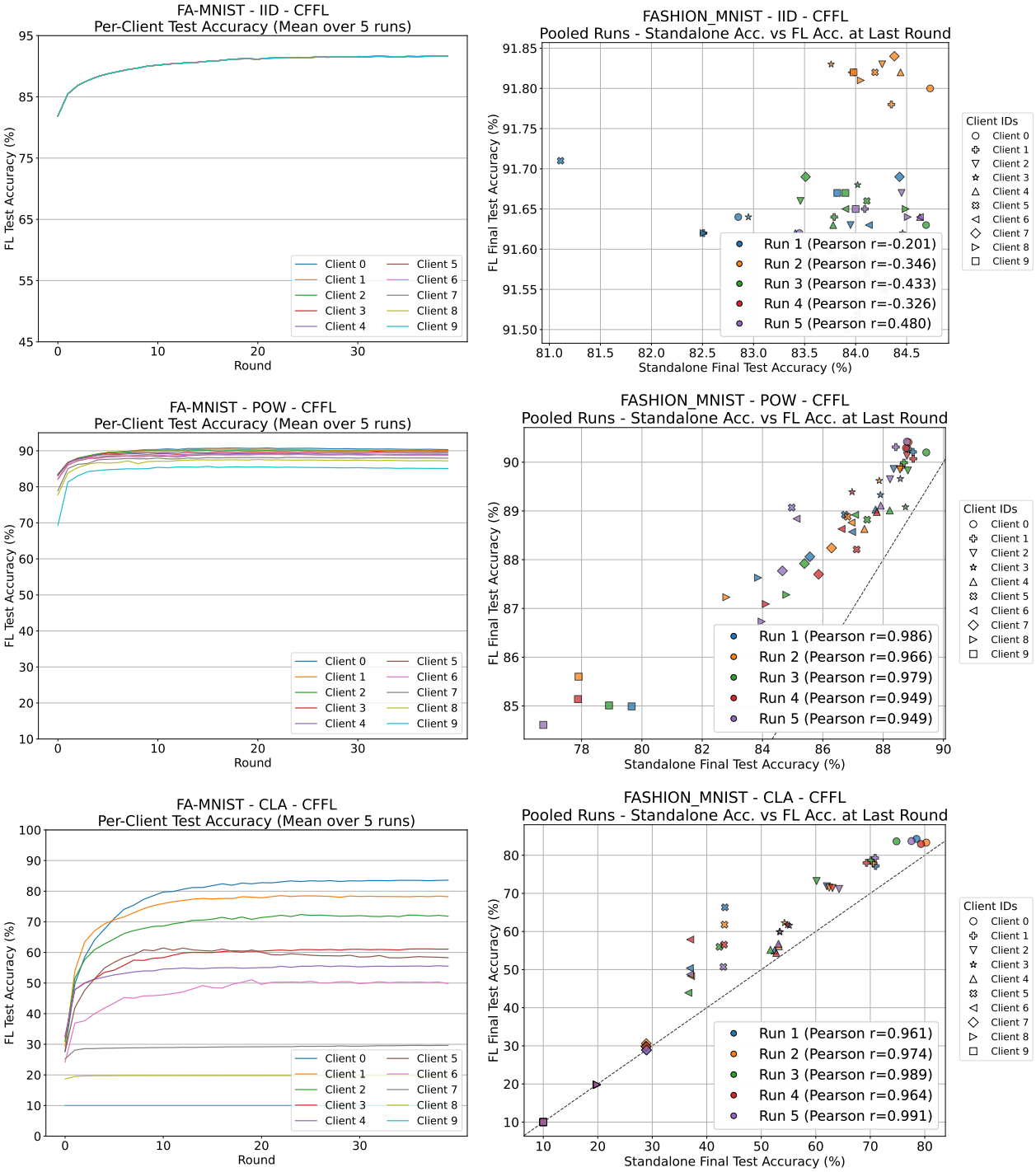


Figure 4.2: Per-client test accuracy results for the CFFL framework on FA-MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

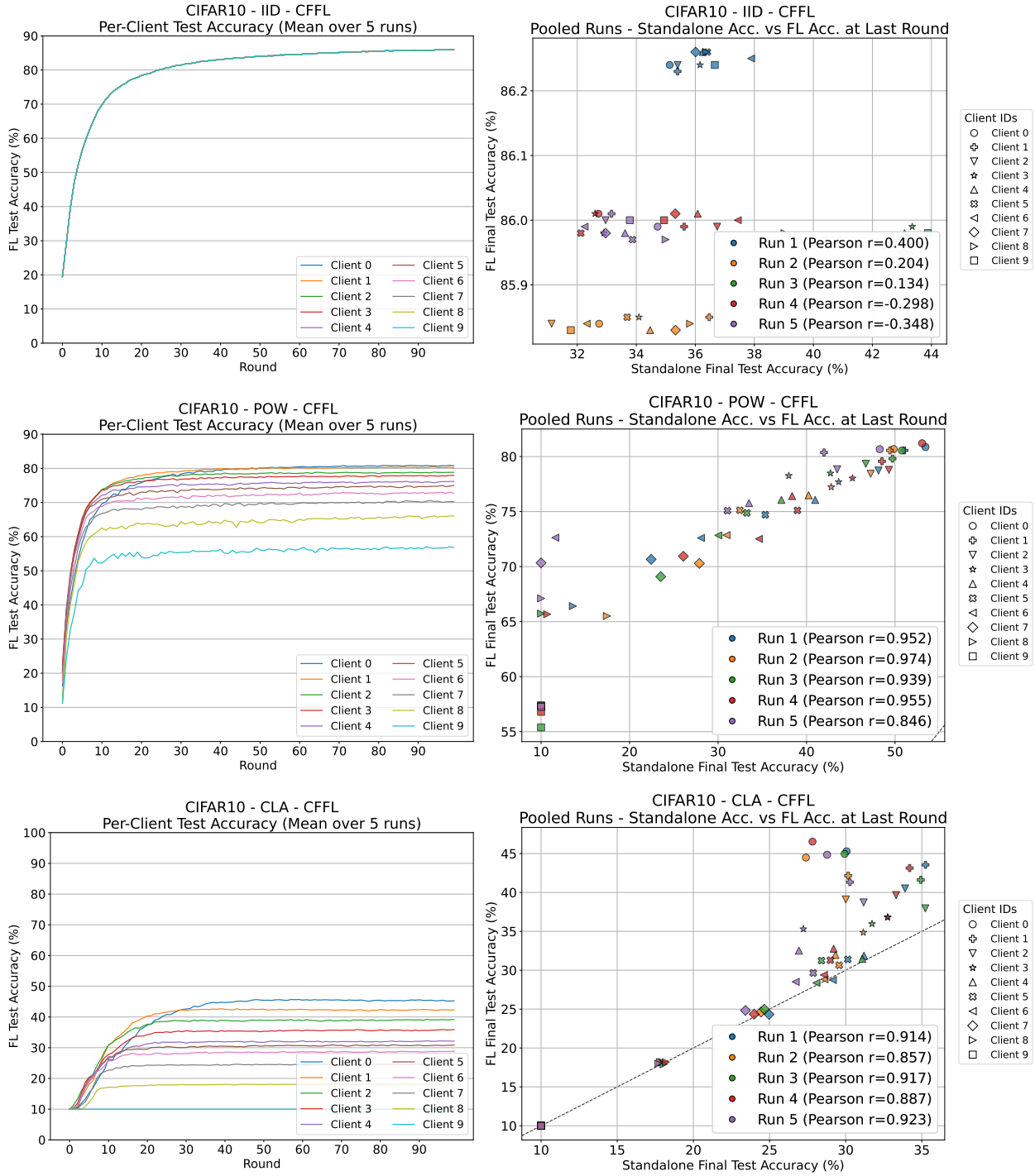


Figure 4.3: Per-client test accuracy results for the CFFL framework on CIFAR10 under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

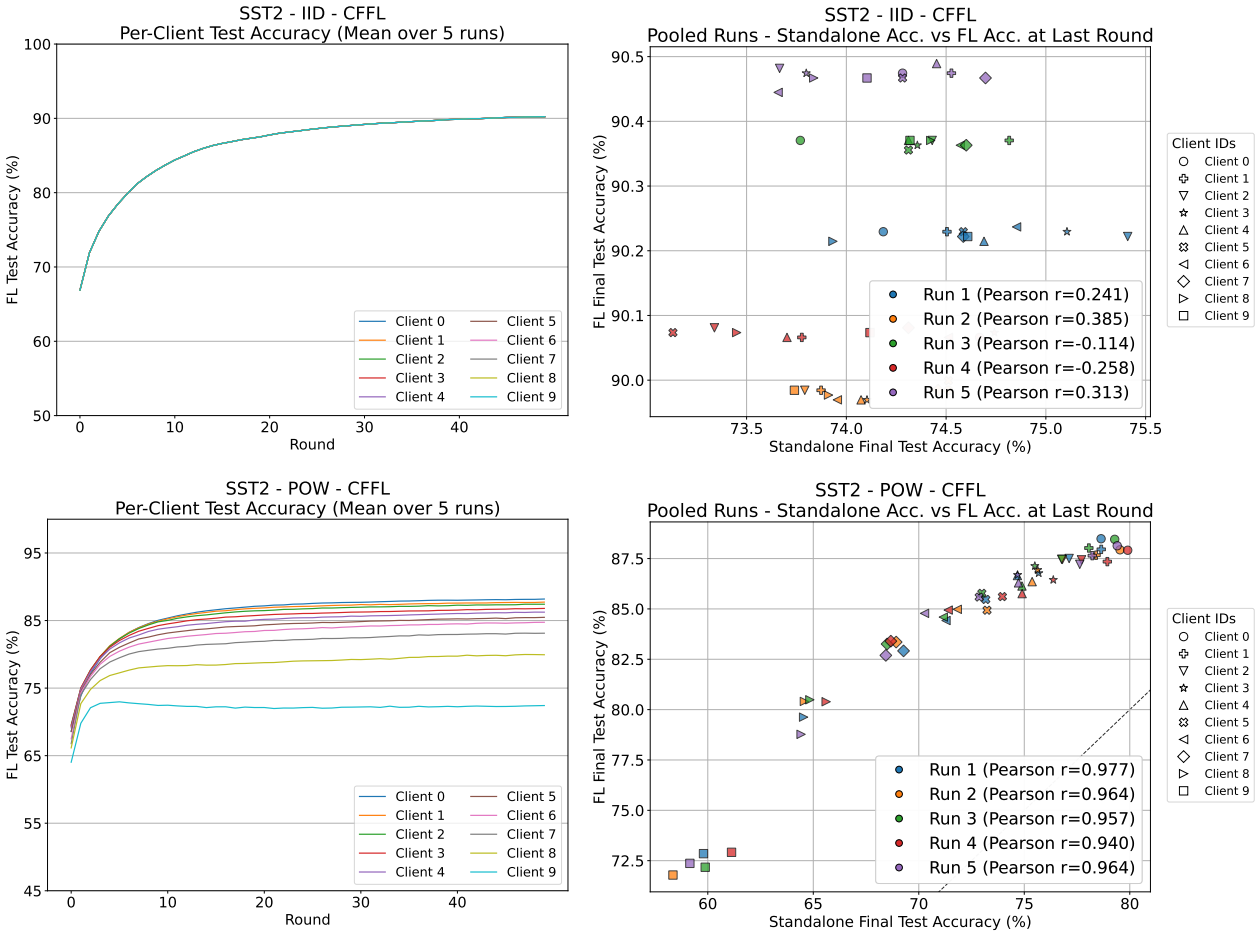


Figure 4.4: Per-client test accuracy results for the CFFL framework on SST2 under IID and POW data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

4.3 Experiment Results: ACGSV Framework

The ACGSV results differ substantially from the CFFL results in the non-IID settings reported in Section 4.2; however, this difference is not evident by direct inspection of the per-run Pearson's r values alone.

Table 4.8 reports the contribution fairness values obtained by ACGSV, measured as the Pearson's r correlation coefficient between clients' final standalone test accuracies and their final federated test accuracies. The table includes both the per-run, denoted as r_i, where i identifies each of the five runs, the average Pearson's r values, and the mean GMD, computed according to the procedures described in Subsection 4.1.1.

Runs	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	0.479	0.988	0.718	-	0.817	0.796	0.179	0.732	0.826	-	0.751
r_2	-0.267	0.955	0.715	0.358	0.816	0.751	-	0.778	0.864	-	0.734
r_3	-0.483	0.964	0.784	-0.088	0.922	0.809	-	0.673	0.814	-	0.738
r_4	-0.438	0.933	0.772	-	0.912	0.791	-	0.717	0.809	-	0.691
r_5	-0.746	0.920	0.741	0.231	0.910	0.819	0.292	0.622	0.852	-	0.765
Mean r	-0.329	0.959	0.747	0.172	0.884	0.795	0.159	0.709	0.834	-	0.737
Mean GMD	0.00	0.11	1.16	0.00	0.09	5.25	0.01	0.26	4.44	0.0	0.08

Table 4.8: Per-run Pearson's r correlation coefficient obtained with the ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

In the POW and CLA settings, Pearson's r exceeds 0.7 in most runs, indicating that ACGSV often obtains a strong contribution fairness [14, 21, 58]. However, these Pearson values provide only a partial view of the experimental outcome. If CFFL and ACGSV are compared only through the average Pearson's r values in Table 4.9, as in related work [7, 14, 21, 22, 23, 30, 41], the two frameworks appear to achieve comparable contribution fairness, with ACGSV achieving lower fairness in the selected dataset-partitioning settings.

Framework	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
CFFL	-	0.969	0.987	-0.167	0.970	0.980	0.019	0.944	0.902	0.119	0.962
ACGSV	-0.329	0.959	0.747	0.172	0.884	0.795	0.159	0.709	0.834	-	0.737

Table 4.9: Mean Pearson's r correlation coefficients comparison between ACGSV and CFFL framework in IID, POW and CLA data. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. **-** values represent the correlation value is undefined or there is no correlation.

A Pearson-only comparison obscures the evaluation of contribution fairness because it does not separate two distinct and important aspects of this fairness notion: preservation

of the contribution ranks and the magnitude of client-level model differentiation. This is not due to an incorrect use of Pearson's r, but to the fact that Pearson's r was designed to measure linear association between paired values, not rank-order preservation or the spread of the federated test accuracies.

By computing GMD over the final federated test accuracies, the comparison between CFFL and ACGSV directly quantifies the average pairwise difference in client model performance produced by each framework, which is summarized in Table 4.10. Across the same datasets and partitioning strategies, ACGSV consistently obtains substantially lower GMD values than CFFL.

Framework	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
CFFL	0.00	1.26	33.4	0.02	1.77	29.44	0.013	8.31	12.95	0.00	5.00
ACGSV	0.00	0.11	1.16	0.00	0.09	5.25	0.01	0.26	4.44	0.0	0.08

Table 4.10: Mean GMD comparison between CFFL and ACGSV under IID, POW, and CLA data partitioning. Values are reported in percentage points and measure the average pairwise difference between final client test accuracies.

Although a strong linear relationship can occur even when the data points are numerically close to one another, in this context a mean GMD close to zero shows that the final federated test accuracies are highly concentrated. In other words, although ACGSV achieved high Pearson's r correlation values, its low GMD values indicate that the clients often converge to models with very similar final test accuracies.

The scatter plots shown in Figures 4.5, 4.6, 4.7, and 4.8 obtained from the experiments with the ACGSV framework further emphasizes the limitation of using Pearson's r Correlation Coefficient as the only metric for evaluating contribution fairness, which was not visible in the scatter plots shown in Figures 4.1, 4.2, 4.3, and 4.4 for the CFFL framework.

This limitation is particularly visible in the MNIST CLA, FA-MNIST POW, CIFAR10 POW, CIFAR10 CLA, and SST2 POW settings, in Figures 4.5, 4.6, 4.7, and 4.8. In this settings, most clients converge to models with very similar test accuracy, forming a cluster of data points, and only a small subset of clients remains outside this cluster with substantially lower test accuracy.

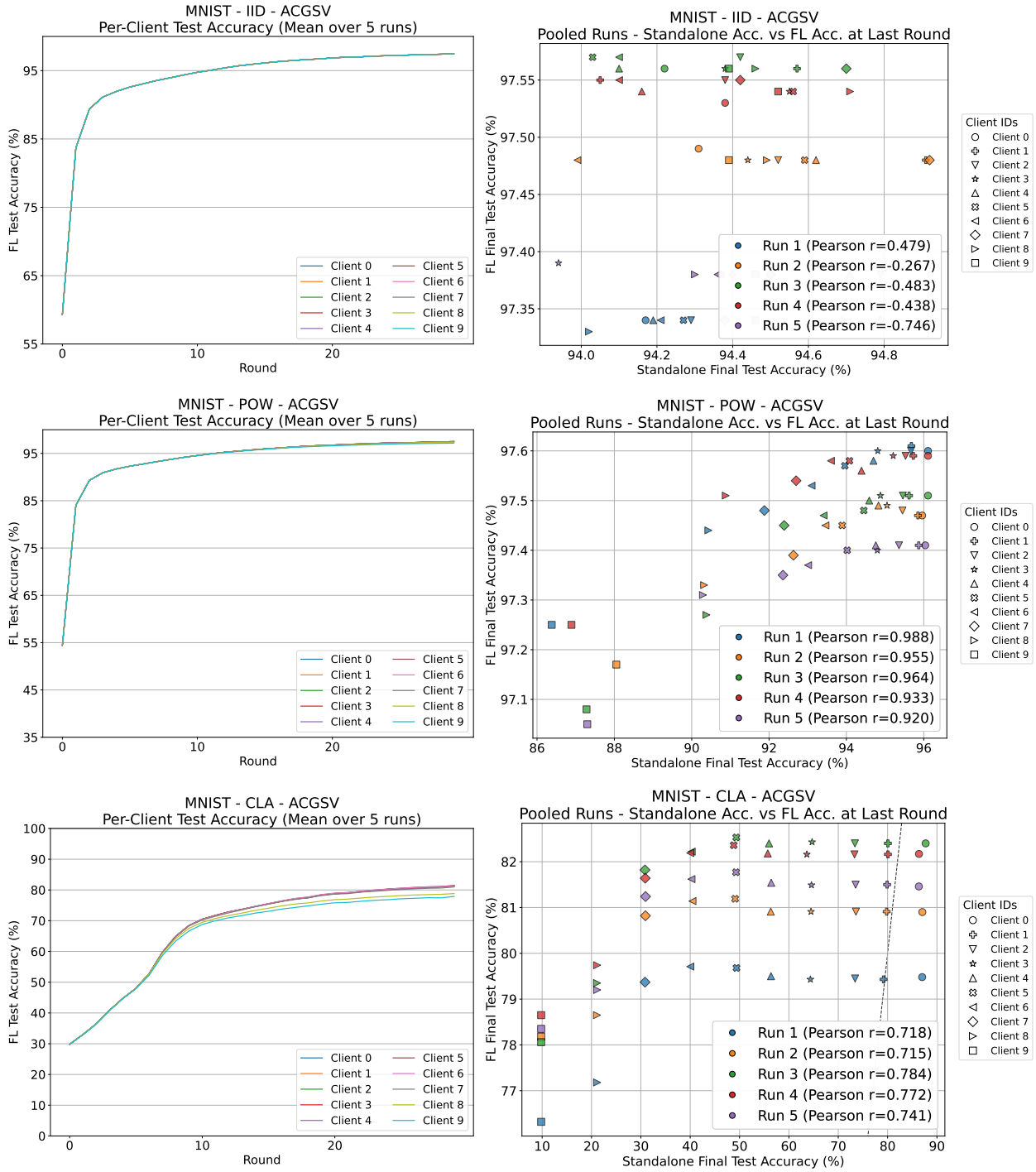


Figure 4.5: Per-client test accuracy results for the ACGSV framework on MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

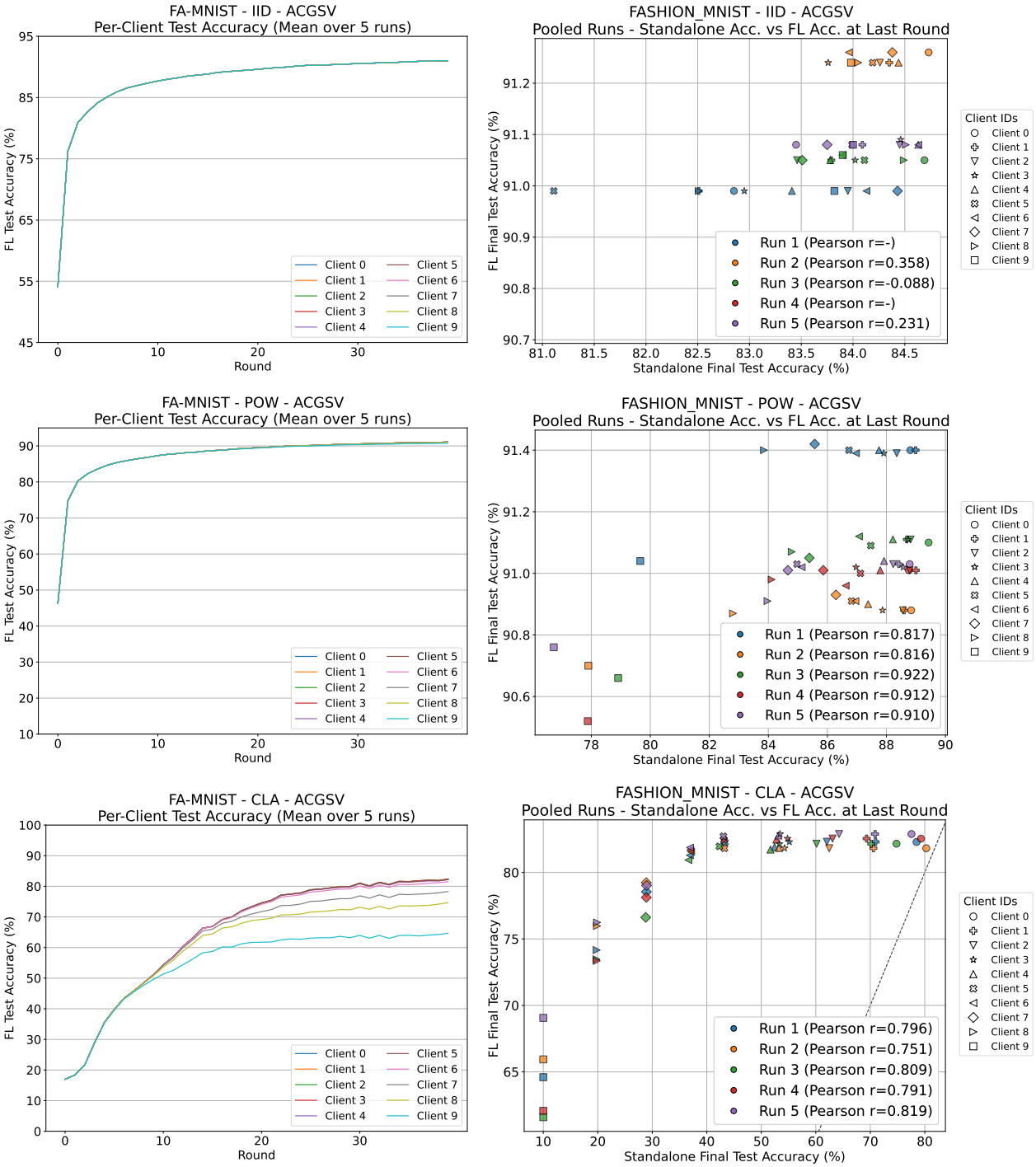


Figure 4.6: Per-client test accuracy results for the ACGSV framework on FA-MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

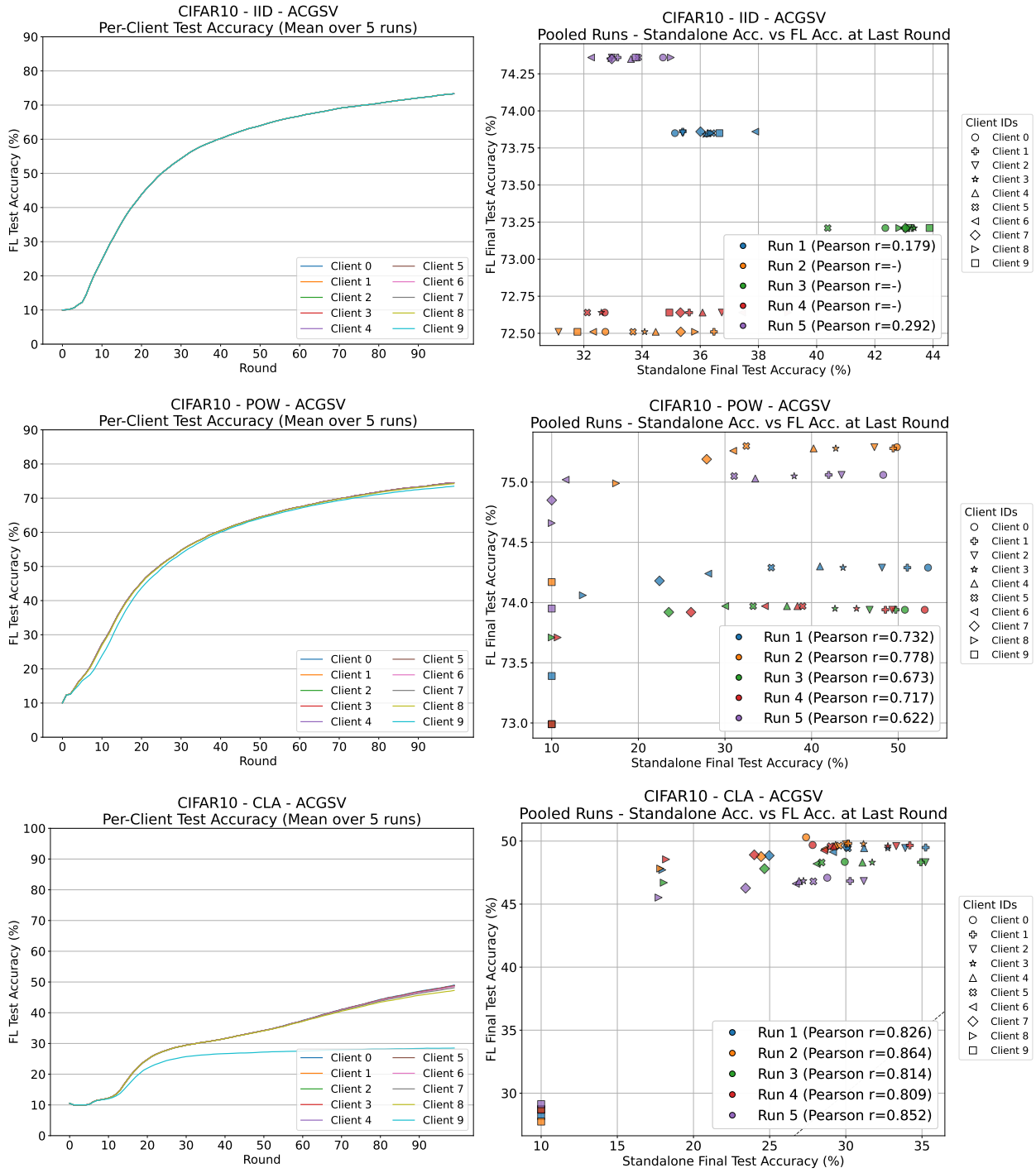


Figure 4.7: Per-client test accuracy results for the ACGSV framework on CIFAR10 under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

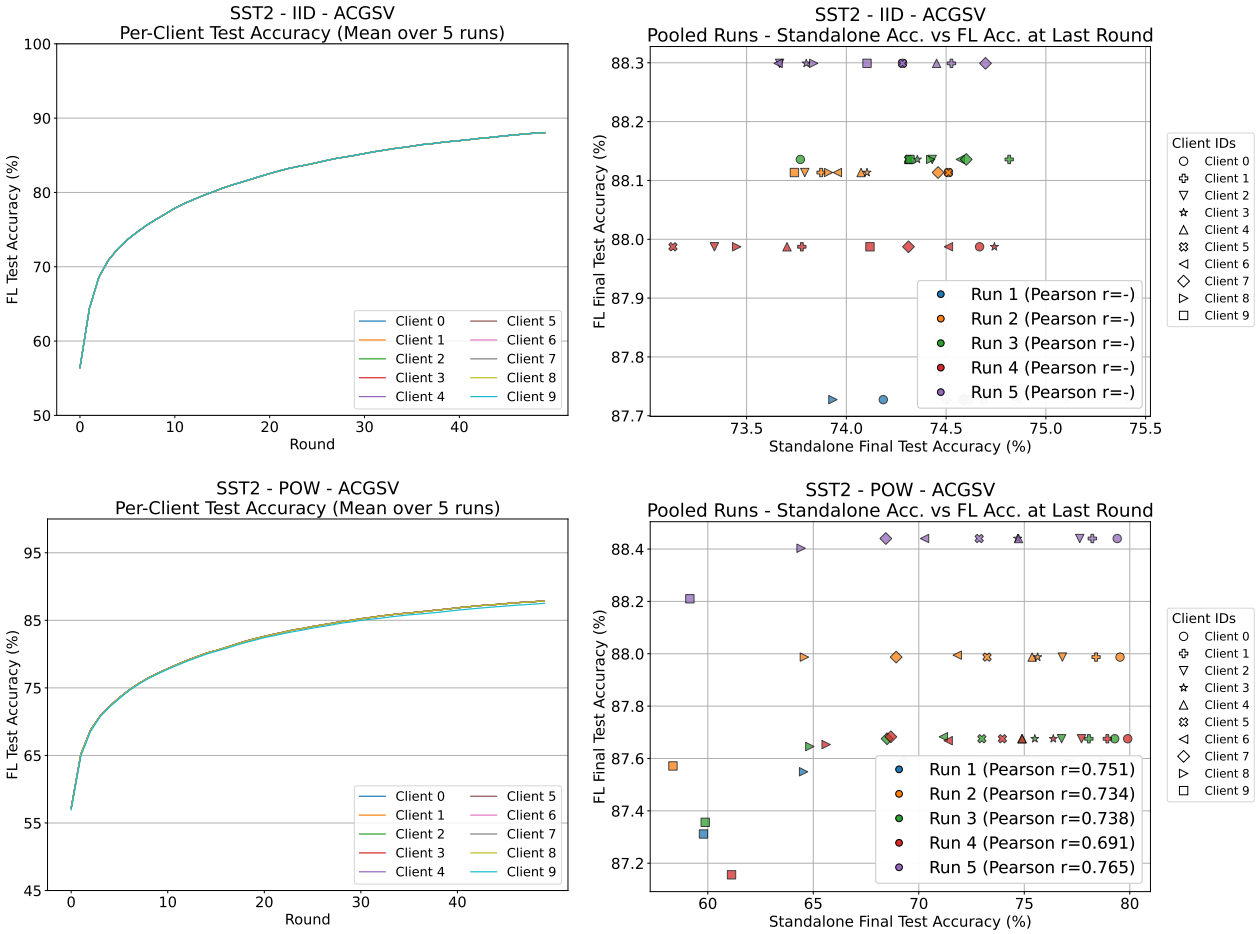


Figure 4.8: Per-client test accuracy results for the ACGSV framework on SST2 under IID and POW data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

4.4 Experiments Summary

The experiments presented in this chapter show that CFFL and ACGSV[1], despite sharing the common objective of achieving contribution fairness, based in Definition 1, obtain different fairness outcomes.

However, since these two frameworks have multiple differences in their internal mechanisms, it's not possible to determine by direct inspection which component is responsible

[1]Appendix C contain the experiments conducted with the ACGSV in which the training had more training rounds. The purpose was to verify whether increasing the number of communication rounds would change the observed behavior. However, the results reported no changes in the conclusions obtained from the main experiments.

for the different outcomes. In fact, several framework-specific components could plausibly explain the observed behavior. For example, CFFL constrains the density of the Top-K reward using dataset-dependent statistics, such as the number of local samples or the number of local classes, which creates stratified degrees of reward sparsity and may directly impact in model differentiation and reference contribution rank preservation. In addition, clients in CFFL retain portions of their locally accumulated gradients, whereas clients in ACGSV do not keep their locally accumulated gradients. These observations motivate the ablation studies presented in Chapter 5, where I isolate the relevant design choices and identify which components are primarily responsible for achieving contribution fairness, measure through the Pearson's r correlation coefficient.

Additionally, the experiments conducted in this chapter reveal a limitation in the evaluation protocol adopted in FL contribution fairness literature [7, 14, 21, 22, 25, 23, 39, 40, 41, 42]. Pearson's r, computed according to Definition 3, measures the linear association between standalone and federated client test accuracy values, but a high correlation value may be reported even when most clients converge to models that achieve similar, or even the same, final test accuracies and there is only a small subset of lower-performing clients.

This limitation of the Pearson's r correlation coefficient has not been explicitly addressed in the related FL literature reviewed in this thesis. Therefore, I argue that contribution fairness should not be evaluated solely through Pearson's r. Instead, contribution fairness should be evaluated by Kendall's τ_b and GMD.

Together, these two metrics provide a more complete evaluation of contribution fairness by measuring the reference contribution rank preservation and model differentiation, which are two aspects that Pearson's r cannot capture because it was not designed to measure either. Kendall's τ_b, computed on the same paired client-level measurements as Pearson's r, evaluates whether the set of ranks formed by the FL test accuracy values obtained by each client has a pairwise agreement with the reference contribution ranks observed in standalone training. The GMD, in turn, quantifies the average pairwise difference among the final federated test accuracies, measuring the magnitude of client-level model differentiation. The detailed rationale that supports this evaluation protocol is presented in Section 4.5.

In this chapter, standalone test accuracy is used as the contribution reference value to remain consistent with the evaluation protocol adopted in related work. However, the same conclusion is reinforced in Section 4.6, where Shapley Values are used as the contribution reference. The Shapley Values-based analysis confirms that the limitation is not specific to the standalone proxy; rather, it reflects a broader weakness of relying on Pearson's r alone to evaluate contribution fairness.

4.5 New Contribution Fairness Evaluation Protocol: Kendall's τ and GMD

Chapter 2, through Definition 1, introduced contribution fairness as the principle that each client should receive a reward that is commensurate with its contribution to the federated learning training. However, evaluating contribution fairness by correlating the estimated contribution values produced by a FL framework and the clients' FL test accuracy creates a circularity problem. Since every FL framework relies on its own FLCE algorithm and, thus, estimates contribution through a different mechanism, (e.g., gradient alignment or validation accuracy), a framework may appear fair merely because its outcome is internally consistent with its own estimator. Therefore, in order to evaluate contribution fairness, FL frameworks require an external set of values to be used as contribution reference.

Although Shapley Values provide the most principled method for estimating a client's marginal contribution, there are two major challenges in using them as contribution reference values. First, their computation grows exponentially with the number of clients, and new FLCE algorithms are commonly evaluated across several federation sizes to assess its scalability [14, 21, 22, 23, 39, 7, 40, 41, 25, 10, 73]. Therefore, computing Shapley Values for every experimental configuration is prohibitively expensive. Second, Shapley Values are defined on a utility function, thus when test accuracy is used as that utility, the resulting contribution values shift with changes in the model architecture, the training hyperparameters, or other implementation details [74].

The common practice in the related work [14, 21, 22, 23, 39, 7, 40, 41, 25, 10, 73], is to use each client's standalone test accuracy as a more tractable external reference for its contribution to training (Definition 3). Therefore, to keep consistent with the existing literature, the argument here is built based on this contribution reference. Nevertheless, to further strengthen the claim Section 4.6 presents the contribution fairness analysis using Shapley Values, which bear the same conclusion being presented in the current section.

The standalone test accuracies establish a pre-FL performance ordering that serves as a valid contribution reference for the analysis: clients with stronger local data are expected to contribute more, and, according to the literature and the results reported in the related work, a contribution-fair framework should preserve this ordering after federated training.

The literature evaluates contribution fairness by computing the Pearson's r correlation between the the standalone and the federated test accuracies, where a high positive r is interpreted as evidence that the framework promotes contribution fairness. However, the ACGSV results showed that a high Pearson's r correlation coefficient can be driven by a small subset of clients whose models achieve a lower test accuracy and remains far from

the cluster of clients that converge to similar final accuracies, as shown in the ACGSV scatter plots in Figures 4.5, 4.6, 4.7, and 4.8. Those distant points dominate the mean-centered products from which Pearson's r is built, so the coefficient reports a strong linear association.

Based on these observations, I propose a new contribution fairness evaluation protocol that replaces Pearson's r correlation coefficient with two metrics: Kendall's τ_b, which evaluates whether the reference contribution rank ordering is preserved after FL training, and the GMD, which quantifies the extent of model differentiation across clients in the federation.

Kendall's τ_b is used to evaluate the reference contribution rank preservation in contribution fairness. The coefficient operates on pairs of clients and examines whether the ordering established by the reference contribution ranks is preserved in the final FL outcomes. For a pair of clients i and j, the pair is concordant when the client with the greater reference contribution also obtains the greater final FL test accuracy, and discordant when this ordering is reversed. This pairwise comparison reflects the ordering requirement of contribution fairness, according to which clients with greater contributions should receive better-performing models than clients with lower contributions. Kendall's τ_b is preferred because it account for ties, thus pairs tied in both the reference and FL rankings do not contribute as discordant, while pairs tied in only one ranking reduce the strength of the observed association. For notational simplicity, Kendall's τ refers throughout the remainder of this thesis to Kendall's τ_b, computed as defined in Equation 29.

However, Kendall's τ captures only the pairwise ordering agreement and does not consider the magnitude of the performance differences that establish this ordering. Therefore, Kendall's τ cannot independently determine whether the preserved ordering corresponds to meaningful differentiation among the client models.

To evaluate this second aspect of contribution fairness, Kendall's τ is interpreted together with the Gini Mean Difference (GMD), defined in Section 2.9. In this context, the GMD is computed from the clients' final FL test accuracies and represents the average absolute performance difference, in percentage points, across all pairs of clients. Therefore, it quantifies the overall extent of model differentiation after federated training. A GMD close to zero indicates that the final client models obtain similar predictive performance, whereas a larger GMD indicates that the framework produces more differentiated client outcomes.

In the proposed evaluation protocol, Kendall's τ and the GMD are interpreted sequentially rather than combined into a single score. Kendall's τ constitutes the primary criterion because contribution fairness first requires the rank ordering of the final FL client test accuracies to be aligned with the reference contribution rank ordering. The GMD is subsequently used to determine whether this aligned ordering is accompanied by non-negligible differences in client performance. Therefore, the GMD does not compensate for the absence of contribution-rank preservation; instead, it verifies whether an observed positive rank association represents meaningful differentiation.

A negative Kendall's τ indicates that clients with higher reference contributions ranks tend to receive worse-performing models than clients with lower reference contributions ranks. This outcome directly contradicts the objective of contribution fairness. Consequently, a framework with a negative Kendall coefficient is not considered contribution-fair, regardless of the magnitude of its GMD.

A Kendall coefficient close to zero indicates that no consistent relationship is established between the reference contribution ordering and the ordering of the final FL client test accuracies. Thus, values of Kendall's τ close to zero do not provide evidence of contribution fairness, regardless of whether the resulting client models are differentiated.

A positive Kendall coefficient indicates that the ordering of the final client-model performances is at least partially aligned with the reference contribution ordering. Within this positive range, values of Kendall's τ closer to 1 indicate stronger preservation of the reference contribution rank. Accordingly, among frameworks that exhibit non-negligible model differentiation, a larger Kendall coefficient provides stronger evidence that the benefits assigned to clients follow their relative contributions. Nevertheless, a positive Kendall coefficient alone is insufficient to establish contribution fairness, because the observed ordering may be determined by small numerical differences among client models with practically similar predictive performance.

The GMD is therefore examined after a positive rank association has been established. Its role is to determine whether the contribution-aligned ordering is accompanied by meaningful performance differentiation. As established in Section 2.9, this thesis considers

$$\text{GMD} < 1$$

percentage point to represent negligible model differentiation. Accordingly, a positive Kendall coefficient accompanied by a non-negligible GMD provides evidence of contribution fairness, because the framework both preserves the reference contribution ordering and produces meaningfully differentiated client outcomes. In contrast, a positive Kendall

coefficient accompanied by a negligible GMD indicates that the clients remain numerically ordered according to contribution, but the resulting models are practically similar in predictive performance.

Under this sequential protocol, contribution fairness is therefore supported only when two conditions are satisfied: the framework must exhibit a positive association with the reference contribution ordering, and the resulting model differentiation must be non-negligible. The first condition establishes whether the learning benefits are allocated in the correct contribution order, while the second establishes whether the differences among those benefits are meaningful. Neither an ordered collection of practically identical client models nor differentiated outcomes that are unrelated to or inversely aligned with contribution is considered contribution-fair.

For the formal comparison of two frameworks, both frameworks must satisfy both the reference contribution rank preservation and the non-negligible GMD requirements. Then, Kendall's τ becomes the primary comparison criterion, and the framework with the larger coefficient is considered to preserve the reference contribution ordering more successfully. If the frameworks obtain equivalent Kendall coefficients, the GMD may be used as a secondary discriminator, with the larger GMD indicating stronger model differentiation under the same degree of rank preservation.

Scatter and convergence plots remain useful for illustrating how the final client outcomes are distributed and how they emerge throughout training. In particular, they may reveal whether a high rank association is supported by visible model separation or instead arises from small numerical differences among nearly converged client models. The proposed protocol does not replace the scatter plots; rather, it provides a complementary and more comprehensive quantitative evaluation of contribution fairness than the Pearson's r, which is particularly useful when space constraints prevent the inclusion of multiple charts.

The per-run Kendall's τ coefficients, mean Kendall's τ, and the mean GMD values obtained across runs for CFFL and ACGSV are reported in Tables 4.11 and 4.12, respectively.

Runs	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	-	0.989	1.000	0.143	0.911	0.911	0.315	1.000	0.778	0.051	0.956
τ_2	-	0.911	1.000	-0.193	0.867	0.956	0.182	1.000	0.600	0.027	0.956
τ_3	-	0.822	1.000	-0.163	0.867	0.956	0.270	0.989	0.778	-0.058	1.000
τ_4	-	0.867	1.000	-0.209	0.733	0.867	-0.248	0.911	0.733	-0.211	0.956
τ_5	-	0.944	1.000	0.238	0.911	1.000	-0.241	0.920	0.778	0.126	0.956
Mean τ	-	0.907	1.000	-0.036	0.857	0.938	0.055	0.964	0.733	-0.013	0.964
Mean GMD	0.00	1.26	33.4	0.02	1.77	29.4	0.01	8.31	12.9	0.00	5.00

Table 4.11: Per-run Kendall's τ correlation coefficients obtained with the CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

Runs	MNIST			FA-MNIST			CIFAR			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	0.447	0.920	0.270	-	0.053	0.764	0.058	0.680	0.764	-	0.304
τ_2	-0.348	0.598	0.263	0.228	-0.097	0.597	-	0.675	0.644	-	0.181
τ_3	-0.395	0.931	0.549	-	0.406	0.920	-	0.193	0.796	-	0.365
τ_4	-0.392	0.822	0.322	-	0.479	0.828	-	0.215	0.705	-	0.435
τ_5	-0.452	0.871	0.225	0.151	0.532	0.931	0.329	0.868	0.690	-	0.615
Mean τ	-0.228	0.828	0.326	0.126	0.275	0.808	0.193	0.526	0.719	-	0.380
Mean GMD	0.00	0.11	1.16	0.00	0.09	5.25	0.01	0.26	4.44	0.0	0.08

Table 4.12: Per-run Kendall's τ correlation coefficients obtained with the ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

Comparing Kendall's τ and the GMD values in Tables 4.11 and 4.12 with the corresponding scatter plots in Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8 shows why these metrics provide a more informative evaluation of Contribution Fairness than Pearson's correlation coefficient alone.

In POW setting, the Kendall's τ value that CFFL obtains, indicates that the framework preserves the client ordering observed in standalone training. This pattern is visible in the CFFL scatter plots in Figures 4.1, 4.2, 4.3, and 4.4: clients with lower standalone test accuracy also obtain lower FL test accuracy. Conversely, clients with stronger standalone performance remain among the better-performing clients after federated training. Therefore, the reference contribution rank established before federation is preserved by CFFL.

In contrast, in the POW setting, ACGSV does not achieve consistently high Kendall's τ correlation values across datasets, except for MNIST, which indicates that the framework does not preserve the reference contribution ordering obtained from the standalone test accuracy. This pattern is visible in ACGSV scatter plots in Figures 4.5, 4.6, 4.7, and 4.8, which show rank inversions happening after clients trained collaboratively through this framework. In FA-MNIST, for example, clients 4, 5, 6, and 7 obtain higher FL test accuracies than clients 1, 2, and 3, even though the latter achieved higher test accuracies in standalone training.

The GMD values in Tables 4.11 and 4.12 further clarify this difference by showing that ACGSV produces substantially weaker model differentiation than CFFL. Thus, even when ACGSV obtains a high Pearson's r, its Kendall's τ and the GMD values reveal that this positive linear association does not necessarily correspond to strong rank preservation and provide neglible model differentiation, therefore, thus showing that ACGSV achieves lower contribution fairness. By contrast, under POW partitioning, CFFL more consistently satisfies both requirements: it preserves the contribution-dependent ordering and produces a higher degree of model differentiation. The same analysis conducted for the POW data partitioning scenario can be extended to the CLA partitioning scenario for both frameworks.

The run-by-run discussion above was included to make the interpretation of Kendall's τ and GMD explicit and to show why their joint use provides a more informative assessment of contribution fairness than Pearson's r alone. However, because run-level comparisons can be difficult to interpret across multiple datasets and frameworks, Tables 4.13 and 4.14 provide a more compact summary of the results. These tables compare ACGSV and CFFL under POW and CLA partitioning by reporting the mean Pearson's r, Kendall's τ, and GMD across five independent runs.

Framework	MNIST			FA-MNIST			CIFAR			SST2		
	r	τ	GMD	r	τ	GMD	r	τ	GMD	r	τ	GMD
ACGSV	0.959	0.828	0.11	0.884	0.275	0.09	0.708	0.526	0.26	0.737	0.380	0.08
CFFL	**0.969**	**0.907**	**1.26**	**0.970**	**0.858**	**1.77**	**0.944**	**0.964**	**8.31**	**0.962**	**0.964**	**4.99**

Table 4.13: Comparison between the ACGSV and CFFL under POW data partitioning, using the average Pearson's r, Kendall's τ, and mean GMD across runs. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values in **bold** font indicate the framework that achieved the highest value.

Framework	MNIST			FA-MNIST			CIFAR		
	r	τ	GMD	r	τ	GMD	r	τ	GMD
ACGSV	0.748	0.326	1.16	0.795	0.808	5.24	0.834	0.720	4.43
CFFL	**0.987**	**1.000**	**33.44**	**0.980**	**0.938**	**29.44**	**0.902**	**0.733**	**12.95**

Table 4.14: Comparison between the ACGSV and CFFL under CLA data partitioning, using the average Pearson's r, Kendall's τ, and mean GMD across runs. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values in **bold** font indicate the framework that achieved the highest value.

For the remainder of this thesis, the analysis focuses on the POW and CLA data partitioning scenarios because these settings provide the most informative conditions for evaluating Contribution Fairness. Unlike the IID setting, where clients are expected to have comparable data distributions and therefore similar learning opportunities, POW and CLA introduce data-quantity and class-label heterogeneity. These forms of heterogeneity create the conditions under which contribution-aware reward mechanisms are expected to produce measurable differences in client outcomes. Therefore, POW and CLA are the primary settings for assessing whether ACGSV and CFFL preserve contribution-based ordering and induce meaningful model differentiation.

4.6 Fairness Analysis with Shapley Values

As discussed in Subsection 4.1.2, the federation size used in this thesis makes Shapley Value computation feasible. Therefore, computing each client's Shapley Value with test accuracy

as the coalition utility yields a coalition-based contribution reference that accounts for a client's marginal contribution across all possible coalitions. This section conducts the fairness analysis using the clients' Shapley Values instead of their standalone test accuracies as the contribution reference.

Tables 4.15 and 4.16 report the mean Pearson's r, Kendall's τ, and GMD values obtained under this Shapley-value-based analysis. The GMD values are the same as those reported in Tables 4.13 and 4.14, because GMD is computed only over the final federated test accuracies and does not depend on the contribution reference used for the correlation analysis.

Framework	MNIST			FA-MNIST			CIFAR			SST2		
	r	τ	GMD	r	τ	GMD	r	τ	GMD	r	τ	GMD
ACGSV	0.949	0.829	0.11	0.777	0.333	0.09	0.834	0.539	0.26	0.657	0.380	0.08
CFFL	**0.958**	**0.915**	**1.26**	**0.983**	**0.938**	**1.77**	**0.991**	**1.000**	**8.31**	**0.922**	**0.982**	**4.99**

Table 4.15: Comparison between the ACGSV and CFFL under POW data partitioning, using the average Pearson's r, Kendall's τ, and mean GMD across runs. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values in **bold** font indicate the framework that achieved the correlation coefficient value.

Framework	MNIST			FA-MNIST			CIFAR		
	r	τ	GMD	r	τ	GMD	r	τ	GMD
ACGSV	0.537	0.326	1.16	0.570	0.808	5.24	0.565	0.900	4.43
CFFL	**0.873**	**1.000**	**33.44**	**0.871**	**0.938**	**29.44**	**0.954**	**0.947**	**12.95**

Table 4.16: Comparison between the ACGSV and CFFL under CLA data partitioning, using the average Pearson's r, Kendall's τ, and mean GMD across runs. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values in **bold** font indicate the framework that achieved the correlation coefficient value.

The Pearson's r and Kendall's τ values differ slightly between the standalone-accuracy and the Shapley-based analyses, which is expected, since one of the two paired variables

was changed.[2] Because the distribution of exact Shapley Values is not identical to that of the standalone accuracies, the correlation values shift accordingly. The effect is most pronounced for Pearson's r, which depends on each observation's deviation from the mean and is therefore sensitive to the spacing of the contribution values; Kendall's τ, operating on ranks, is affected less. Although these shifts lead to minor changes in the magnitude of the correlations, overall, replacing standalone test accuracy with a coalition-based does not alter the conclusions stated in Section 4.5.

The Shapley-based analysis corroborates the standalone-accuracy analysis. CFFL again preserves the contribution ordering more strongly than ACGSV, and Pearson's r again proves unsuitable as a contribution-fairness metric, since it was not designed to measure either rank preservation or model differentiation. The conclusions of this chapter are therefore not an artifact of using standalone test accuracy as the contribution proxy: the limitations of Pearson's r, and the differences between CFFL and ACGSV, persist when contribution is measured through exact coalition-based marginal utility.

Figures 4.9 and 4.10 present scatter plots charts where the x-axis represents the shapley values for each client, while the y-axis represents the final test accuracy after federated training with CFFL. Each marker represents one client in one experimental run, and the marker shape identifies the client ID. In the figures, the scatter plots on the left are respective to the ACGSV framework, while the scatter plot on the right correspond to the CFFL framework. The legend uses colored circles to distinguish the experimental runs. These symbols are not the same as the circular marker used to identify a specific client.

[2]The per-run Pearson's r and Kendall's τ for both frameworks are reported in Tables D.1, D.2, D.5, and D.6.

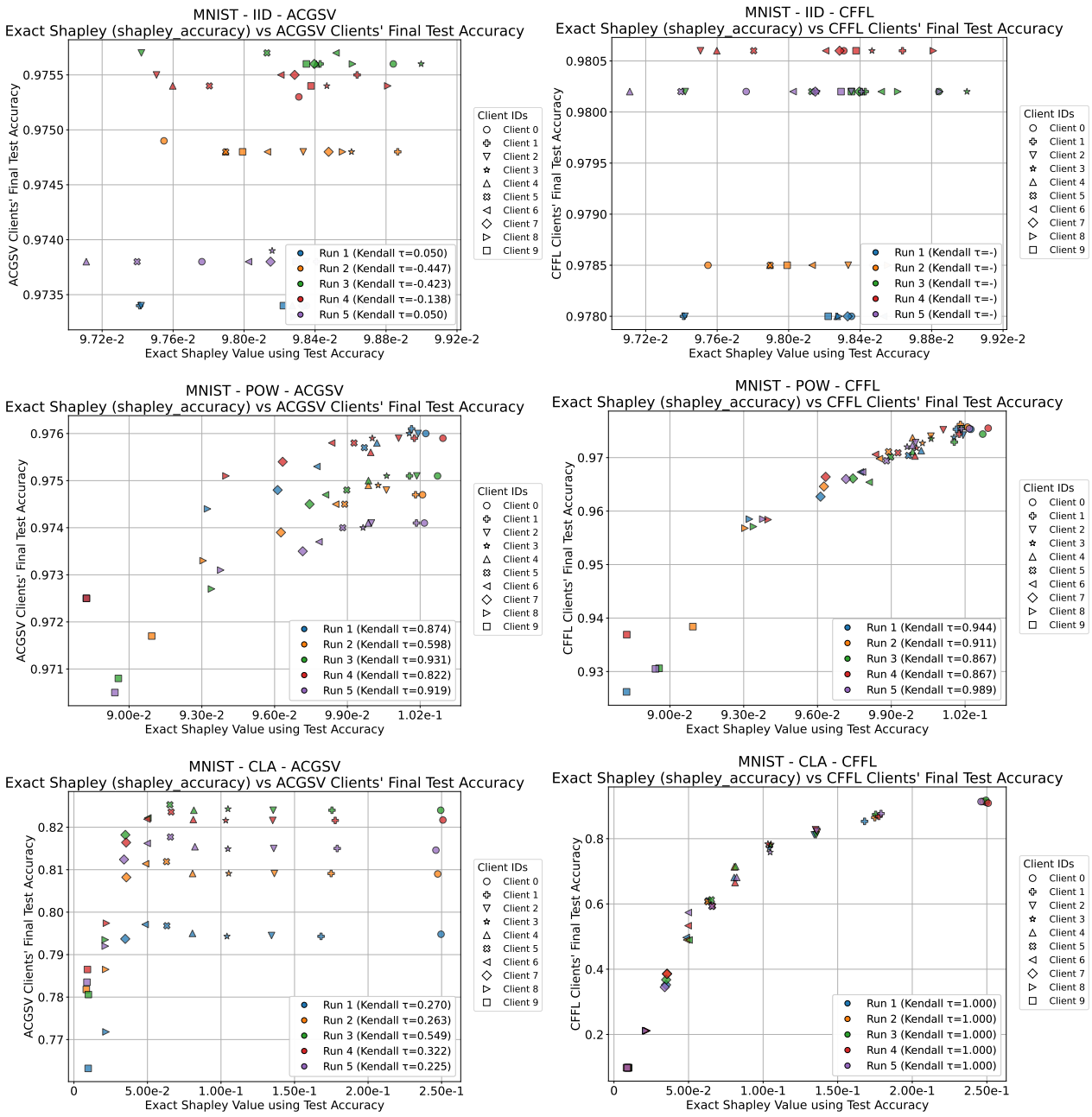


Figure 4.9: Shapley-value-based fairness analysis for ACGSV and CFFL on MNIST under IID, POW, and CLA data partitioning. The left column reports the ACGSV results, while the right column reports the CFFL results. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the clients' Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

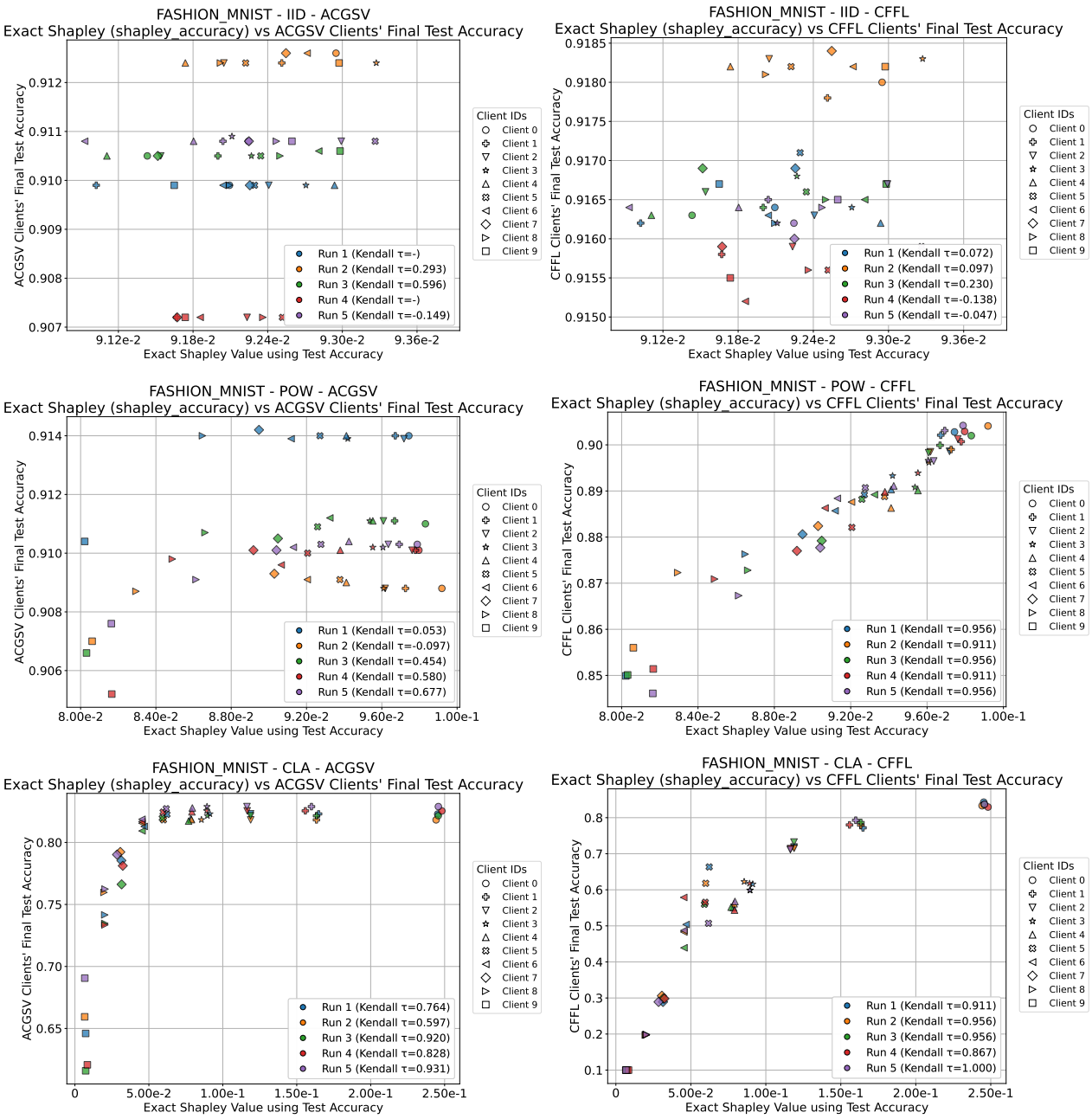


Figure 4.10: Shapley-value-based fairness analysis for ACGSV and CFFL on FA-MNIST under IID, POW, and CLA data partitioning. The left column reports the ACGSV results, while the right column reports the CFFL results. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the clients' Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

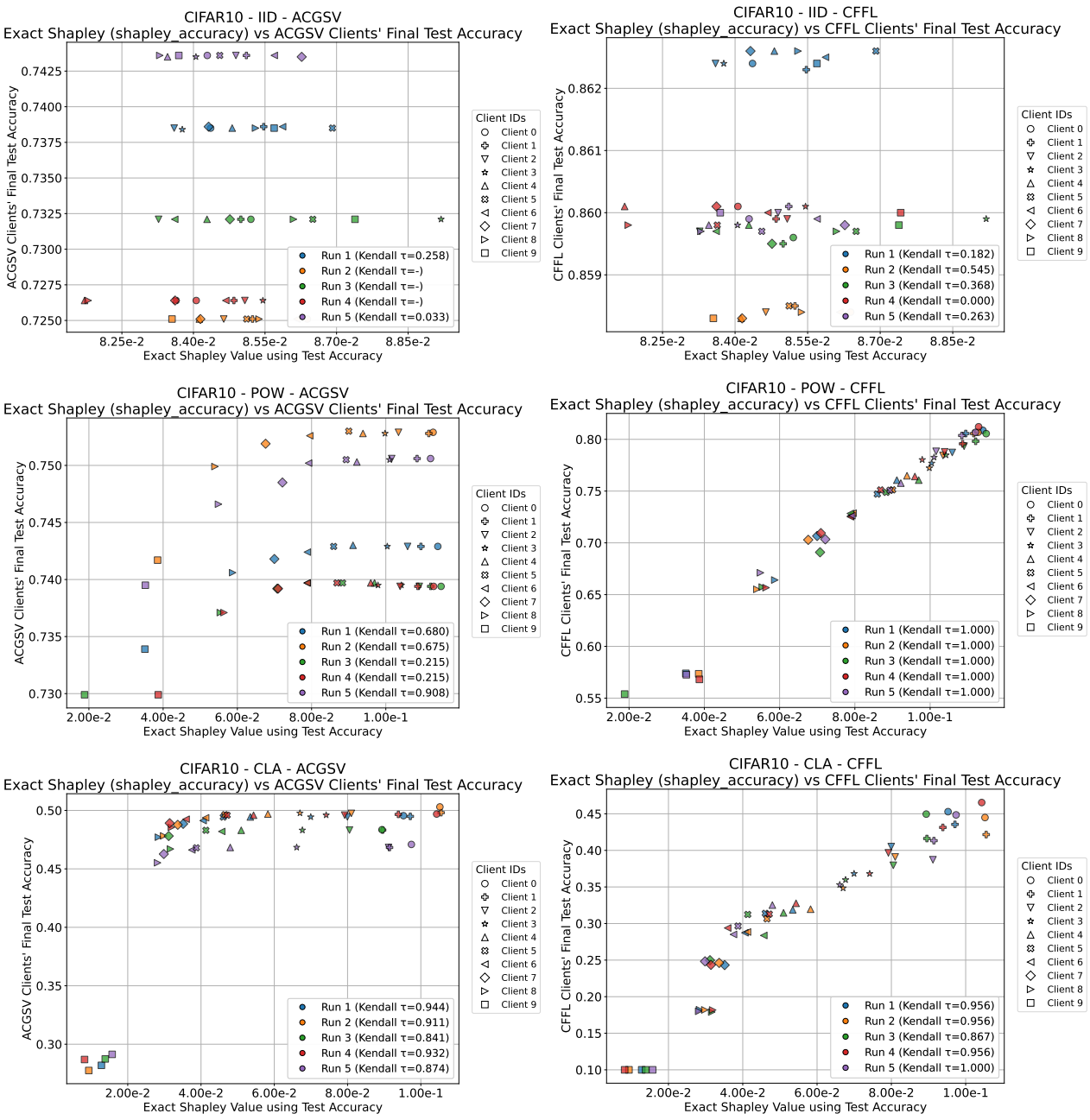


Figure 4.11: Shapley-value-based fairness analysis for ACGSV and CFFL on CIFAR10 under IID, POW, and CLA data partitioning. The left column reports the ACGSV results, while the right column reports the CFFL results. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the clients' Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

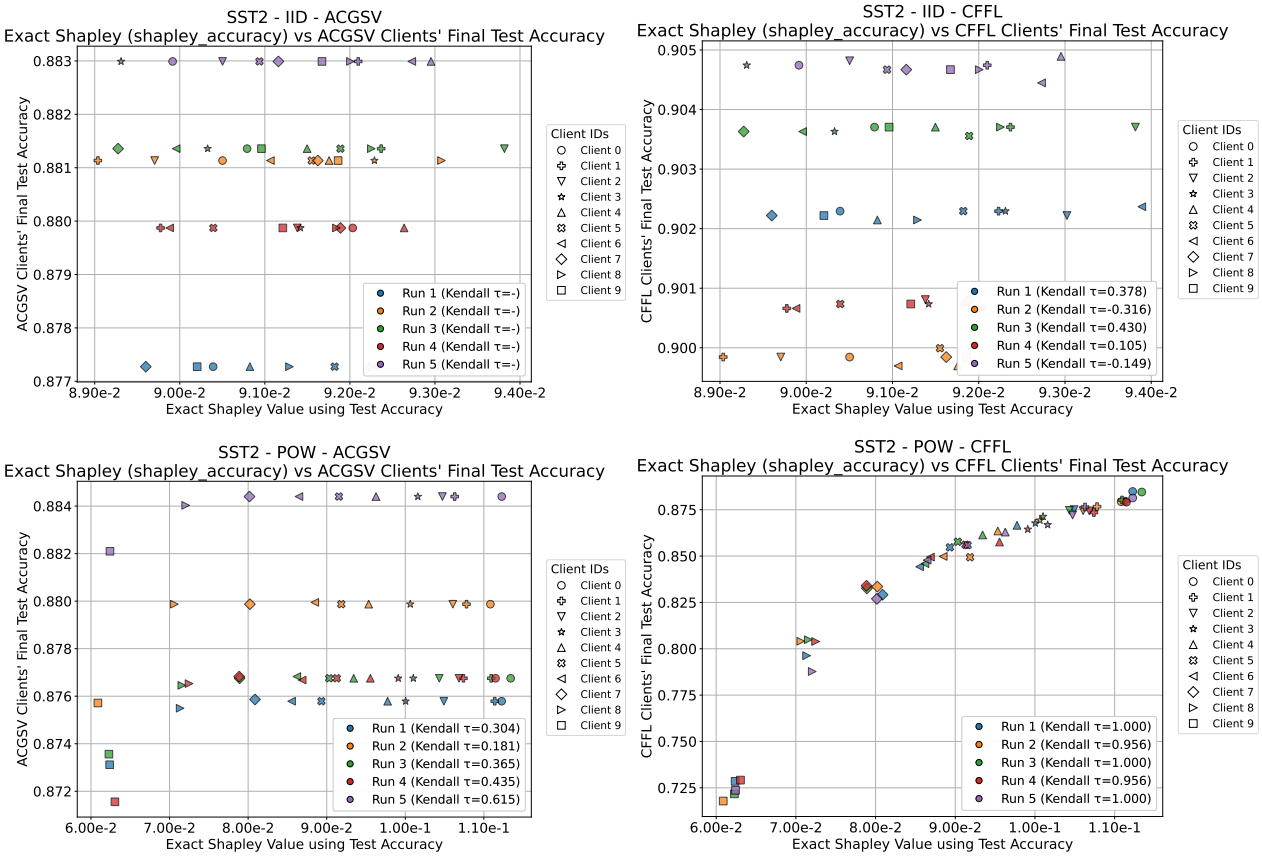


Figure 4.12: Shapley-value-based fairness analysis for ACGSV and CFFL on SST2 under IID, POW, and CLA data partitioning. The left column reports the ACGSV results, while the right column reports the CFFL results. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the clients' Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

4.7 Chapter Conclusion

This chapter showed that Pearson's r, when used as the sole fairness metric, is insufficient to evaluate contribution fairness. A positive Pearson correlation between standalone and federated test accuracy indicates that the two performance profiles vary together, but it does not determine whether the contribution ordering is preserved or whether the final client models are meaningfully differentiated. As a result, frameworks with similar Pearson values can exhibit substantially different fairness outcomes once the convergence behavior and client-level performance distribution are examined.

To address this limitation, this chapter introduced an evaluation protocol based on

Kendall's τ and the GMD, and that can more accurately evaluate the key contribution fairness aspects that Pearson's r fails to. Kendall's τ serves as the primary criterion because it determines whether the contribution ranking is preserved after federated training. Values closer to 1 indicate stronger rank preservation, values closer to 0 indicate weaker rank preservation, and negative values indicate rank reversal. In latter two, the framework should not be interpreted as satisfying contribution fairness, regardless of the amount of model differentiation. Then, the GMD is used to determine whether a preserved ordering is accompanied by meaningful model differentiation. In this thesis, GMD values below 1 percentage point are treated as negligible because the final client models are practically similar in performance.

Using this protocol, the experiments show that CFFL satisfies the two conditions for contribution fairness more consistently than ACGSV in the evaluated heterogeneous settings. CFFL consistently preserves the contribution ranks and produces non-negligible differences among the final client models, whereas ACGSV can obtain apparently favorable Pearson values despite weak differentiation or rank inversions. Therefore, the results show that Pearson's r alone can overstate the fairness achieved by contribution-aware reward mechanisms.

However, the direct comparison between CFFL and ACGSV does not identify which internal design choices produce these differences. The frameworks differ in their FLCE method, the gradient to which Top-K sparsification is applied, the construction of the client-specific $K_{i,t}$ values, and the client model update rule. These differences motivate the ablation studies presented in Chapter 5, where the mechanisms responsible for model differentiation and Contribution Fairness are isolated.

Chapter 5

Ablation Studies

The previous chapter showed that ACGSV and CFFL produce different contribution fairness outcomes, despite sharing the same objective of assigning contribution-aware rewards to clients. It also showed that Pearson's r correlation coefficient is a limited fairness metric because it does not capture contribution rank preservation nor model differentiation effectively, which are fundamental aspects of contribution fairness. This motivated the proposal of a novel fairness evaluation protocol that uses Kendall's τ to measure the contribution rank preservation and the Gini Mean Difference (GMD) to measure the model performance differentiation.

Building on these observations, this chapter presents a set of ablation studies designed to identify which design mechanisms are primarily responsible for the stronger contribution fairness behavior observed in CFFL.

The chapter proceeds in three steps. First, I identify the main design differences between ACGSV and CFFL and explain which mechanisms are expected to strongly influence model differentiation and rank outcomes. Second, I define the ACGSV and CFFL variants used in the ablation study, specifying which mechanisms are removed, added, or modified in each case. Third, I describe the experimental methodology and report the results obtained from the ablation experiments.

The ablation study results show that Top-K sparsification does not map linearly to model convergence and its effectiveness at generating model differentiation and contribution rank preservation heavily depends on the pipeline that maps the estimated contribution to reward sparsity. In the evaluated settings, visible performance degradation occurred mainly when clients consistently received highly sparsified rewards, typically at sparsity

levels of approximately 70% or higher, and it was not the main driver of CFFL high contribution fairness. The results do not advocate against the usage of Top-K sparsified rewards as a method to promote collaborative fairness, but it brings evidence that in order to achieve contribution fairness the reward-generation pipeline needs to map estimated contributions to sufficiently distinct sparsity levels. Otherwise, it either has little effect on model differentiation and rank preservation, or acts primarily as a coarse performance-degradation mechanism rather than a fine-grained method for translating contribution differences into gradient rewards of distinct quality.

A second finding from the ablation studies is that CFFL's local gradient retention is the primary component responsible for model differentiation and preservation of the reference contribution ranking, regardless of whether standalone test accuracy or Shapley Values are used as the contribution reference. In CFFL, the local-gradient retention, is not governed by the FL framework's contribution estimation algorithm and the contribution-score-to-reward mapping pipeline, however, it acts as an under-recognized confounding factor for substantial model differentiation, high Pearson's r and high Kendall's τ values. Despite this finding, this thesis does not advocate the use of local gradient retention as a mechanism of promoting contribution fairness, because the resulting differentiation does not arise from the framework's estimated contribution and ontribution-score-to-reward mapping pipeline, but rather from the continuous client-specific optimization.

5.1 Design Differences between CFFL and ACGSV

Although CFFL and ACGSV share the same goal, the two frameworks differ in several design choices that can affect fairness by influencing model differentiation and rank preservation outcomes. In this subsection, I summarize the main differences between the frameworks without repeating the full mathematical formulation already presented in Sections 3.5 and 3.6.

1. **Gradient aggregation method:** In ACGSV, the server aggregates client gradients using the clients' cumulative contribution estimates, denoted by $r_{i,t}$, as aggregation weights. Therefore, clients with higher estimated contribution have a larger influence on the aggregated gradient, and clients with lower estimated contribution have lower influence on the aggregated gradient. In CFFL, the aggregation rule is determined by the type of data heterogeneity considered by the framework. In data-quantity-skewed settings, the server weights each client update according to the number of local training samples. In class-label-skewed settings, the server weights each update

according to the number of distinct classes present in the client's local dataset[1]. These aggregation rules assume that, clients with more samples and more class diversity in their local datasets have more meaningful data, and should, therefore, have higher weight during the aggregation step on the server.

2. **FLCE algorithm:** ACGSV estimates contribution through gradient-alignment measured as the cosine similarity between each client's gradient and the aggregated gradient, assigning higher contribution to clients whose gradients are more directionally aligned with the aggregated training direction. In contrast, CFFL uses a validation-set-dependent contribution estimator, assigning higher contribution to clients whose reconstructed models achieve higher validation accuracy.

3. **Gradient used for reward generation:** ACGSV applies Top-K sparsification to an aggregated gradient $\vec{u}_{\mathcal{N},t}$ formed from L2-normalized client gradients, while in CFFL, the server applies Top-K sparsification to an aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$ formed from the original accumulated gradients sent by clients, because CFFL does not apply an L2-normalization step before aggregation or reward generation.

4. **Computation of the Top-K sparsification level:** In ACGSV, the server computes $K_{i,t}$ from the client's cumulative estimate contribution $r_{i,t}$ and the maximum value for $r_{i,t}$, making the reward density always defined by the top contributor in the federation. In CFFL, the value of $K_{i,t}$ depends on both the client's cumulative estimated contribution $r_{i,t}$ and the dataset statistic used to characterize the federation. In data-quantity-skewed settings, the server uses the number of local training samples while in class-label-skewed settings, uses the number of distinct local classes. In CFFL, the dataset statistic acts as an additional upper bound on the density of the reward that a client can receive.

5. **Client locally accumulated gradient retention:** In ACGSV, each client model is updated only with the sparsified reward $\vec{\nu}i, t$ received from the server. In CFFL, the clients update their models using the sparsified reward received from the server and part of its own locally accumulated gradient $\Delta\vec{w}i, t$. The gradient retained portion is determined by the client's dataset properties, such as its number of samples or number of local classes, rather than by its estimated contribution.

Based on the design differences listed above, it is not possible to determine by direct inspection whether the stronger fairness and model-differentiation outcomes observed in

[1]As discussed in Subsection 3.6.2, the CFFL class-label-skewed experiments use the sum-normalized class-count weighting instead of the original max-normalized rule as a stabilization step.

CFFL are caused by one specific mechanism or by the interaction among several mechanisms.

5.1.1 Framework Components that Influence Client Model State

Although all the design differences discussed in Section 5.1 have influence on the client model outcomes, these three components have direct influence on the model differentiation and, therefore, FL client ranks: (i) the computation of the Top-K sparsification level, (ii) the gradient used for reward generation, and (iii) the client locally accumulated gradient.

The first direct mechanism is the computation of the Top-K sparsification level. In both frameworks, the server uses the estimated contribution to determine the density of each client's reward vector. Let $q_{i,t} \in [0, 1]$ denote the reward density assigned to client i at round t, where $q_{i,t} = 1$ corresponds to a fully dense reward vector[2] and $q_{i,t} = 0$ corresponds to a fully sparsified reward vector. Equivalently, the reward sparsity is defined as

$$\text{Reward Sparsity}_{i,t} = 1 - q_{i,t}. \tag{55}$$

As discussed in Section 3.5, ACGSV computes the reward density from the client's estimated contribution score $r_{i,t}$, as shown in Equation 56:

$$q_{i,t} \leftarrow \frac{\tanh(\beta \cdot r_{i,t})}{\max_{j \in \mathcal{N}} \tanh(\beta \cdot r_{j,t})}. \tag{56}$$

Under this rule, the client with the largest estimated contribution receives $q_{i,t} = 1$, and therefore receives a fully dense version of the aggregated gradient. The remaining clients receive reward densities proportional to their transformed contribution scores. Consequently, the density of each client's reward depends on how far its estimated contribution is from the top contributor's estimated contribution after the nonlinear transformation $\tanh(\beta \cdot r_{i,t})$.

In contrast, CFFL combines the client's contribution estimate $r_{i,t}$ with a dataset-dependent factor, such as the number of local samples or the number of local classes, as shown in Equations 57 and 58, respectively. This design means that CFFL imposes an additional dataset-dependent upper bound on the reward density that each client can receive. Since Top-K sparsification is used here to control how much aggregated gradient information is returned to each client, this difference would directly affect client convergence and

[2] A fully dense reward vector does not necessarily mean that there are no parameters with value 0 (zero), it means that no gradient sparsification was applied.

model differentiation. Under this interpretation, CFFL can induce stronger differentiation because the reward density is constrained not only by the estimated contribution score, but also by the client's local dataset statistics.

In the data-quantity-skewed setting, CFFL computes

$$q_{i,t} = \frac{r_{i,t}}{\max_{j\in\mathcal{N}} r_{j,t}} \cdot \frac{|\mathcal{D}_i|}{\max_{j\in\mathcal{N}} |\mathcal{D}_j|} \tag{57}$$

In the class-label-skewed setting, CFFL replaces the sample-size factor with a class-diversity factor:

$$q_{i,t} = \frac{r_{i,t}}{\max_{j\in\mathcal{N}} r_{j,t}} \cdot \frac{|\mathcal{C}_i|}{\max_{j\in\mathcal{N}} |\mathcal{C}_j|} \tag{58}$$

Therefore, these reward-density equations, together with the dataset distribution defined in Section 4.1, impose a stratified upper bound on each client's reward density, as summarized in Table 5.1.

Client ID	**Reward Density Upper Bound**	
	POW	**CLA**
Client 0	100.0%	100.0%
Client 1	89.6%	90.0%
Client 2	79.2%	80.0%
Client 3	68.8%	70.0%
Client 4	58.4%	60.0%
Client 5	48.0%	50.0%
Client 6	37.5%	40.0%
Client 7	27.1%	30.0%
Client 8	16.7%	20.0%
Client 9	6.3%	10.0%

Table 5.1: Client-specific reward density upper bounds used by CFFL under POW and CLA partitioning. These values define the maximum fraction of the aggregated gradient that each client can receive before contribution-dependent scaling is applied.

The second direct mechanism is the gradient used for reward generation. In ACGSV, the reward is produced by applying Top-K sparsification to an aggregated gradient $\vec{u}_{\mathcal{N},t}$ formed from L2-normalized client gradients $\vec{u}_{i,t}$. Although the L2-normalization step is

required by the ACGSV contribution-estimation procedure, as discussed in Section 3.5, the framework also uses this aggregated gradient to generate the per-client Top-K sparsified gradient reward.

In CFFL, the server applies Top-K sparsification to the aggregated accumulated gradient $\Delta\vec{w}_{\mathcal{N},t}$, which is formed from the original, unnormalized client aggregated gradients. Therefore, CFFL rewards are generated from a gradient that preserves the original update magnitudes, while ACGSV rewards are generated from a normalized aggregate. This difference may affect the strength of the reward signal delivered to clients. In ACGSV [14], client i updates its model at round t as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \vec{\nu}_{i,t}, \tag{59}$$

where $\vec{\nu}_{i,t}$ denotes the reward vector received by client i from the server. Since all clients start from the same initial server model, i.e., $\vec{w}_{i,0} = \vec{w}_{j,0} = \vec{w}_{0,0}$, for any two distinct clients i and j, the difference between their model parameters at round t is

$$\vec{w}_{i,t} - \vec{w}_{j,t} = \sum_{k=1}^{t} \left(\vec{\nu}_{i,k} - \vec{\nu}_{j,k}\right). \tag{60}$$

Hence, the learning difference between clients i and j is entirely determined by the cumulative differences between their reward vectors across training rounds. In particular, since $\vec{\nu}_{i,t} = \text{mask}(\vec{u}_{\mathcal{N},t}, K_{i,t})$ and $\vec{\nu}_{j,t} = \text{mask}(\vec{u}_{\mathcal{N},t}, K_{j,t})$, the difference $\vec{\nu}_{i,t} - \vec{\nu}_{j,t}$ is nonzero only on the coordinates for which the two masks disagree. Thus, the only additional learning information at round t comes from the magnitude contained in the coordinates selected by one client's mask but excluded by the other. Therefore, one possibility is that the reduced norm of the coordinates that the two masks disagree could not carry enough magnitude to differentiate the models.

The third direct mechanism is whether the clients update their models using their locally accumulated gradients or not. As shown in Equation 59, in the ACGSV framework, the clients only update their models using the Top-K sparsified rewards that they receive from the server. However, CFFL [23], in POW data partitioning scenario, client $i \in \mathcal{N}$ in updates its local model at round t according to the following update rule

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{D}_i}{\max_{j\in\mathcal{N}} \mathcal{D}_j} \cdot \Delta\vec{w}_{i,t} \tag{61}$$

where $\mathcal{D}_i$ represents the number of samples in client's $i \in \mathcal{N}$ local dataset, $\vec{w}_{i,t-1}$ it's the client model at the beginning of round t, $\Delta\vec{w}_{i,t}$ is the client's locally computed gradient,

and $\vec{\nu}_{i,t}$ it the Top-K sparsified reward received by the client. In the scenario where the dataset follows a class-label-skewed scenario, each client updates its local model according to the following update rule

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{\nu}_{i,t} - \frac{\mathcal{C}_i}{\max_{j\in\mathcal{N}} \mathcal{C}_j} \cdot \Delta\vec{w}_{i,t} \tag{62}$$

where $\mathcal{C}_i$ represents the number of distinct classes in client's $i \in \mathcal{N}$ local dataset.

In CFFL, since all clients start from the same initial server model, i.e., $\vec{w}_{i,0} = \vec{w}_{j,0} = \vec{w}_{0,0}$, for any two distinct clients i and j, the difference between their model parameters at round t in the POW data partitioning scenario is

$$\vec{w}_{i,t} - \vec{w}_{j,t} = \sum_{k=1}^{t} \left[\left(1 - \frac{\mathcal{D}_i}{\max_{\ell\in\mathcal{N}} \mathcal{D}_\ell} \right) \Delta\vec{w}_{i,k} - \left(1 - \frac{\mathcal{D}_j}{\max_{\ell\in\mathcal{N}} \mathcal{D}_\ell} \right) \Delta\vec{w}_{j,k} + (\vec{\nu}_{i,k} - \vec{\nu}_{j,k}) \right] \tag{63}$$

and in the CLA data partitioning scenario is

$$\vec{w}_{i,t} - \vec{w}_{j,t} = \sum_{k=1}^{t} \left[\left(1 - \frac{\mathcal{C}_i}{\max_{\ell\in\mathcal{N}} \mathcal{C}_\ell} \right) \Delta\vec{w}_{i,k} - \left(1 - \frac{\mathcal{C}_j}{\max_{\ell\in\mathcal{N}} \mathcal{C}_\ell} \right) \Delta\vec{w}_{j,k} + (\vec{\nu}_{i,k} - \vec{\nu}_{j,k}) \right] \tag{64}$$

Therefore, the expanded pairwise model update equations demonstrate that model differentiation in CFFL is not produced exclusively by the contribution-aware sparsified reward. The pairwise difference between two client models contains two distinct components: the difference between the rewards received by the clients and the difference between their retained locally accumulated gradients.

Although CFFL formulates this operation as a client-side procedure, in which clients willingly retain a prescribed portion of their locally accumulated gradients, this assumption is not practical in a real-world deployment because it requires clients to follow the reward protocol correctly and transparently. However, the same gradient-retention procedure could be implemented directly on the server side, as done in Wu *et al.* [7]. Since the server receives the client updates before constructing the reward, it could inject the locally accumulated gradient component into each client-specific reward vector. In this equivalent formulation, the client would not need to perform the retention operation locally, since the server would construct a reward that already contains both the client-specific reward gradient update and the prescribed local-gradient retention term.

Whether implemented by the client or incorporated directly by the server, the retention of the local-gradient term is independent of the FLCE algorithm and the contribution-score-to-reward mapping, and directly embedded into each client's model update. Table 5.2 shows the per-client values for local gradient retention, which range from 0.0% to 93.7% under POW and from 0.0% to 90.0% under CLA. Thus, CFFL introduces a deterministic and stratified source of model differentiation through local gradient retention. Consequently, part of the observed client-level differentiation could be structurally induced by the retained local updates, rather than by the contribution-aware reward alone.

Client ID	**Locally Accumulated Gradient Retention**	
	POW	**CLA**
Client 0	0.0%	0.0%
Client 1	10.4%	10.0%
Client 2	20.8%	20.0%
Client 3	31.2%	30.0%
Client 4	41.6%	40.0%
Client 5	52.0%	50.0%
Client 6	62.5%	60.0%
Client 7	72.9%	70.0%
Client 8	83.3%	80.0%
Client 9	93.7%	90.0%

Table 5.2: Client-specific local accumulated gradient retention induced by CFFL under POW and CLA partitioning. These values indicate the fraction of the client update that remains locally accumulated.

5.2 Ablation Studies: Methodology

Based on the rationale presented in Subsection 5.1.1, the objective of these ablation studies is to isolate the framework components that directly affect contribution fairness through client-level model differentiation and contribution rank preservation. Therefore, each framework variant is introduced as an analytical tool, not as a proposed improvement over the original frameworks. The purpose of these variants is not to design a framework that performs better than CFFL or ACGSV, but to remove, replace, or preserve specific components in order to identify which design choices are responsible for the observed fair-

ness outcomes. By modifying one component at a time, these ablation studies determine whether performance differences across clients are actually produced by contribution-aware reward mechanisms or by other structural components of the framework.

This distinction is central to the analysis conducted in this chapter. In particular, locally accumulated gradient retention is not treated as a desirable fairness mechanism, nor is this thesis advocating its use as a solution for contribution fairness. Instead, it is treated as a structural factor that must be isolated because it directly affects the final client model state. The ablation studies separate local gradient retention from the contribution-aware reward to determine whether the observed fairness outcomes are caused by the intended reward mechanism or by retained local updates embedded in the client models.

To avoid introducing additional confounding factors, the ablation studies use the same neural architectures, datasets, dataset partitioning strategies, training hyperparameters, framework-specific hyperparameters, and experimental assumptions defined in Section 4.1. The framework variants evaluated in this chapter are introduced in Section 5.4. For each framework variant, I use the same fixed seeds adopted in the previous chapter to control dataset shuffling, client data partitioning, and model weights initialization. Each experimental setting is repeated over five independent runs, increasing the reliability of the reported values and making the comparison between the original frameworks and their ablated variants more consistent.

Since the focus of these ablation studies is to understand the mechanisms that affect model differentiation and contribution rank preservation, I omit the IID partitioning scenarios from the main discussion. Therefore, the main analysis focuses on the POW and CLA partitioning strategies, while the IID results for the framework variants are reported in Appendix D, Section D.3.

5.3 Ablation Studies: Evaluation Metrics

The ablation studies use the evaluation metrics introduced in Subsection 4.1.1 and Section 4.5. Pearson's r is retained for consistency with the contribution fairness metric adopted in related work. Kendall's τ and GMD constitute the novel evaluation protocol proposed in this thesis: Kendall's τ measures contribution-rank preservation, while GMD quantifies the magnitude of client-level model differentiation.

In addition to these fairness-related metrics, I introduce the Mean Paired Accuracy Difference (MPAD) as a complementary measure of predictive performance. Let x_i denote

the standalone test accuracy of client i, and let y_i denote its final federated test accuracy. MPAD is defined as

$$\text{MPAD} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} (y_i - x_i) \,. \tag{65}$$

MPAD is not introduced as a contribution fairness metric or as a criterion for determining which framework is superior. Instead, it provides additional context for the ablation results by showing whether the mechanisms that affect model differentiation and rank preservation are also associated with changes in average client accuracy. The MPAD is expressed in percentage points and measures the average difference between each client's final federated test accuracy and its corresponding standalone test accuracy.

This allows the analysis of potential trade-offs between predictive performance and contribution fairness to be examined without changing the analytical purpose of the framework variants. Values greater than zero indicate an average performance improvement under federated training, while values below zero indicate an average performance deterioration. A value of zero indicates no net average difference between the two training settings, although individual clients may still experience gains or losses that offset one another.

To facilitate comparison across framework variants, I report four aggregate values. The mean Pearson's r is computed using the Fisher's z back-transformation procedure, as described in Section 4.1.1, the mean GMD is computed as the arithmetic average of the GMD values obtained across the five experimental runs, the mean Kendall's τ is computed as the arithmetic average across runs, and the mean MPAD is computed as the arithmetic mean across the runs. The per-run values are described in Appendix D

In the following section, I describe the framework variants evaluated in this ablation study. Each section explains the modification introduced, the motivation behind it, and the specific mechanism it is intended to isolate.

5.4 Ablation Studies: Framework Variants

To validate the intuitions presented in Subsection 5.1.1, I evaluate four framework variants, that were modified to isolate the effects of three components that may strongly influence model differentiation and contribution fairness: the gradient used for reward generation, the use of Top-K sparsification as the reward-generation mechanism, and the retention of the client's locally accumulated gradient during the model update.

5.4.1 UR-ACGSV

The UR-ACGSV variant, with UR standing for "Unnormalized Reward", is designed to evaluate whether the limited model differentiation observed in ACGSV is caused by the type of gradient used for reward generation. In the original ACGSV framework, Top-K sparsification is applied to the aggregated gradient $\vec{u}_{\mathcal{N},t}$, which is obtained from the aggregation of L2-normalized client gradients. Therefore, the reward vectors distributed to the clients are generated from a normalized aggregated update.

For this reason, UR-ACGSV keeps the ACGSV workflow defined in Algorithm 3 unchanged, but replaces the normalized aggregated gradient $\vec{u}_{\mathcal{N},t}$ used for reward generation with the unnormalized aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$.

The intuition behind this variant is that two clients' reward vectors differ only at the coordinates selected by one Top-K mask but not the other. When the unnormalized aggregated gradient is used, the expectation is that the rewards can induce stronger client-level model differentiation and reference contribution ranks preservation. Because the non-overlapping coordinates retain their original magnitudes, therefore, they carry more magnitude information which may be sufficient to push clients' models into difference optimization directions.

In the original ACGSV framework, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\vec{u}_{\mathcal{N},t}, K_{i,t}) \tag{66}$$

In the UR-ACGSV framework variant, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_{i,t}) \tag{67}$$

This variant isolates whether using the raw aggregated gradient for reward generation is sufficient to increase model differentiation in ACGSV. However, in both ACGSV and UR-ACGSV, the reward density assigned to client i is determined only by its estimated contribution relative to the highest estimated contribution in the federation, as shown in Equation 68:

$$q_{i,t} = \frac{\tanh(\beta \cdot r_{i,t})}{\max_{j\in\mathcal{N}} \tanh(\beta \cdot r_{j,t})} \tag{68}$$

Therefore, when a client i's estimated contribution is numerically close to the top contributor's estimated contribution, its reward density also becomes close to the top contributor's reward density, and if the reward densities are close, the resulting sparsified rewards may remain too similar to induce strong model differentiation.

5.4.2 OTOPK-CFFL

Compared to ACGSV, and UR-ACGSV, CFFL provides a different setting for evaluating the effect of using Top-K sparsification to generate rewards because it imposes an explicitly stratified reward-density structure bounded by dataset-dependent statistics, as previously discussed in 5.1.1.

In the original CFFL framework, each client updates its model using both the Top-K sparsified reward received from the server and a portion of its own locally accumulated gradient. As a result, the client model state is affected by two components: the server-generated sparsified reward and the client's own local update.

The OTOPK-CFFL framework variant, with OTOPK standing for "Only Top-K Sparsification", isolates the effect of the Top-K sparsified reward on model differentiation and contribution rank preservation. In this variant, the locally accumulated gradient term is fully removed from the client model update, thus each client updates its model exclusively with the Top-K sparsified reward vector received from the server. The reward density remains determined by the client contribution estimate and by the dataset-dependent statistic used in CFFL

In the original CFFL framework, for the POW data partitioning scenario the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_{i,t}) + \left(1 - \frac{\mathcal{D}_i}{\max_{j\in\mathcal{N}} \mathcal{D}_j}\right) \cdot \Delta\vec{w}_{i,t} \tag{68}$$

and for the CLA data partitioning scenario, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_{i,t}) + \left(1 - \frac{\mathcal{C}_i}{\max_{j\in\mathcal{N}} \mathcal{C}_j}\right) \cdot \Delta\vec{w}_{i,t} \tag{69}$$

In the OTOPK-CFFL framework variant, for both data partitioning scenarios, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\Delta\vec{w}_{\mathcal{N},t}, K_{i,t}) \tag{70}$$

This framework variant is of particular interest because each client's reward density is bound to a known level (Table 5.1), which provides a controlled sweep of density levels, allowing the relationship between reward density and converged federated accuracy to be examined directly, and in particular whether it is linear.

5.4.3 DPR-CFFL

The DPR-CFFL framework variant stands for "Dense Gradient with Partial Retention". This naming came from its design, because it removes the Top-K sparsification step and assigns the same dense aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$ to every client, independently of its estimated contribution, but it preserves CFFL's local gradient retention mechanism: each client retains a portion of its locally accumulated gradient according to its dataset statistics, such as the number of local samples or local classes.

In the original CFFL framework, for the POW data partitioning scenario the client update rule is defined in Equation 68 and for the CLA data partitioning scenario, the client update rule is defined in Equation 69, which is omitted here to prevent repetition.

In the DPR-CFFL framework variant, for the POW data partitioning scenario the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{\mathcal{N},t} + \left(1 - \frac{\mathcal{D}_i}{\max_{j\in\mathcal{N}} \mathcal{D}_j}\right) \cdot \Delta\vec{w}_{i,t} \tag{71}$$

and for the CLA data partitioning scenario, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{\mathcal{N},t} + \left(1 - \frac{\mathcal{C}_i}{\max_{j\in\mathcal{N}} \mathcal{C}_j}\right) \cdot \Delta\vec{w}_{i,t} \tag{72}$$

This DPR-CFFL variant isolates locally accumulated gradient retention to determine whether the contribution fairness outcome obtained by CFFL are driven by a contribution-independent mechanism. Since all clients receive the same dense server update, any remaining model differentiation cannot be attributed to contribution-dependent Top-K reward allocation. Instead, it must be produced by the retained local gradient component embedded in the client model update rule.

5.4.4 DFR-ACGSV

The DFR-ACGSV, with DFR standing for "Dense Gradient with Full Retention", follows a similar design to DPR-CFFL in Section 5.4.3, however, instead of applying a dataset-statistics dependent local gradient retention, DFR-ACGSV allows full local gradient retention. Therefore, each client retains its full locally accumulated gradient, while also applying the dense aggregated gradient generated by the server to update its local model.

Despite retaining the full locally accumulated gradient, denoted as $\Delta\vec{w}_{i,t}$, DFR-ACGSV is not equivalent to standalone training because every client also incorporates a shared server update. At each round, the server constructs the dense normalized aggregated gradient $\vec{u}_{\mathcal{N},t}$ from the L_2-normalized client gradients $\vec{u}_{i,t}$ and distributes the same vector to all clients without applying Top-K sparsification.

Each client then updates its model using both this shared aggregated gradient and its own locally accumulated gradient $\Delta\vec{w}_{i,t}$. Thus, the framework preserves client-specific local information while maintaining collaborative updates from the federation. This local-gradient retention rule follows a design similar to those adopted in related work [22, 7].

In the original ACGSV framework, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \text{Top-K}(\vec{u}_{\mathcal{N},t}), \tag{75}$$

In the DFR-ACGSV framework variant, the client update rule is defined as

$$\vec{w}_{i,t} = \vec{w}_{i,t-1} + \Delta\vec{w}_{i,t} + \vec{u}_{\mathcal{N},t}. \tag{76}$$

5.5 Ablation study results: UR-ACGSV Variant

Before interpreting the effect of the ablation, I first describe how the experimental results are presented and organized. Explaining their organization beforehand allows the analysis to refer directly to the observed convergence patterns, client-level performance relationships, and reward-sparsity trajectories.

Figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, and 5.7 compare the results obtained with the original ACGSV framework and the UR-ACGSV variant across the evaluated datasets under POW and CLA partitioning. Each figure is arranged as a comparison grid: the left column presents the original ACGSV results, while the right column presents the corresponding UR-ACGSV results. Although the ACGSV results were previously reported in Section 4.3, they are repeated here to support a direct comparison with the tested framework variant. The first row presents the client convergence trajectories, the second row presents the relationship between standalone and federated test accuracy, and the final row presents the evolution of the per-client reward sparsity.

The convergence line plots show each client's mean FL test accuracy over the five runs, where the x-axis represents the FL training round and the y-axis represents the test

accuracy. In the reward sparsity line plot, the x-axis represents the FL training round, the y-axis represents the reward sparsity, and each line corresponds to one client's reward sparsity. In the scatter plots, the x-axis represents the final test accuracy in standalone mode, while the y-axis represents the final test accuracy after federated training with the respective FL framework, each marker represents one client in one experimental run, and the marker shape identifies the client ID. The legend uses colored circles to distinguish the experimental runs. These symbols are not the same as the circular marker used to identify a specific client. The scatter plots show that most clients converge to very similar federated test accuracies, forming a dense cluster of points, while only a small subset of lower-performing clients remains separated from this cluster.

As in Chapter 4, the identity line was included in the scatter plots as a visual reference to indicate whether each client obtained an accuracy gain or loss after federated training relative to its standalone baseline. Points above the line correspond to clients that obtained test accuracy gain, while points below the line correspond to clients that had test accuracy loss after federated training. This identity line might not always be entirely visible, such as in Figure 5.3 which represents the FA-MNIST POW scenario.

Tables 5.3 and 5.4 report the mean Pearson's r, Kendall's τ, GMD, and MPAD values obtained for the frameworks being compared in this study in the POW and CLA settings, respectively.[3]

The first observation from this ablation study is based on the Kendall's τ values reported in Tables 5.3 and 5.4. Across all datasets except CIFAR10 under the CLA distribution, UR-ACGSV produced substantially lower Kendall's τ values than ACGSV, with most values lying closer to zero and therefore indicating a weaker association between the reference contribution ranking and the final client performance ranking. This weaker rank association, although present in most scenarios, is particularly evident in the MNIST CLA and CIFAR10 POW scenarios shown in Figures 5.2 and 5.3, respectively. In MNIST CLA, some clients with higher standalone test accuracy obtained lower federated test accuracy than clients with lower standalone test accuracy, resulting in multiple rank inversions. In the CIFAR10 scenario, the weak rank association was instead primarily caused by the convergence of almost all client models to tied federated test accuracy values, despite their distinct standalone test accuracy values.

[3]The Kendall's τ, GMD and MPAD values computed with standalone test accuracy for each run are with the UR-ACGSV reported in Tables 5.5, D.9, D.10, and D.11, respectively.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (POW)	ACGSV	**0.959**	**0.828**	**0.11**	4.02
	UR-ACGSV	0.750	0.038	0.05	**4.53**
FA-MNIST (POW)	ACGSV	**0.884**	**0.275**	**0.09**	**4.76**
	UR-ACGSV	0.877	0.221	**0.09**	3.99
CIFAR10 (POW)	ACGSV	**0.708**	**0.526**	0.26	41.14
	UR-ACGSV	0.586	-0.020	**3.28**	**51.46**
SST2 (POW)	ACGSV	**0.737**	**0.380**	**0.08**	15.52
	UR-ACGSV	0.647	0.204	0.05	**18.09**

Table 5.3: Comparison between ACGSV and UR-ACGSV under POW data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric; tied values are both highlighted.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (CLA)	ACGSV	**0.747**	**0.326**	**1.16**	**29.52**
	UR-ACGSV	0.037	0.110	0.26	22.37
FA-MNIST (CLA)	ACGSV	**0.795**	**0.808**	**5.24**	**33.65**
	UR-ACGSV	0.717	0.556	1.32	18.70
CIFAR10 (CLA)	ACGSV	0.834	0.719	4.44	20.07
	UR-ACGSV	**0.838**	**0.733**	**5.19**	**21.08**

Table 5.4: Comparison between ACGSV and UR-ACGSV under CLA data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

The second observation from this ablation study is based on the GMD metrics also reported in Tables 5.3 and 5.4. The GMD values produced by the UR-ACGSV framework variant were only larger than the GMD values produced by the ACGSV under CIFAR10 POW and CIFAR10 CLA, while for the other scenarios the UR-ACGSV in fact produced

an inferior model differentiation. These results suggest that the Top-K sparsified rewards generated from the unnormalized aggregated gradient in UR-ACGSV preserve the reference contribution ranking less effectively than those generated from the normalized aggregated gradient in ACGSV, which contradict the intuition presented in Section 5.4.1.

The substantially lower Kendall's τ values and, in most scenarios, lower GMD values can be understood by examining the convergence curves in Figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, and 5.7. This behavior is particularly evident in the CIFAR10 POW and SST2 POW scenarios shown in Figures 5.5 and 5.7, respectively. Under UR-ACGSV, the client models reached their performance plateaus considerably earlier than those trained with ACGSV because the rewards generated from the unnormalized aggregated gradient retain a larger norm and, consequently, they induce larger effective parameter updates and accelerate optimization along the dominant directions of the aggregated gradient.

Although the rewards are being sparsified based on the estimated contribution, the Top-K sparsification procedure in Algorithm 2 retains the coordinates with the largest absolute magnitudes while removing those with smaller absolute magnitudes. Consequently, two clients receiving rewards with different densities still share many of the same dominant coordinates, and their updates differ primarily in lower-magnitude coordinates that exert a smaller influence on the optimization trajectory. Since all rewards are derived from the same aggregated gradient, this overlap produces highly aligned client update directions. The client models therefore approach similar solutions before the differences introduced by their reward sparsity levels can accumulate sufficiently to produce persistent differences in predictive performance.

Once the client models have converged and their generalization performance has stabilized, subsequent client-specific rewards have limited capacity to establish contribution-aligned model differentiation. Producing further separation would require the lower-magnitude differences between rewards to accumulate over additional rounds, potentially extending training beyond the point of best generalization and increasing the risk of overfitting. Consequently, the final client accuracies become concentrated within a narrow range or, in some cases, converge to tied values, thereby reducing both the GMD and the preservation of the reference contribution ranking.

A third observation concerns the MPAD values reported in Tables 5.3 and 5.4. As discussed in Section 5.3, MPAD is not a contribution-fairness metric; rather, it provides complementary information about the average predictive-performance gain obtained by the client models relative to their standalone baselines. Under POW partitioning, UR-ACGSV increased MPAD in three of the four datasets: from 4.02 to 4.53 percentage points for MNIST, from 41.14 to 51.46 percentage points for CIFAR10, and from 15.52 to

18.09 percentage points for SST2. The only exception was FA-MNIST, for which MPAD decreased from 4.76 to 3.99 percentage points. Therefore, under POW partitioning, the use of the unnormalized aggregated gradient generally increased the average client accuracy gain.

The opposite tendency was observed under CLA partitioning. For MNIST, MPAD decreased from 29.52 to 22.37 percentage points, while for FA-MNIST it decreased more substantially, from 33.65 to 18.70 percentage points, and only CIFAR10 exhibited a small increase, from 20.07 to 21.08 percentage points. The convergence curves help explain these reductions. In both MNIST-CLA and FA-MNIST-CLA, UR-ACGSV initially produced a faster increase in client test accuracy, but the client models reached a performance plateau earlier and exhibited a slight decline during the later rounds. By contrast, the client models trained with ACGSV continued to improve for a longer period and reached higher final test accuracies.

A fourth observation that can be drawn from the scatter plots and reward-sparsity charts in Figures 5.3, 5.4, 5.5, 5.6, and 5.7 is that the relationship between reward sparsity and client performance degradation is nonlinear. Across several scenarios in both the ACGSV and UR-ACGSV, clients receiving rewards with different low-to-moderate sparsity levels converged to nearly identical test accuracies, indicating that the removal of additional low-magnitude coordinates had only a limited effect on their final performance. When performance degradation occurred, it was generally concentrated among the clients receiving the most aggressively sparsified rewards within a given scenario, which revolved around the value of 70% sparsity.

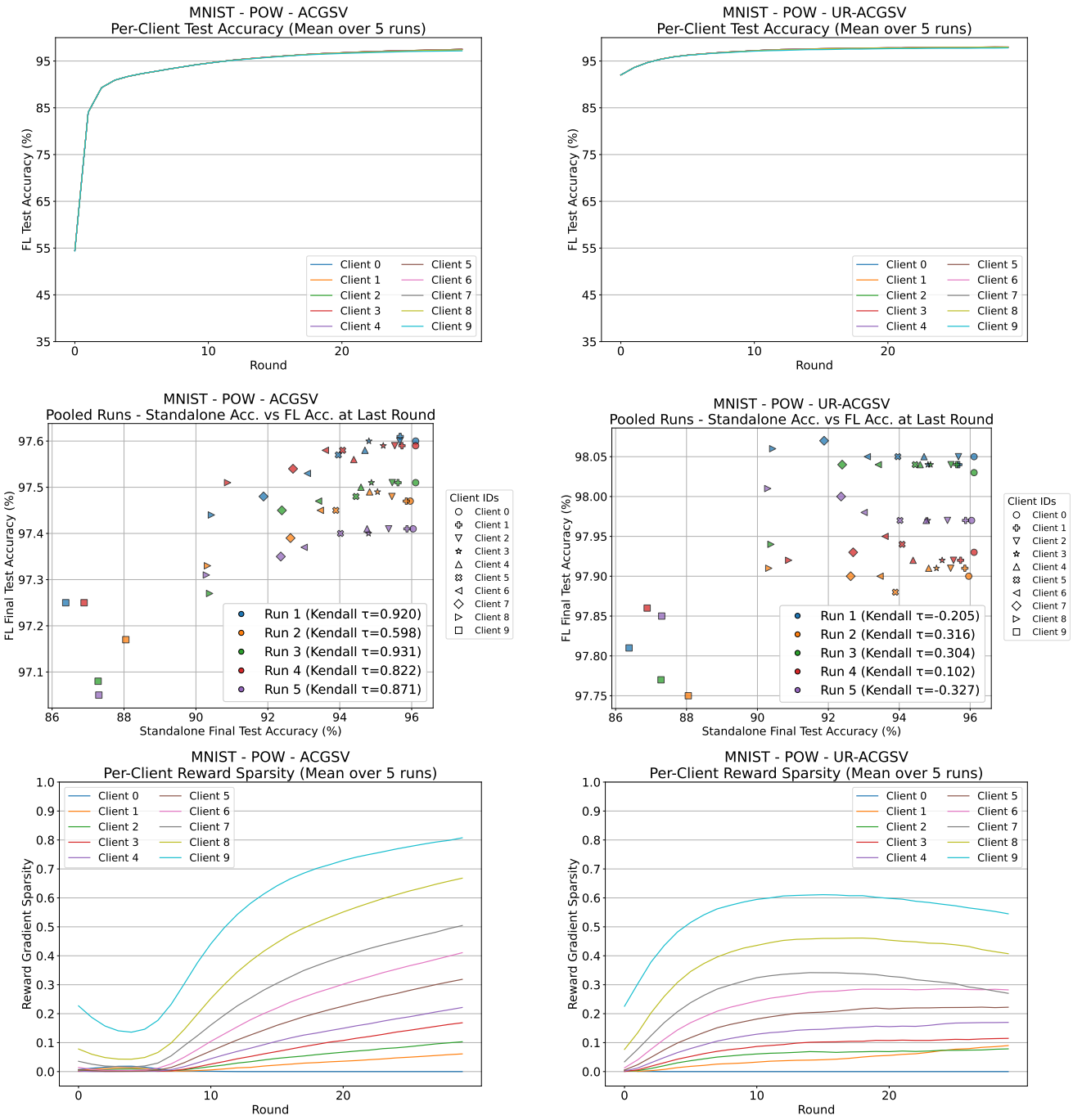


Figure 5.1: Per-client results for the ACGSV framework and the UR-ACGSV variant on MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

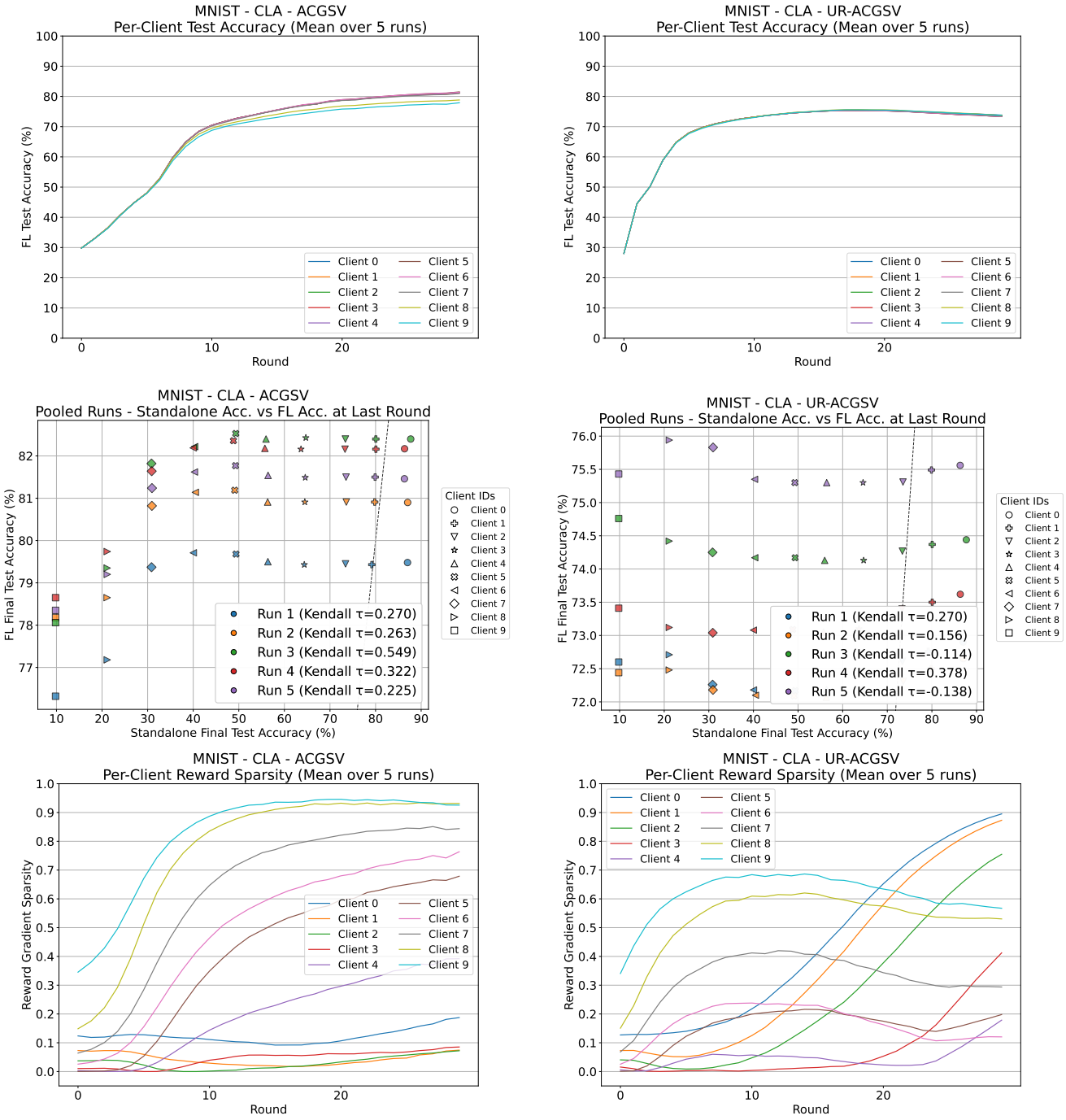


Figure 5.2: Per-client results for the ACGSV framework and the UR-ACGSV variant on MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

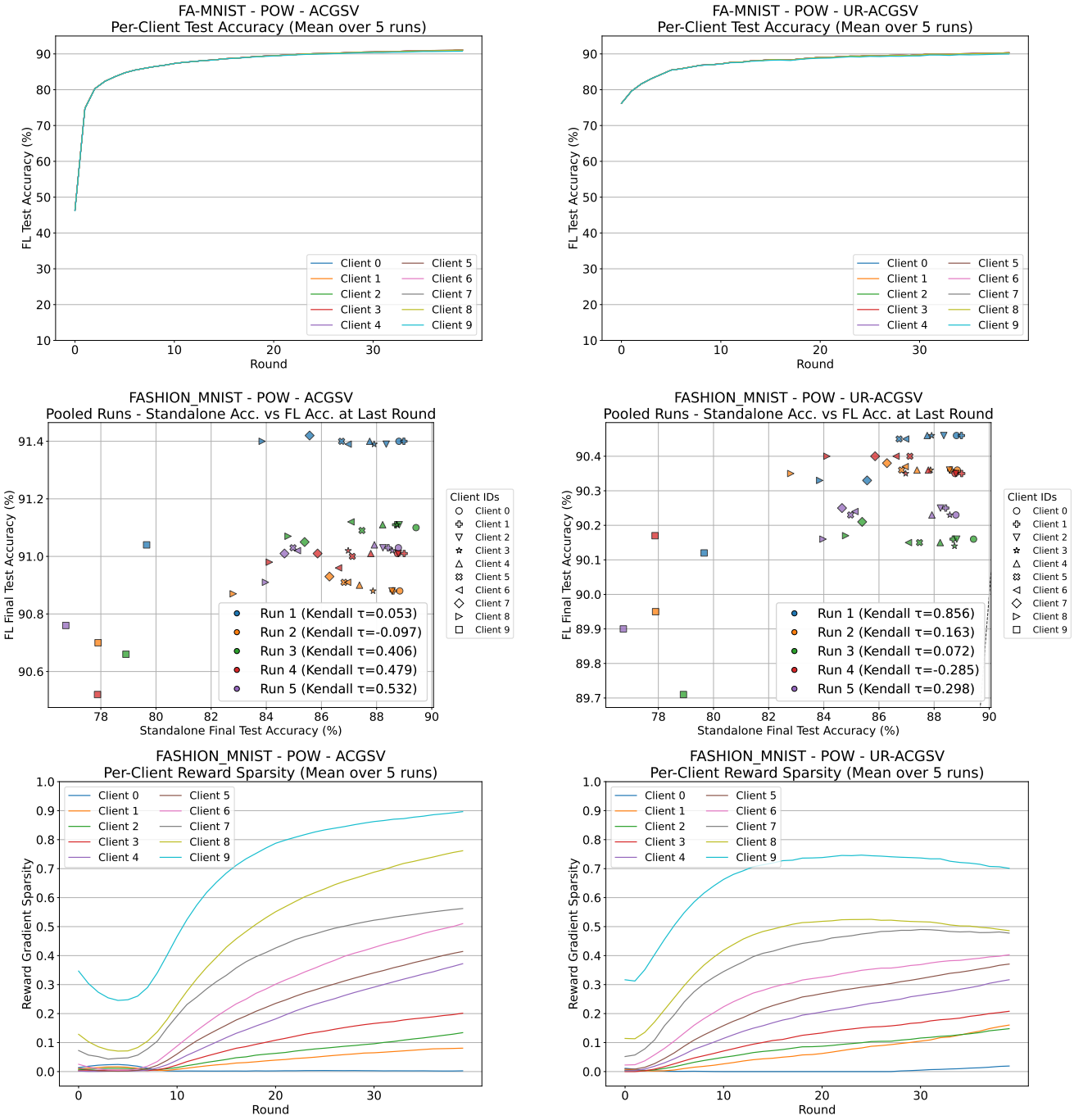


Figure 5.3: Per-client results for the ACGSV framework and the UR-ACGSV variant on FA-MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

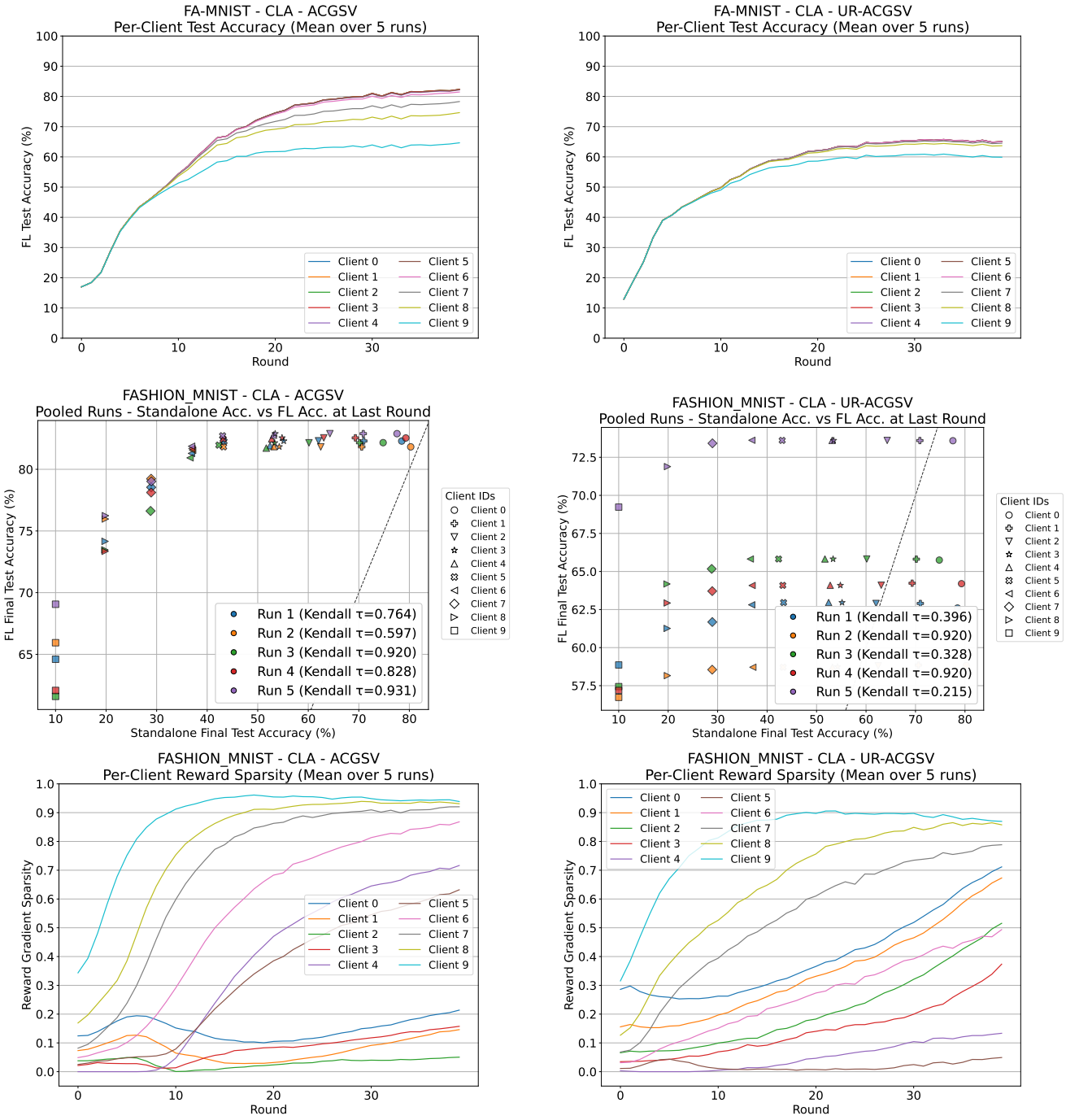


Figure 5.4: Per-client results for the ACGSV framework and the UR-ACGSV variant on FA-MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

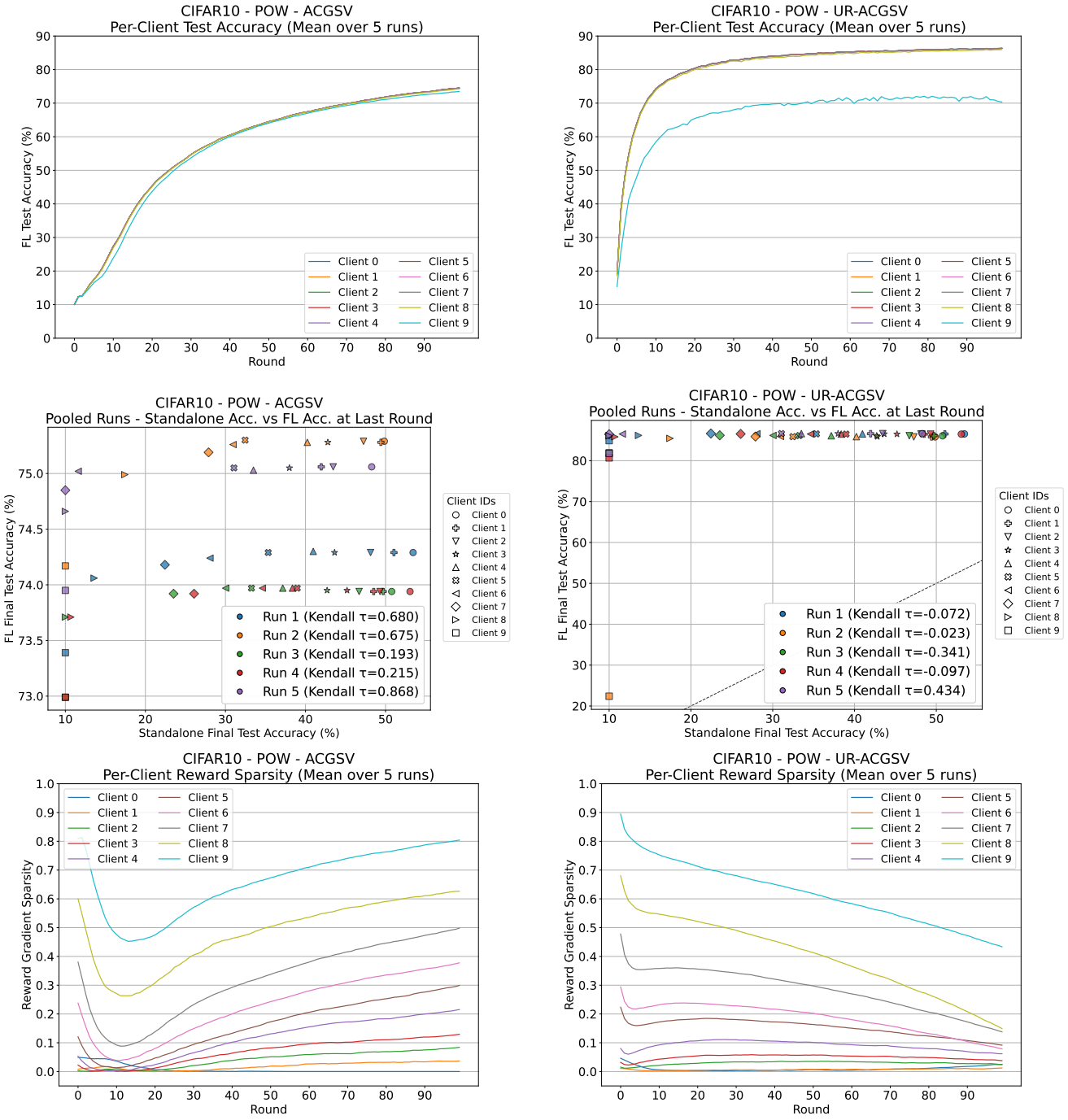


Figure 5.5: Per-client results for the ACGSV framework and the UR-ACGSV variant on CIFAR10 under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

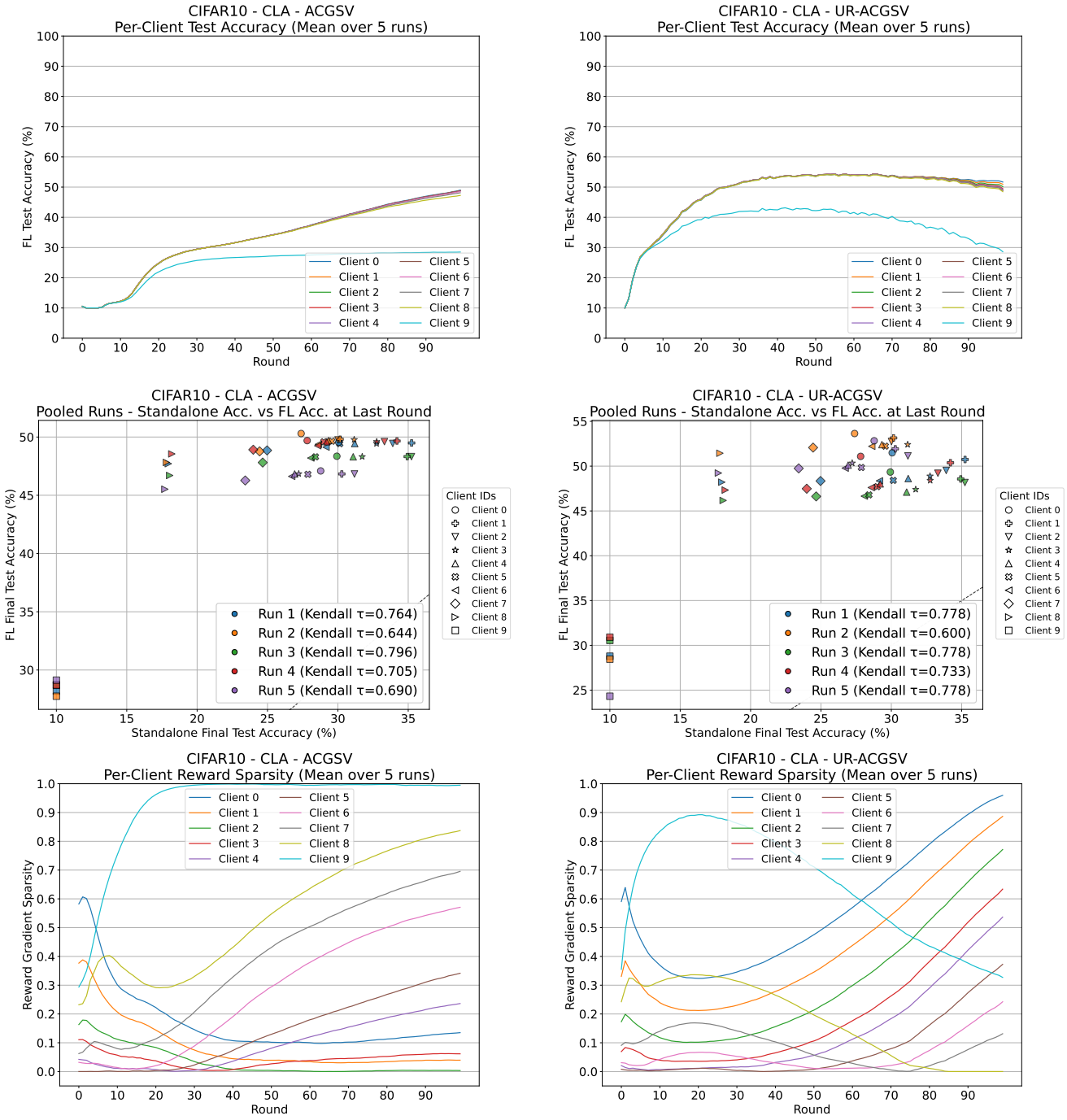


Figure 5.6: Per-client results for the ACGSV framework and the UR-ACGSV variant on CIFAR10 under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

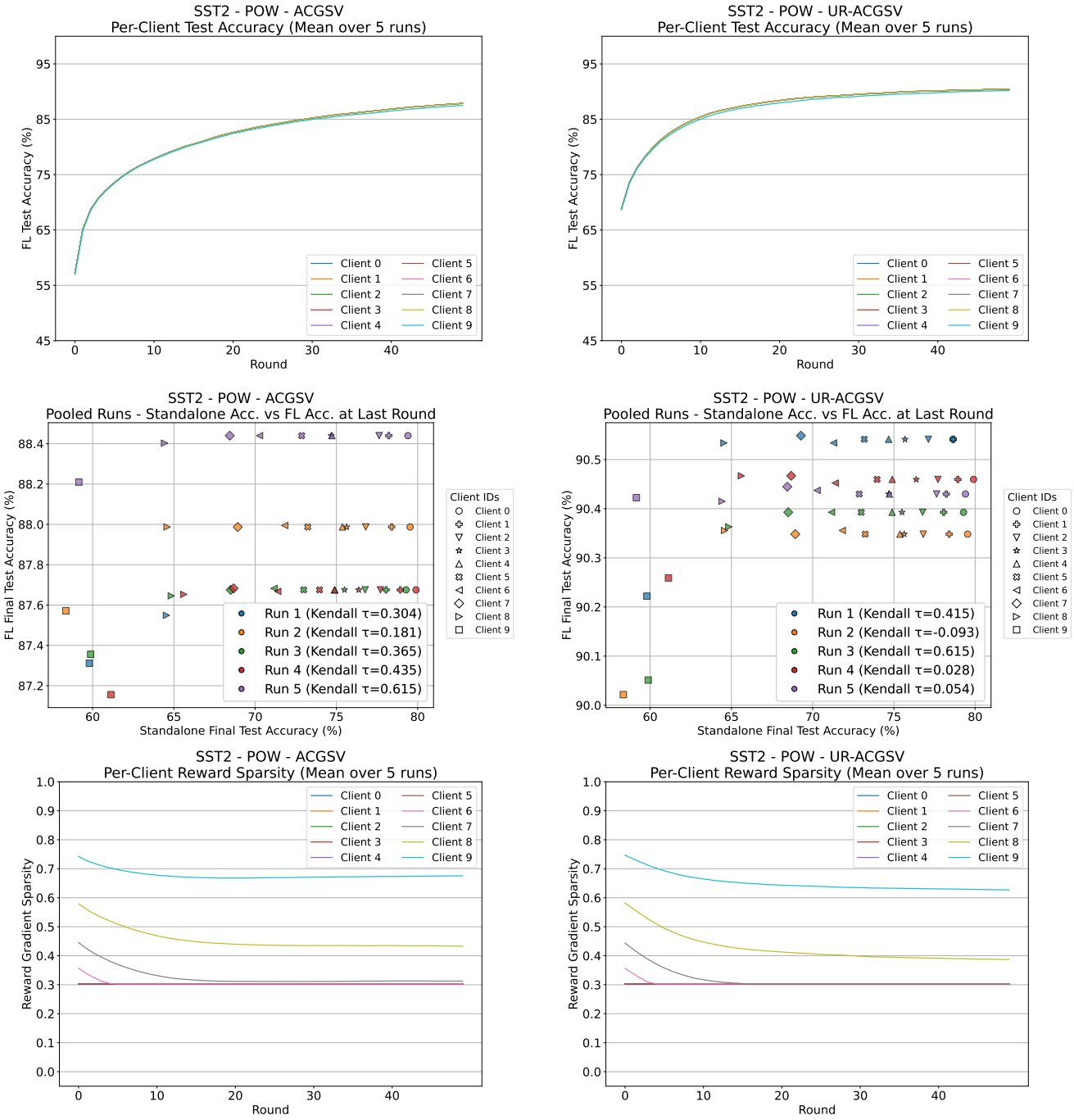


Figure 5.7: Per-client results for the ACGSV framework and the UR-ACGSV variant on SST2 under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

Although not the initial intention of the ablation study with UR-ACGSV, the results obtained with this framework variant emphasize the limitations of using Pearson's r as the only metric for evaluating contribution fairness. Table 5.5 reports the per-run Pearson's r values obtained with UR-ACGSV. In scenarios such as MNIST-POW, FA-MNIST-POW, FA-MNIST-CLA, CIFAR10-CLA, and SST2-POW, Pearson's r remains high, with $r \geq 0.7$. Therefore, under a Pearson-only evaluation, these values could suggest that UR-ACGSV achieves strong contribution fairness, despite the analysis and results shown earlier in this section indicated the opposite.

Runs	MNIST			FA-MNIST			CIFAR10			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	0.061	0.747	0.257	-	0.965	0.761	0.065	0.632	0.821	-	0.714
r_2	0.131	0.744	0.183	0.286	0.863	0.747	0.389	0.642	0.841	-	0.722
r_3	-	0.897	-0.365	0.158	0.828	0.679	0.053	0.524	0.837	-	0.739
r_4	-	0.713	0.491	0.520	0.711	0.671	-0.633	0.645	0.830	-	0.648
r_5	-0.525	0.530	-0.394	0.156	0.890	0.719	-0.178	0.466	0.856	-	0.311

Table 5.5: Per-run Pearson's r correlation coefficients obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Consistent with the analysis conducted in Chapter 4, I also evaluate UR-ACGSV using the client Shapley Values as the contribution reference. Tables 5.6 and 5.7 report the mean Pearson's r, Kendall's τ, GMD, and MPAD values[4] obtained under this Shapley-value-based analysis. The GMD values are the same as those reported in Tables 5.3 and 5.4, because GMD and MPAD are computed only over the final federated test accuracies and does not depend on the contribution reference used for the correlation analysis.

Overall, recomputing the correlation metrics with Shapley Values and evaluating the results does not change the interpretation of the UR-ACGSV results. Although the Pearson's r and Kendall's τ values differ from those reported in Tables 5.3 and 5.4, this is driven by the change in contribution reference from standalone test accuracy to coalition-based marginal utility. However, the same framework-level pattern remains. UR-ACGSV does

[4]The Pearson's r and Kendall's τ correlation coefficient values computed with shapley value for each run are reported in Tables D.12 and D.13.

not consistently preserve the contribution-dependent ordering across datasets and it does not produce the level of model differentiation observed with CFFL.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (POW)	ACGSV	**0.949**	**0.829**	**0.11**	4.02
	UR-ACGSV	0.725	0.049	0.05	**4.53**
FA-MNIST (POW)	ACGSV	**0.777**	**0.333**	**0.09**	**4.76**
	UR-ACGSV	0.757	0.218	**0.09**	3.99
CIFAR10 (POW)	ACGSV	**0.834**	**0.539**	0.26	41.14
	UR-ACGSV	0.735	-0.013	**3.28**	**51.46**
SST2 (POW)	ACGSV	**0.657**	**0.380**	**0.08**	15.52
	UR-ACGSV	0.563	0.204	0.05	**18.09**

Table 5.6: Comparison between ACGSV and UR-ACGSV under POW data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, GMD, and MPAD across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric; tied values are both highlighted.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (CLA)	ACGSV	**0.537**	**0.326**	**1.16**	**29.52**
	UR-ACGSV	0.333	0.110	0.26	22.37
FA-MNIST (CLA)	ACGSV	**0.570**	**0.808**	**5.24**	**33.65**
	UR-ACGSV	0.496	0.556	1.32	18.70
CIFAR10 (CLA)	ACGSV	0.565	0.900	4.44	20.07
	UR-ACGSV	**0.641**	**0.947**	**5.19**	**21.08**

Table 5.7: Comparison between ACGSV and UR-ACGSV under CLA data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, GMD, and MPAD across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

Therefore, the Shapley-value-based analysis corroborates the standalone-accuracy-based conclusion. Replacing the L2-normalized aggregated gradient $\vec{u}_{\mathcal{N},t}$ with the unnormalized aggregated gradient $\Delta\vec{w}_{\mathcal{N},t}$ in the Top-K reward does not, by itself, provide a reliable mechanism for model differentiation and contribution rank preservation in FL. The limitation of UR-ACGSV is therefore not an artifact of using standalone test accuracy as the contribution proxy; it persists when client contribution is measured through Shapley Values.

Figures 5.8 and 5.9 present the scatter plots used in the Shapley-value-based fairness analysis of UR-ACGSV. In each plot, the x-axis represents the exact client Shapley Value computed using test accuracy as the coalition utility, while the y-axis represents the final federated test accuracy achieved by the same client. These figures visually confirm the results reported in Tables 5.6 and 5.7, showing that UR-ACGSV does not consistently preserve a clear contribution-dependent ordering when contribution is measured through exact Shapley Values.

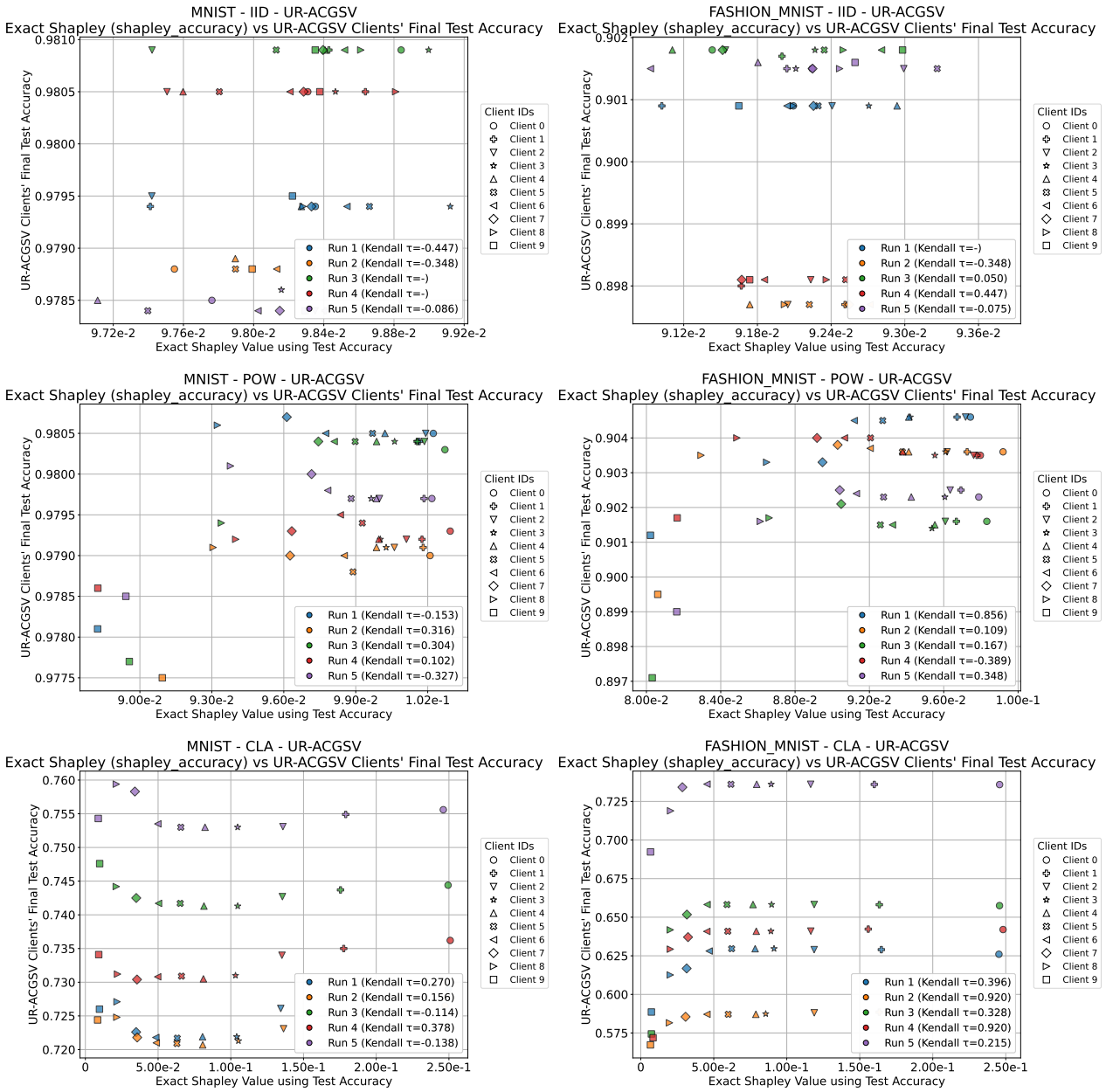


Figure 5.8: Shapley-value-based fairness analysis for UR-ACGSV on MNIST and FA-MNIST under IID, POW, and CLA data partitioning. The left column reports the UR-ACGSV results on the MNIST dataset, while the right column reports the results on the FA-MNIST dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

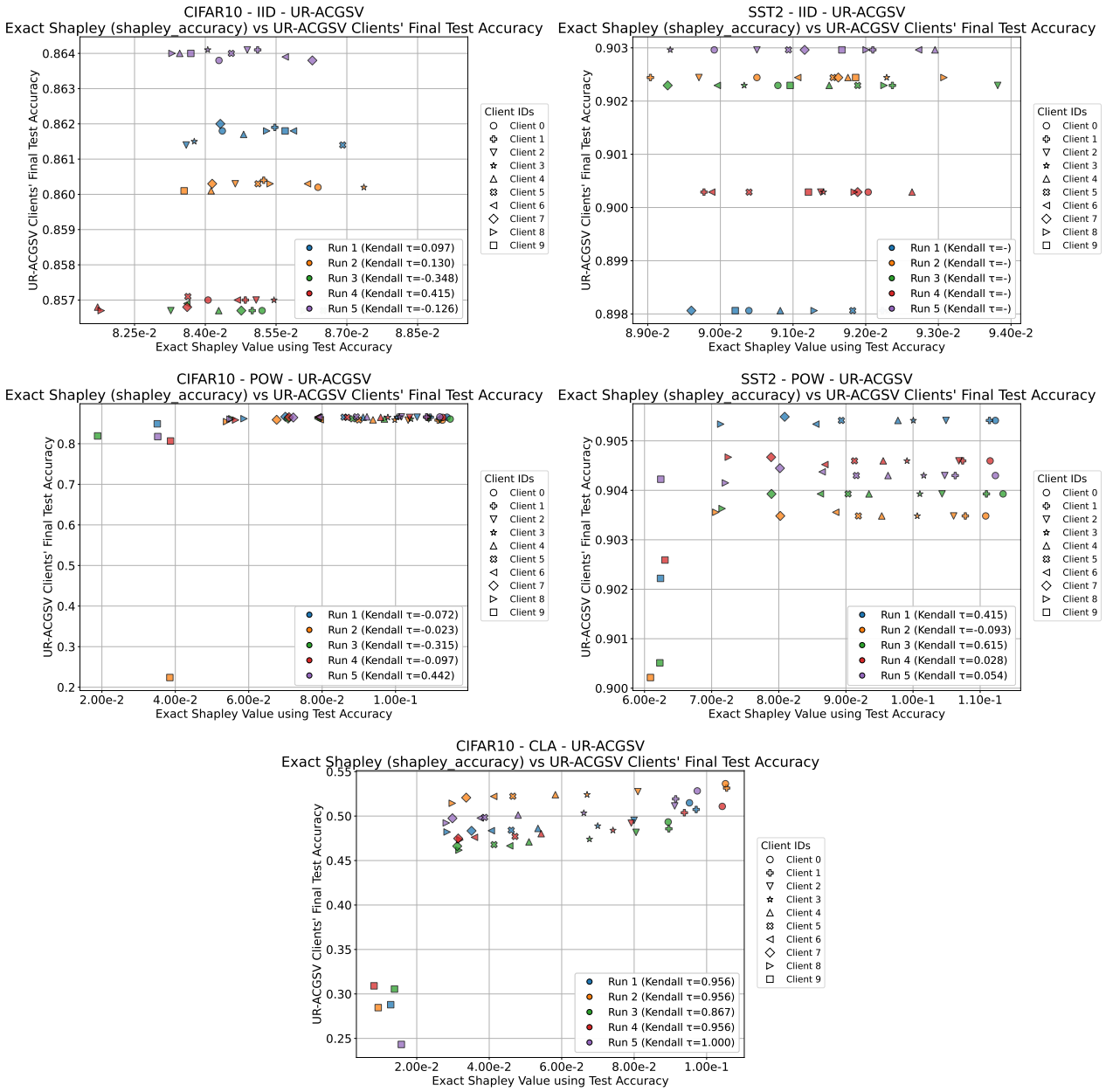


Figure 5.9: Shapley-value-based fairness analysis for UR-ACGSV on CIFAR10 under IID, POW, and CLA data partitioning, and SST under IID and POW data partitioning. The left column reports the UR-ACGSV results on the CIFAR10 dataset, while the right column reports the results on the SST2 dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

5.6 Ablation study results: OTOPK-CFFL Variant

As in Section 5.5, before interpreting the effect of the ablation, I first describe how the experimental results are presented and organized. Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, and 5.16 compare the results obtained with the original CFFL framework and the OTOPK-CFFL variant across the evaluated datasets under POW and CLA partitioning strategies. Each figure is organized in the same comparison grid as stated Section 5.5.

Tables 5.8 and 5.9 report the mean Pearson's r, Kendall's τ, GMD, and MPAD values obtained for the frameworks being compared in this study in the POW and CLA settings, respectively.[5]

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (POW)	CFFL	**0.969**	**0.907**	**1.26**	3.20
	OTOPK-CFFL	0.899	0.854	1.23	**3.87**
FA-MNIST (POW)	CFFL	**0.970**	**0.858**	**1.77**	2.35
	OTOPK-CFFL	0.907	0.467	1.46	**4.70**
CIFAR10 (POW)	CFFL	**0.944**	**0.964**	8.31	40.25
	OTOPK-CFFL	0.701	0.536	**17.22**	**43.68**
SST2 (POW)	CFFL	**0.962**	**0.964**	**4.99**	11.89
	OTOPK-CFFL	0.814	0.745	2.78	**16.54**

Table 5.8: Comparison between CFFL and OTOPK-CFFL under POW data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

[5]The Kendall's τ, GMD and MPAD values computed with standalone test accuracy for each run are with the OTOPK-CFFL reported in Tables D.14, D.15, D.16, and D.17, respectively.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (CLA)	CFFL	**0.987**	**1.000**	**33.44**	7.44
	OTOPK-CFFL	0.746	0.632	5.64	**35.08**
FA-MNIST (CLA)	CFFL	**0.980**	**0.938**	**29.44**	6.14
	OTOPK-CFFL	0.770	0.928	8.82	**31.55**
CIFAR10 (CLA)	CFFL	0.902	0.733	**12.95**	4.25
	OTOPK-CFFL	**0.975**	**0.792**	9.22	**13.60**

Table 5.9: Comparison between CFFL and OTOPK-CFFL under CLA data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

The first observation drawn from the Kendall's τ and GMD values reported in Tables 5.8 and 5.9 is that OTOPK-CFFL exhibits weaker contribution-rank preservation and model differentiation than CFFL. More specifically, OTOPK-CFFL produces lower Kendall's τ values in all settings except CIFAR10 under CLA partitioning and lower GMD values in all settings except CIFAR10 under POW partitioning. These results indicate that the stratified reward sparsity employed by CFFL, which is isolated in OTOPK-CFFL, is not the primary mechanism responsible for the contribution fairness observed in the CFFL experiments presented in Section 4.2. This observation motivates a closer examination of the convergence curves shown in Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, and 5.16.

Across the convergence plots, the clients that consistently converge to models with lower predictive performance are Clients 7, 8, and 9. This pattern is present for all dataset partitioning settings, but is particularly evident in the MNIST CLA, CIFAR10 POW, and SST2 POW settings, shown in Figures 5.11, 5.14, and 5.16, respectively. Furthermore, as shown in the reward-sparsity plots included in these same figures, these clients are assigned the most restrictive reward sparsity levels. Under POW partitioning, their reward sparsities are at least 72.9%, 83.3%, and 93.7%, respectively. Under CLA partitioning, their reward sparsities are at least 70.0%, 80.0%, and 90.0%, respectively. In contrast, the remaining clients receive rewards with sparsity below 70% and converge to models that can achieve identical or very similar test accuracies. These sparsity levels correspond to the complements of the reward density upper bounds reported in Table 5.1.

This ablation study adds empirical evidence that Top-K sparsification is not, by itself, an effective and well-calibrated mechanism for generating contribution-based gradient re-

wards capable of producing both model differentiation and contribution rank preservation. The OTOPK-CFFL results show that Top-K sparsification can affect the final model obtained by each client, but this effect becomes noticeable mainly when clients consistently receive highly sparse rewards. Across the evaluated scenarios, the clients that received rewards with more than 70% sparsity were the ones that converged to models with lower test performance, whereas clients receiving rewards below this sparsity level generally converged to models with similar, or the same, predictive performance. Interestingly, the same pattern can also be observed in the experiment results with ACGSV and UR-ACGSV reported in Section 5.5.

The second observation concerns the MPAD values reported in Tables 5.8 and 5.9. OTOPK-CFFL achieved higher MPAD values than CFFL in all evaluated settings, which shows a trade-off between contribution fairness and average accuracy gain. Although true, it is important to emphasize that the substantial performance degradation under OTOPK-CFFL was concentrated primarily among clients that consistently received the most highly sparsified rewards.

In contrast, clients with intermediate reference contribution values, such as Clients 4, 5, and 6, generally received less restrictive rewards and converged to performance levels comparable to those of the highest contributors, despite their distinct contribution levels. Consequently, the higher MPAD values are driven largely by these insufficiently differentiated clients achieving relatively high federated performance, rather than by a consistent improvement in predictive performance across the federation.

Therefore, the trade-off between accuracy gain and contribution fairness reflects is explained by the convergence of several client models to similar performance plateaus, rather than a deliberate and well-calibrated exchange between the two objectives. This observation is clear in all the OTOPK-CFFL scatter plots showed in Figures 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, and 5.16.

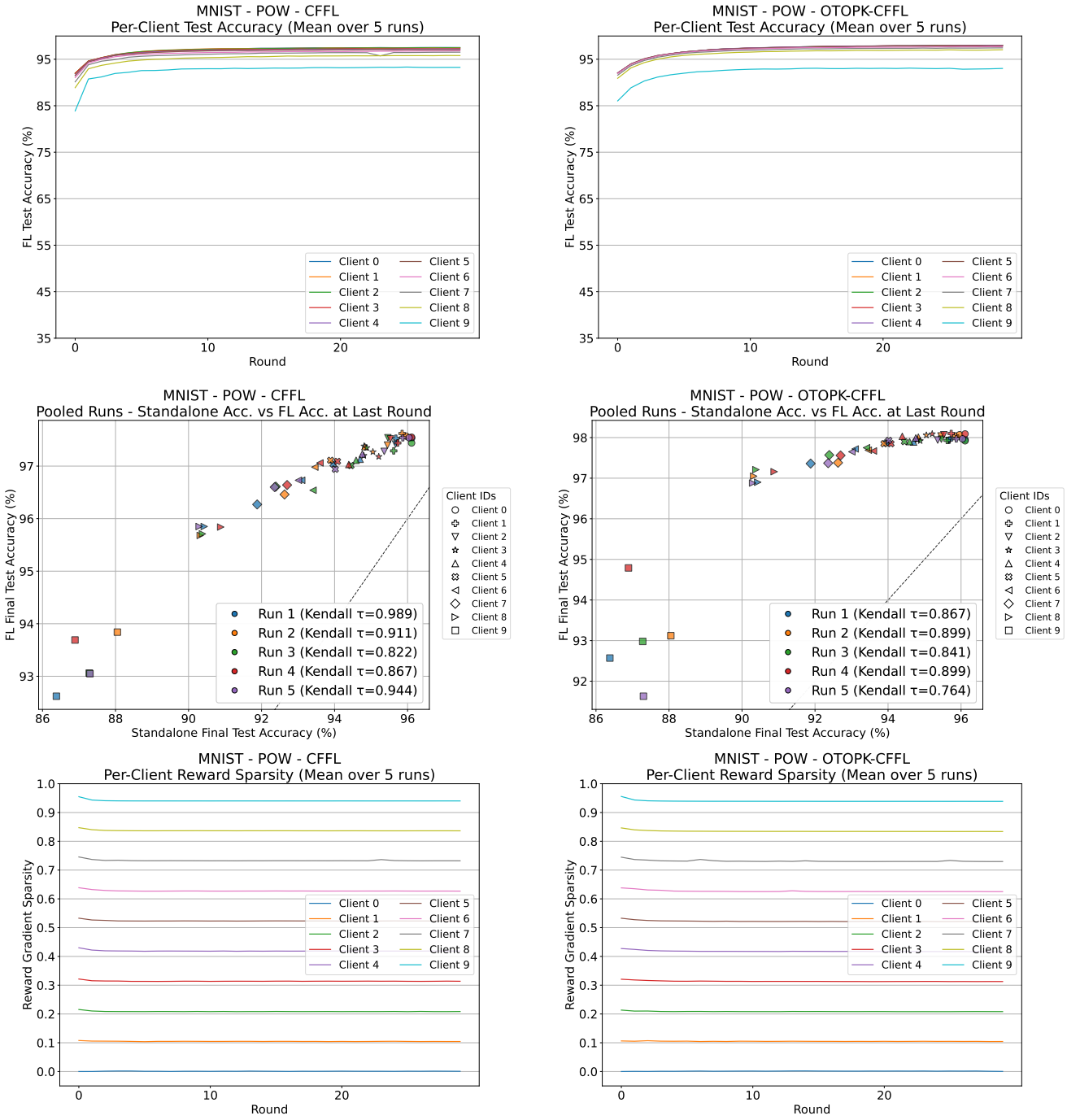


Figure 5.10: Per-client results for the CFFL framework and the OTOPK-CFFL variant on MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

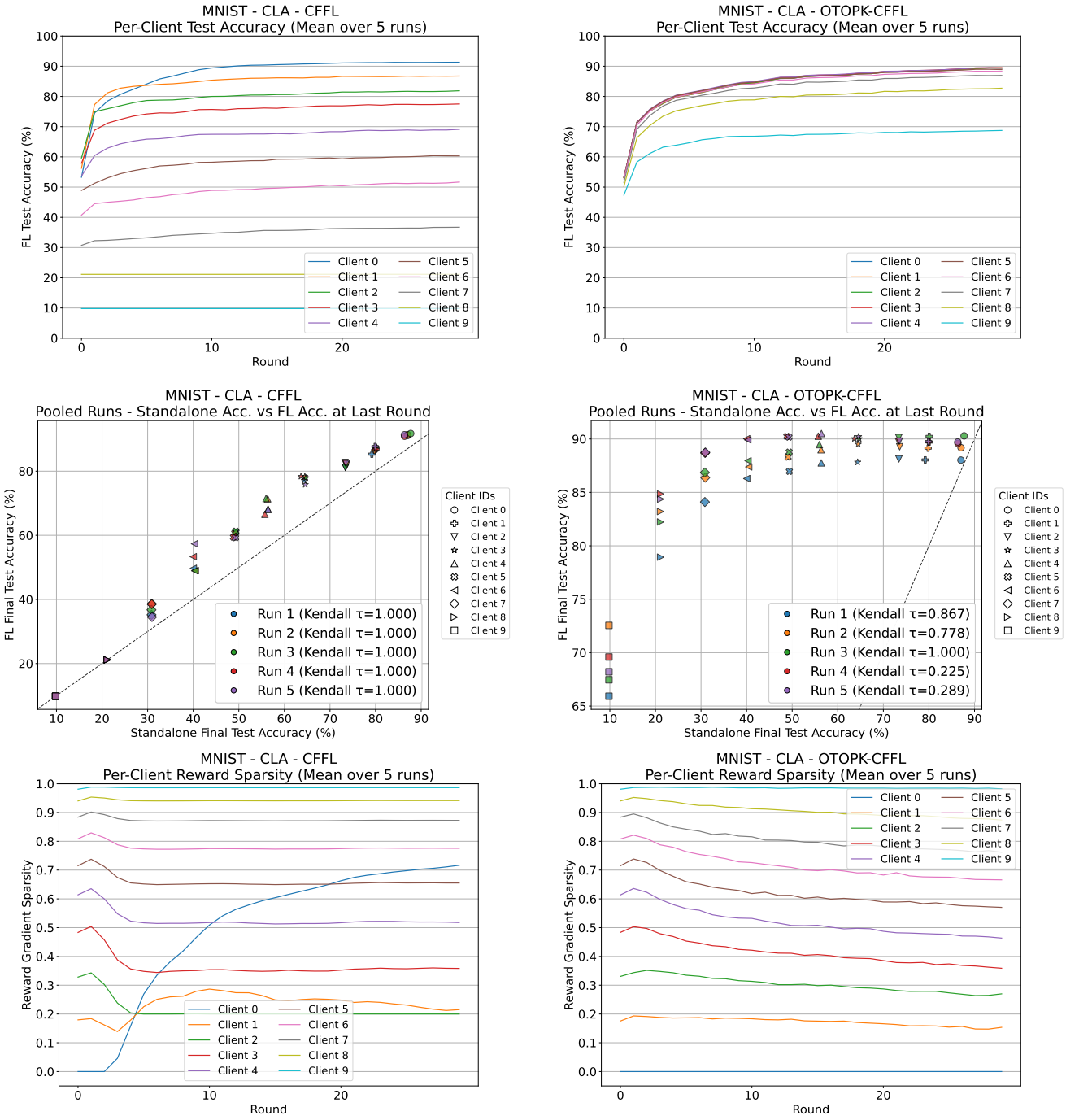


Figure 5.11: Per-client results for the CFFL framework and the OTOPK-CFFL variant on MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

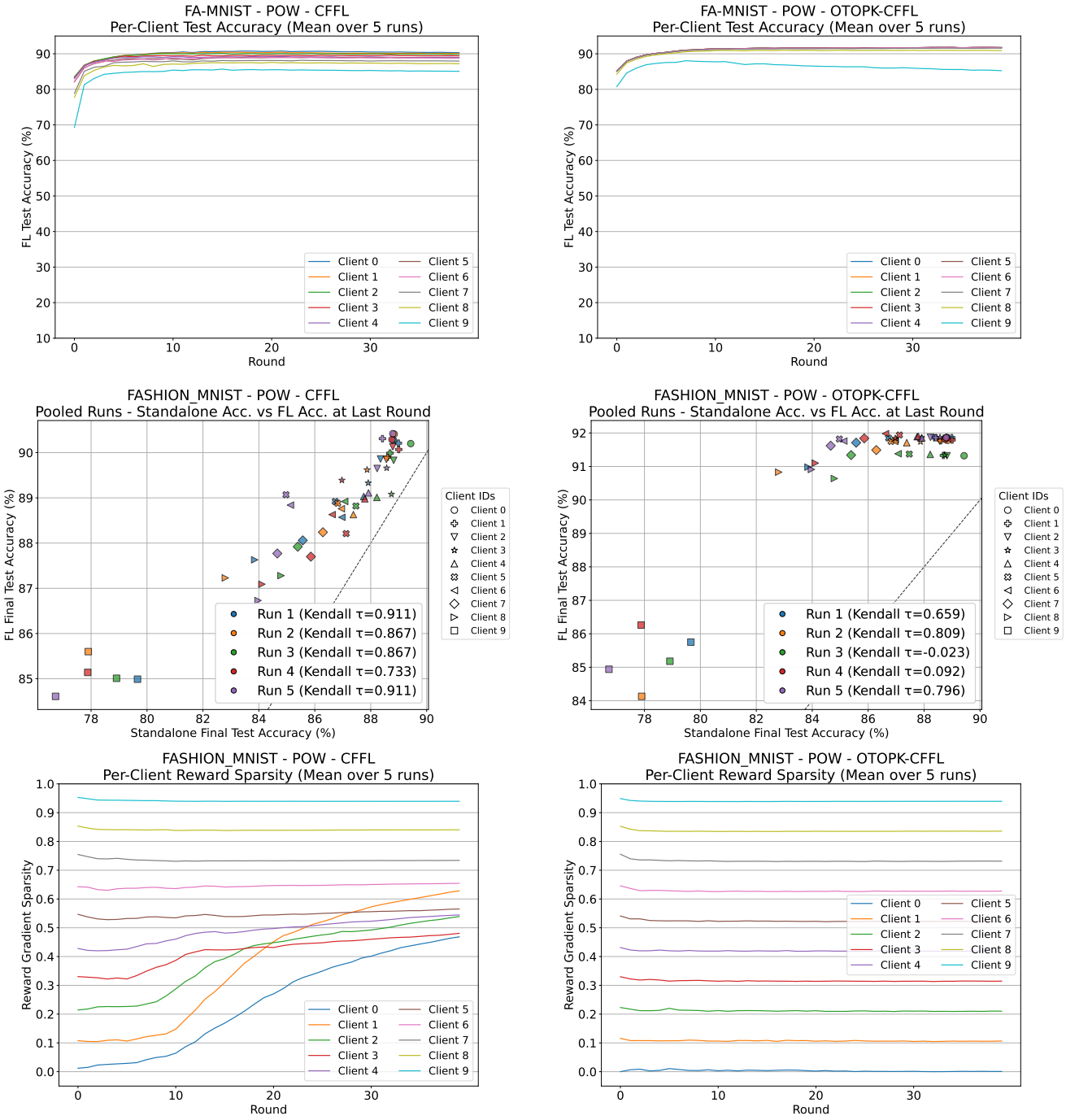


Figure 5.12: Per-client results for the CFFL framework and the OTOPK-CFFL variant on FA-MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

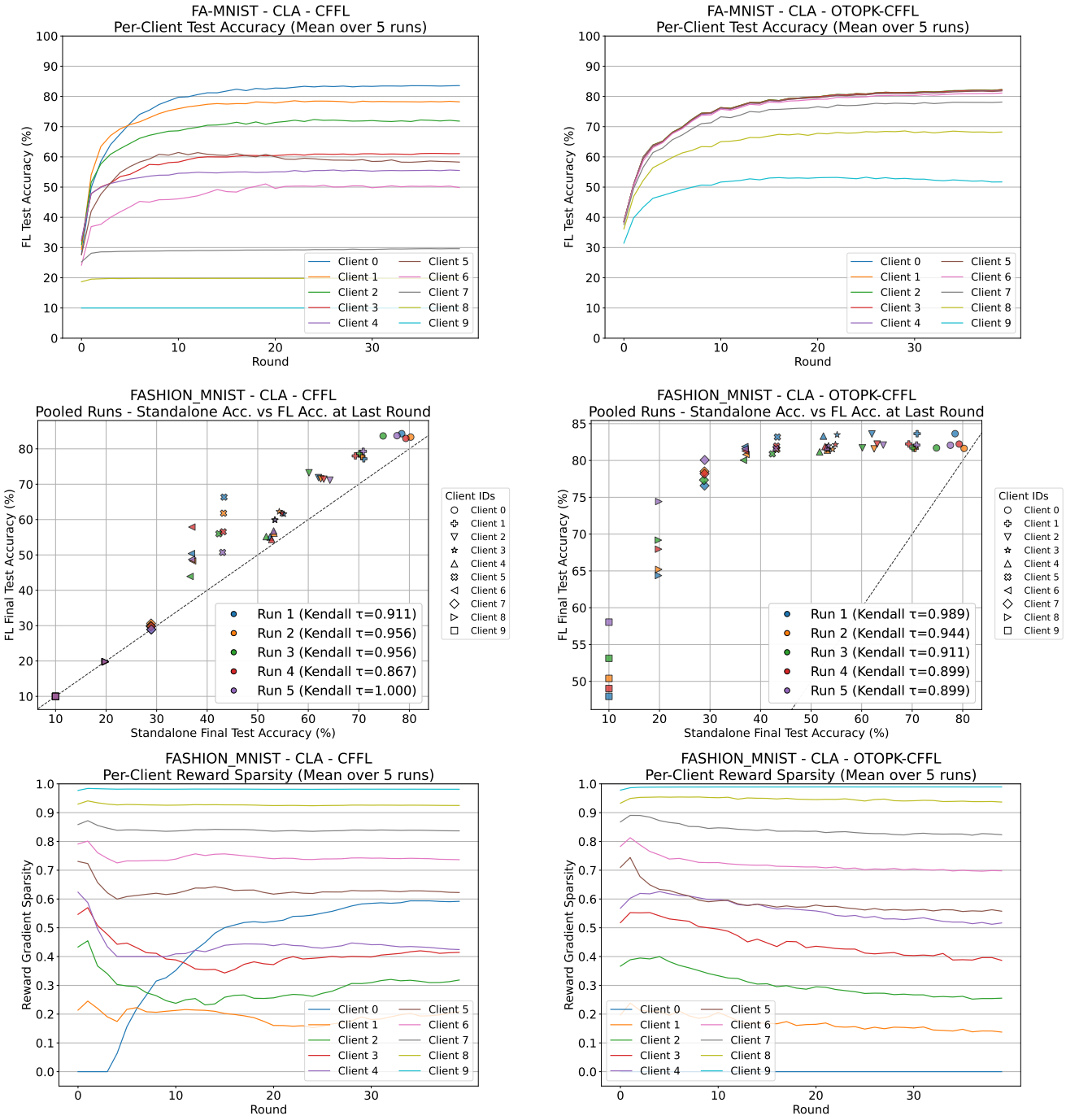


Figure 5.13: Per-client results for the CFFL framework and the OTOPK-CFFL variant on FA-MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

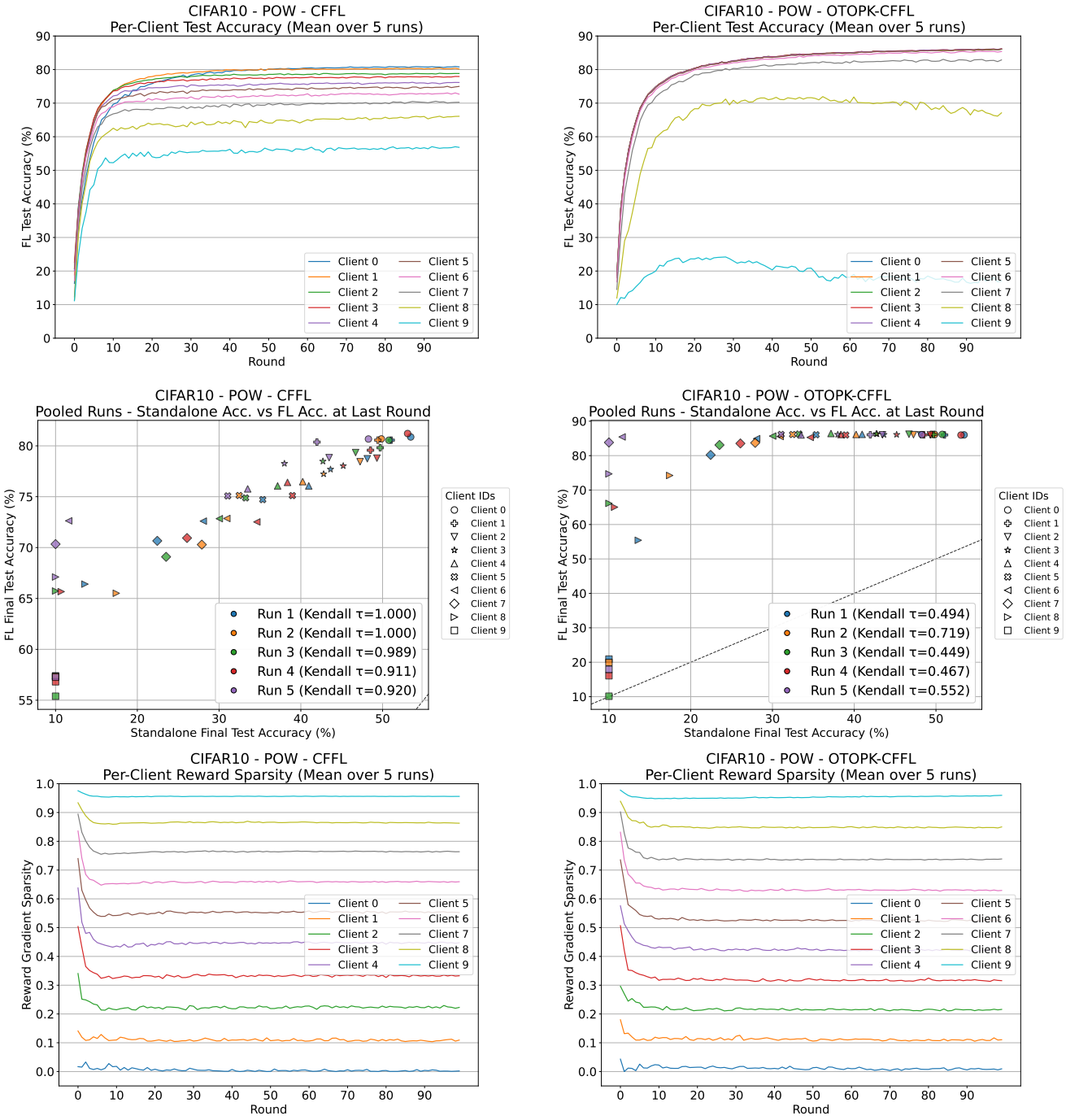


Figure 5.14: Per-client results for the CFFL framework and the OTOPK-CFFL variant on CIFAR10 under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

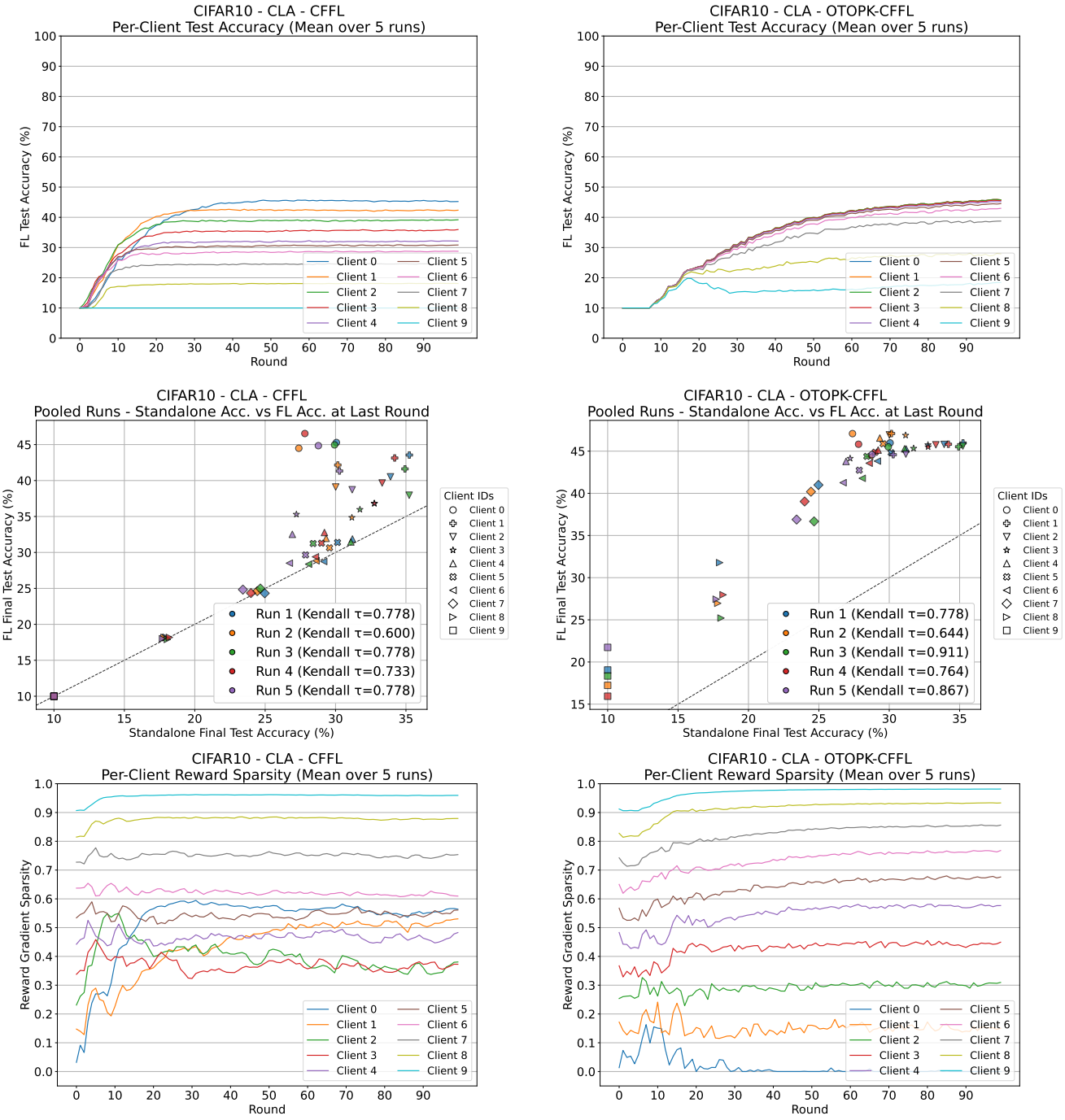


Figure 5.15: Per-client results for the CFFL framework and the OTOPK-CFFL variant on CIFAR10 under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

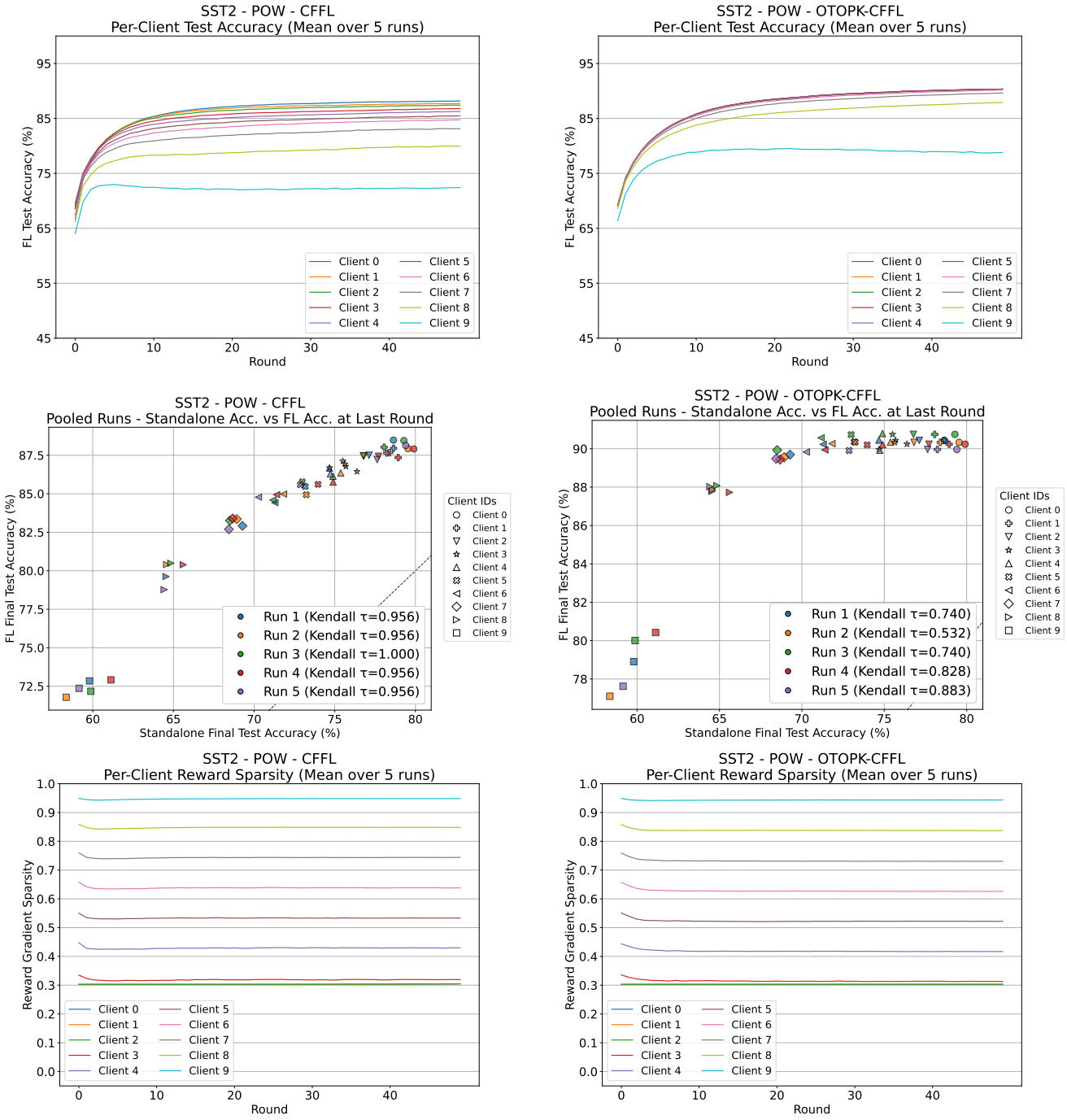


Figure 5.16: Per-client results for the CFFL framework and the OTOPK-CFFL variant on SST2 under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

The observed sparsity level of 70% at which performance degradation, and consequently model differentiation, occurs in OTOPK-CFFL should not be interpreted as a universal threshold, but rather as an empirical transition region under the experimental conditions considered in this thesis. Different data distributions, federation sizes, model architectures, and training hyperparameters could shift the sparsity level at which degradation appears.

The CLA experiments provide a concrete example of this dependence on the training conditions. Although pronounced performance degradation in both the POW and CLA settings generally emerged at sparsity levels of approximately 70% or higher, the class-skewed data distribution in CLA slightly shifted the range over which degradation became visible across datasets.

In MNIST CLA (Figure 5.11), the transition occurs approximately when rewards have sparsity levels ranging between 65% to 55%. In FA-MNIST CLA (Figure 5.13), the transition occurs when reward sparsity ranges from approximately 83% to 70% sparsity, and in CIFAR10 CLA (Figure 5.15), the corresponding transition appears with reward sparsity levels ranging from approximately 65% to 77%. These transition points should not be compared as fixed sparsity thresholds, because the datasets also differ in sample availability and task difficulty. MNIST and FA-MNIST use 600 samples per client, whereas CIFAR10 uses 3000 samples per client, and the visual and semantic complexity of the classification task increases substantially from MNIST to FA-MNIST and CIFAR10.

Nevertheless, the results expose an important limitation of the Top-K reward mechanism. Model differentiation becomes clearly visible only after rewards become highly sparse, indicating that Top-K sparsification does not provide a smooth or proportional mapping between reward sparsity and model convergence. Therefore, if a FL framework designed to promote contribution fairness adopts the Top-K sparsification mechanism to generate contribution-based rewards, its reward-generation pipeline needs to map the client estimated contributions to sufficiently distinct sparsity levels. Otherwise, it either has little effect on model differentiation and rank preservation, as observed from the experiments with ACGSV and UR-ACGSV, or act primarily as a coarse performance-degradation mechanism rather than a fine-grained method for translating contribution differences into gradient rewards of distinct quality, as seen from the experiments with OTOPK-CFFL.

Consistent with the previous experiments, Tables 5.10 and 5.11 report the correlation values obtained using each client's Shapley Value, instead of standalone test accuracy.[6]

[6]The per-run Pearson's r and Kendall's τ are reported in Tables D.18 and D.19.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (POW)	CFFL	**0.958**	**0.915**	**1.26**	3.20
	OTOPK-CFFL	0.878	0.863	1.23	**3.87**
FA-MNIST (POW)	CFFL	**0.983**	**0.938**	**1.77**	2.35
	OTOPK-CFFL	0.786	0.476	1.46	**4.70**
CIFAR (POW)	CFFL	**0.991**	**1.000**	8.31	40.25
	OTOPK-CFFL	0.820	0.576	**17.22**	**43.68**
SST2 (POW)	CFFL	**0.922**	**0.982**	**4.99**	11.89
	OTOPK-CFFL	0.738	0.754	2.78	**16.54**

Table 5.10: Comparison between CFFL and OTOPK-CFFL under POW data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, GMD, and MPAD across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (CLA)	CFFL	**0.873**	**1.000**	**33.44**	7.44
	OTOPK-CFFL	0.542	0.632	5.64	**35.08**
FA-MNIST (CLA)	CFFL	**0.871**	**0.938**	**29.44**	6.14
	OTOPK-CFFL	0.546	0.928	8.82	**31.55**
CIFAR (CLA)	CFFL	**0.954**	**0.947**	**12.95**	4.25
	OTOPK-CFFL	0.746	0.927	9.22	**13.60**

Table 5.11: Comparison between CFFL and OTOPK-CFFL under CLA data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, GMD, and MPAD across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

Figures 5.17 and 5.18 report the scatter plots obtained from using the Shapley Values as their reference contribution values.

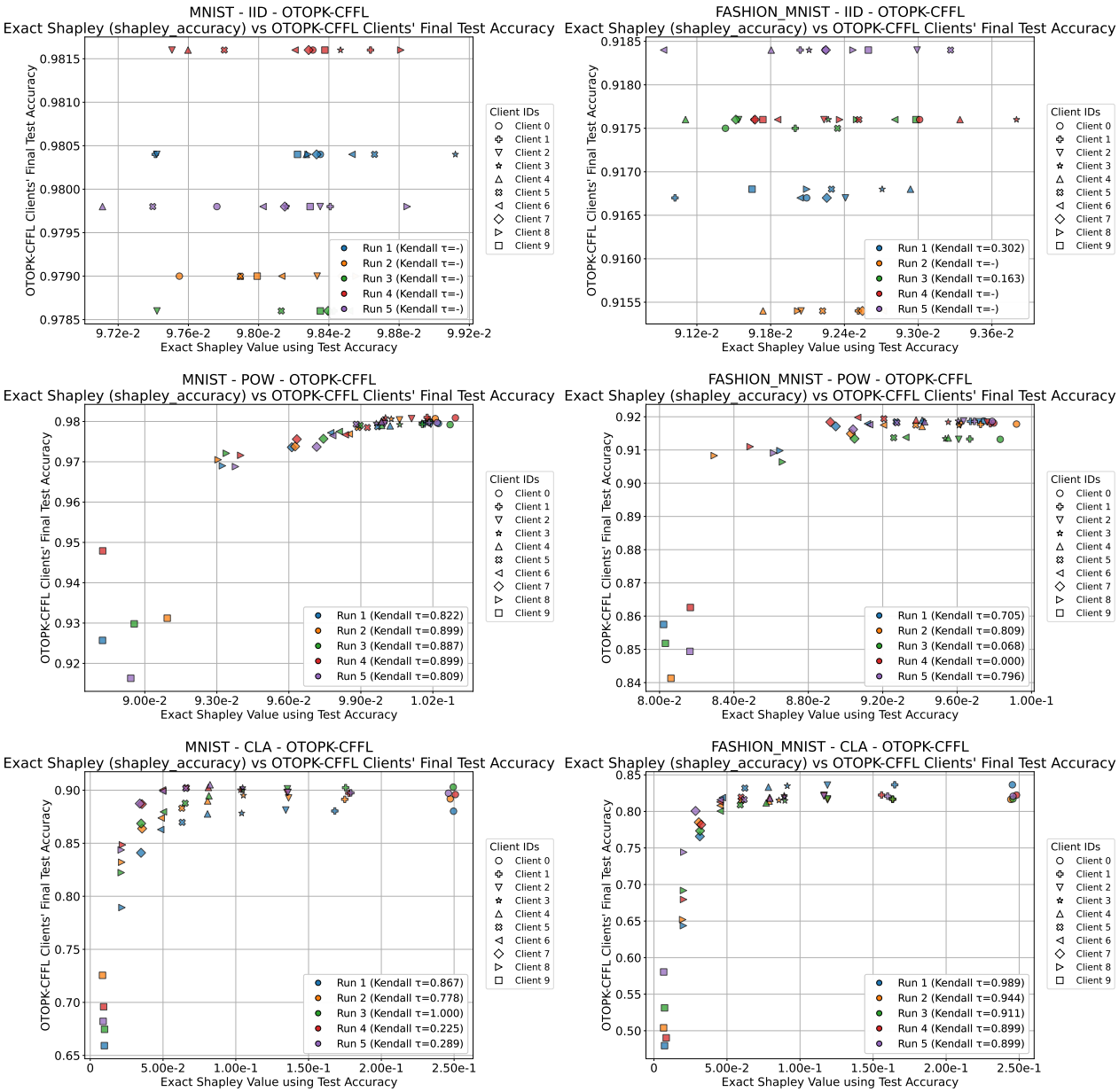


Figure 5.17: Shapley-value-based fairness analysis for OTOPK-CFFL on MNIST and FA-MNIST under IID, POW, and CLA data partitioning. The left column reports the OTOPK-CFFL results on the MNIST dataset, while the right column reports the results on the FA-MNIST dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

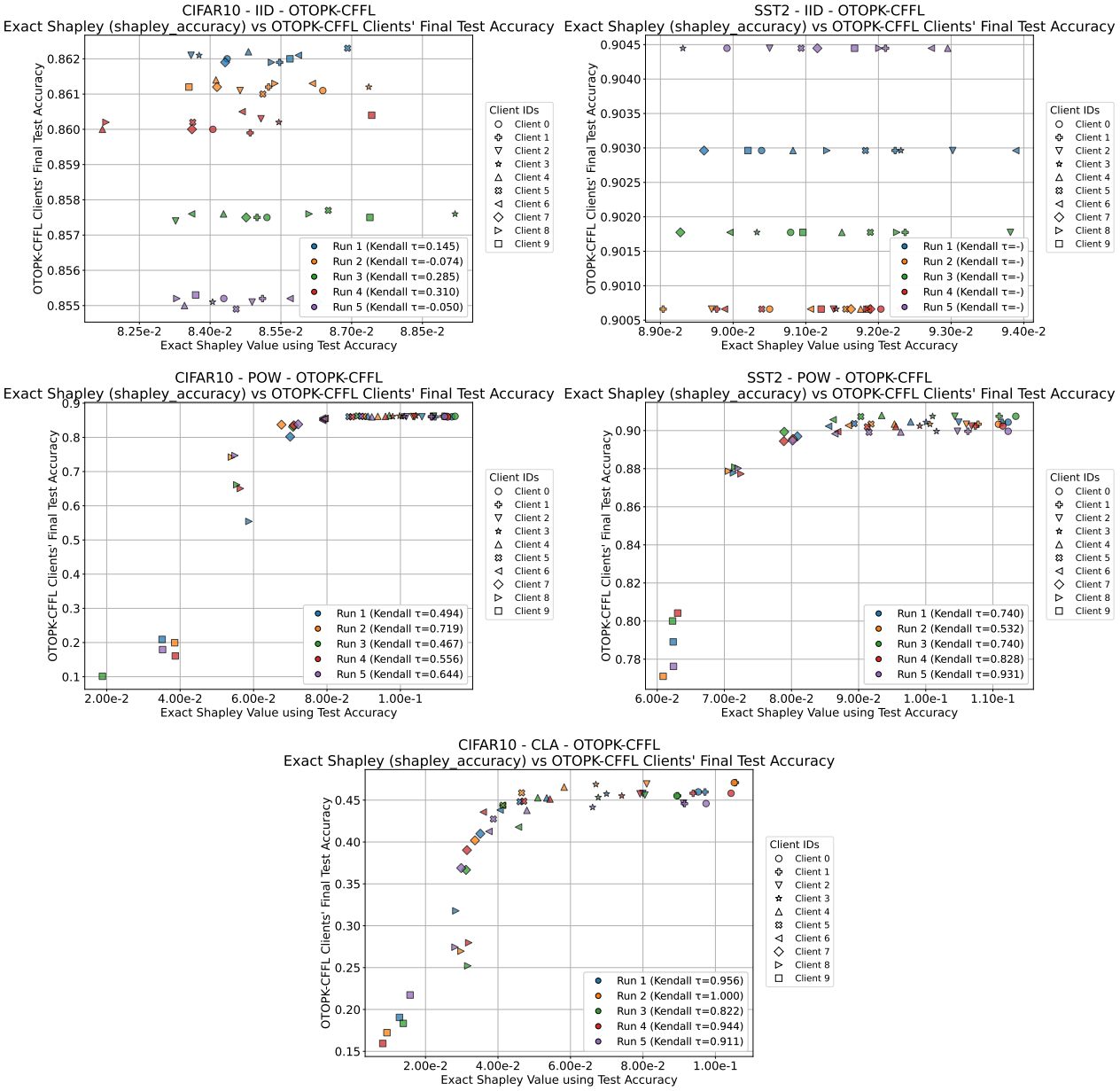


Figure 5.18: Shapley-value-based fairness analysis for OTOPK-CFFL on CIFAR10 under IID, POW, and CLA data partitioning, and SST under IID and POW data partitioning. The left column reports the OTOPK-CFFL results on the CIFAR10 dataset, while the right column reports the results on the SST2 dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

Overall, replacing standalone accuracy with Shapley values as the reference contribution

signal does not fundamentally alter the interpretation of the OTOPK-CFFL results presented previously. Moreover, when Shapley values are used to establish the reference contribution ranking, the CIFAR CLA setting, which was the only scenario in which OTOPK-CFFL had previously appeared to achieve stronger rank preservation results than CFFL, no longer exhibits this advantage. Since GMD and MPAD are independent of the reference contribution ranking and are computed directly from the clients' federated test accuracies, their interpretation remains unchanged. Therefore, these results continue to support the conclusion that Top-K sparsification behaves more like a coarse reward-degradation mechanism than a smoothly calibrated mechanism for translating contribution differences into proportionally differentiated rewards.

Although the experiments did not exhaustively evaluate sparsity levels beyond those assigned to Clients 7, 8, and 9, extending the rewards to even higher sparsity is not necessary to establish the central conclusion of this ablation study. These clients already exhibit a consistent pattern in which increasing reward sparsity is accompanied by greater performance degradation and more pronounced model differentiation. Therefore, the main finding is not that Top-K sparsification is incapable of generating differentiated rewards; rather, meaningful differentiation emerges only when the rewards become sufficiently sparse.

5.7 Ablation study results: DPR-CFFL Variant

As in Section 5.5, before interpreting the effect of the ablation, I first describe how the experimental results are presented and organized. Figures 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, and 5.25 compare the results obtained with the original CFFL framework and the DPR-CFFL variant across the evaluated datasets under POW and CLA partitioning strategies. Each figure is organized in the same comparison grid as stated Section 5.5.

Tables 5.12 and 5.13 report the mean Pearson's r, Kendall's τ, GMD, and MPAD values obtained for the frameworks being compared in this study in the POW and CLA settings, respectively.[7]

As discussed in Subsection 5.4.3, the DPR-CFFL variant isolates the locally accumulated gradient retention to determine whether the contribution fairness results obtained by CFFL are driven by this contribution-independent mechanism. In this variant, all clients receive the same aggregated gradient computed by the server, therefore any remaining

[7]The Pearson's r, Kendall's τ, GMD and MPAD values computed with standalone test accuracy for each run are with the DPR-CFFL reported in Tables D.20, D.21, D.22, and D.23, respectively.

model differentiation cannot be attributed to the Top-K reward allocation. Instead it is produced by the local gradient portion that is retained by each client.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (POW)	CFFL	**0.969**	**0.907**	**1.26**	3.20
	DPR-CFFL	0.950	0.870	0.55	**3.59**
FA-MNIST (POW)	CFFL	**0.970**	**0.858**	**1.77**	2.35
	DPR-CFFL	0.894	0.822	1.34	**2.56**
CIFAR10 (POW)	CFFL	0.944	**0.964**	**8.31**	40.25
	DPR-CFFL	**0.979**	**0.964**	6.03	**41.67**
SST2 (POW)	CFFL	0.962	0.964	**4.99**	11.89
	DPR-CFFL	**0.984**	**0.982**	2.31	**13.28**

Table 5.12: Comparison between CFFL and DPR-CFFL under POW data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric; tied values are both highlighted.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (CLA)	CFFL	**0.987**	**1.000**	**33.44**	7.44
	DPR-CFFL	0.974	0.991	30.90	**9.80**
FA-MNIST (CLA)	CFFL	**0.980**	0.938	**29.44**	6.14
	DPR-CFFL	0.973	**0.956**	28.89	**7.81**
CIFAR10 (CLA)	CFFL	**0.902**	**0.733**	12.95	4.25
	DPR-CFFL	0.899	0.731	**13.09**	**4.44**

Table 5.13: Comparison between CFFL and DPR-CFFL under CLA data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

The first observation drawn from the Kendall's τ and GMD values reported in Ta-

bles 5.12 and 5.13 is that DPR-CFFL produces fairness outcomes that closely match those of CFFL. More specifically, the Kendall's τ values obtained with DPR-CFFL are nearly identical to those obtained with CFFL, while its GMD values are consistently, but only slightly, lower. Thus, DPR-CFFL preserves the reference contribution ranking to approximately the same extent as CFFL and produces a comparable degree of model differentiation, despite not distributing estimated-contribution-based Top-K sparsified rewards. This result indicates that these rewards are not necessary to reproduce the main contribution-fairness outcomes observed with CFFL.

This observation is further supported by the convergence curves shown in Figures 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, and 5.25. Under DPR-CFFL, the client models converge to distinct levels of predictive performance, following convergence patterns that closely resemble those observed under CFFL. Therefore, the model differentiation produced by CFFL remains largely preserved even after its estimated-contribution-based Top-K reward mechanism is removed.

Taken together, this ablation study brings evidence that local gradient retention is not only sufficient to produce client-level model differentiation, but also the main driver of the high fairness outcomes obtained by CFFL with both evaluation protocols used in this thesis: Pearson's r, and the joint Kendall's τ and GMD analysis.

Although all clients receive the same aggregated gradient, this is not equivalent to the model synchronization performed in traditional FL [1, 3, 4, 6]. The common update aligns their learning direction but does not suppress the differences that are originated from their local optimization trajectories. Since the locally accumulated gradients are computed from heterogeneous local datasets, they generally point in distinct optimization directions, and the constant retention causes the client models to gradually diverge.

A second observation that can be drawn from the comparison between CFFL and DPR-CFFL convergence plots in Figures 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, and 5.25 is that most clients follow nearly identical convergence trajectories in both frameworks. However, in CFFL, visible performance degradation is mostly occurring at Clients 7, 8, and 9, which is particularly evident in MNIST CLA (Figure 5.20), FA-MNIST CLA (Figure 5.22), CIFAR10 POW (Figure 5.23), and SST2 POW 5.25), which are the clients that in CFFL are receiveing sufficiently sparsified rewards, as discussed in Section 5.6.

The results of the ablation study presented in this section reinforce the conclusions drawn from the OTOPK-CFFL experiments in Section 5.6. Specifically, Top-K sparsification is not, by itself, an effective mechanism for producing fine-grained model differentiation and preserving the reference contribution ranking. Instead, within CFFL's reward-generation pipeline, it operates primarily as a coarse performance-degradation mechanism.

Taken together, local-gradient retention is the central mechanism behind the CFFL fairness outcomes reported in Section 4.2. It is important to emphasize that local-gradient retention is not unique to CFFL, because other contribution-aware FL frameworks also allow clients to retain their locally accumulated gradients during training, or rely on the server to incorporate clients' accumulated local gradients into their personalized rewards [7, 22, 30].

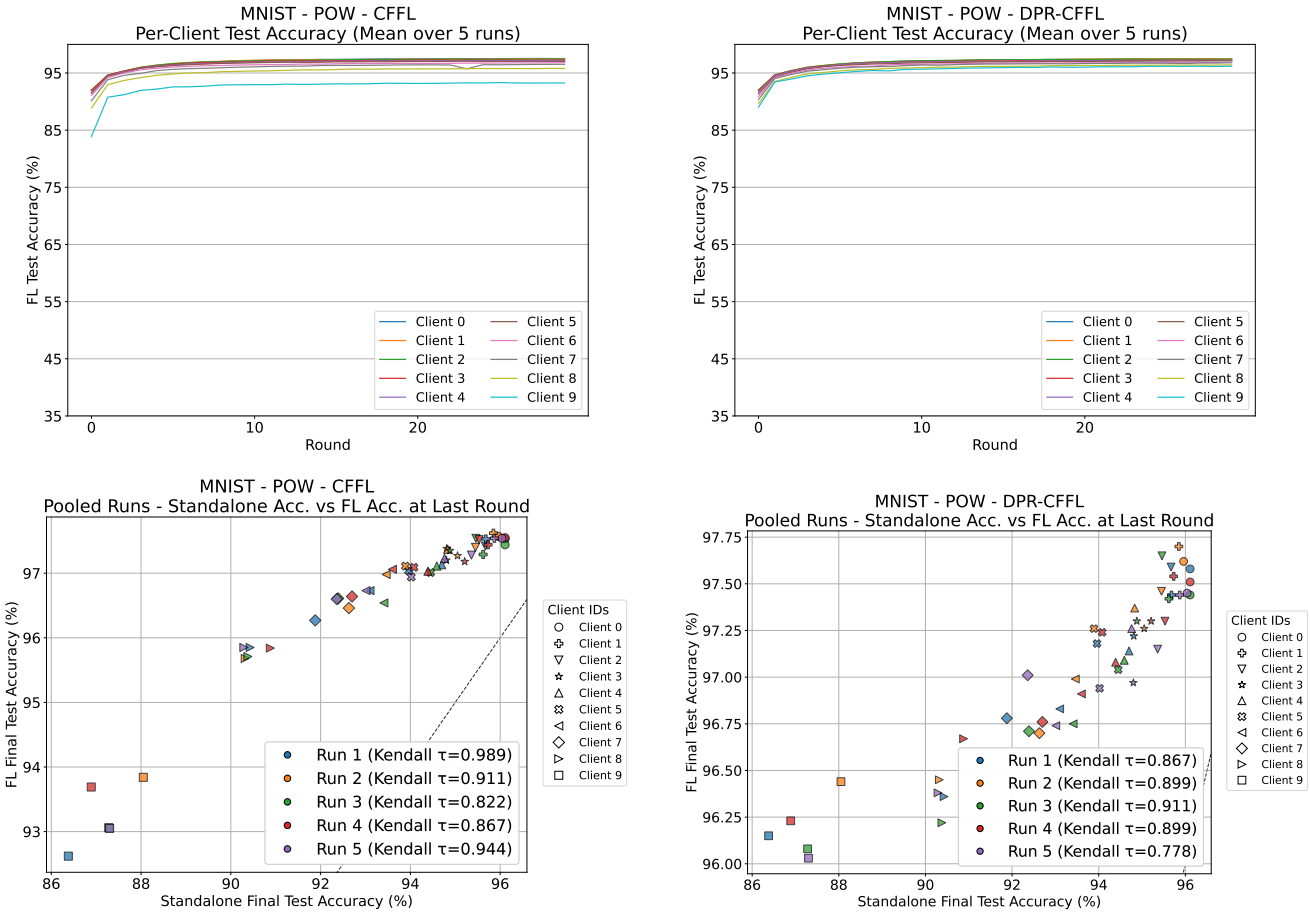


Figure 5.19: Per-client results for the CFFL framework and the DPR-CFFL variant on MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

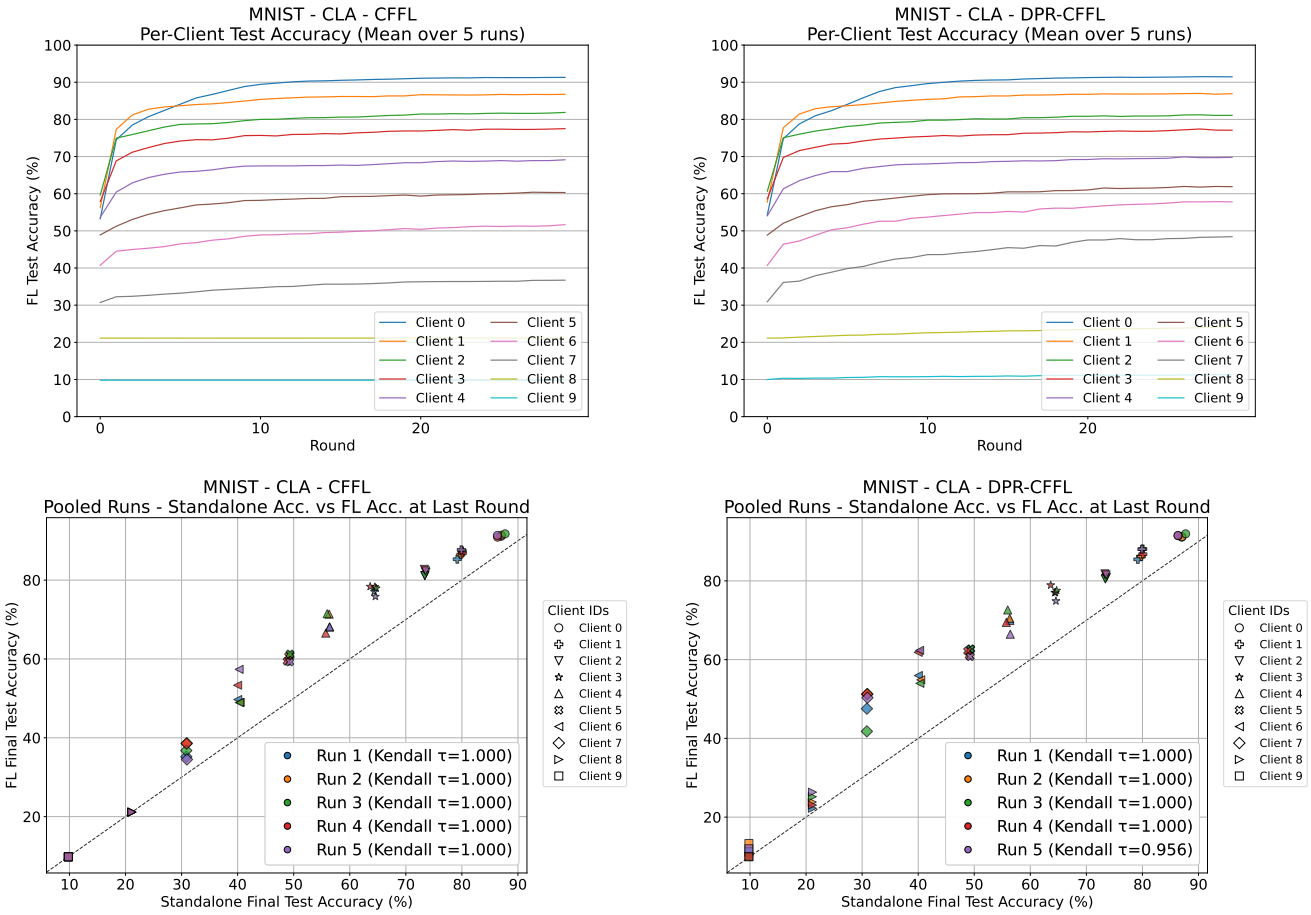


Figure 5.20: Per-client results for the CFFL framework and the DPR-CFFL variant on MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

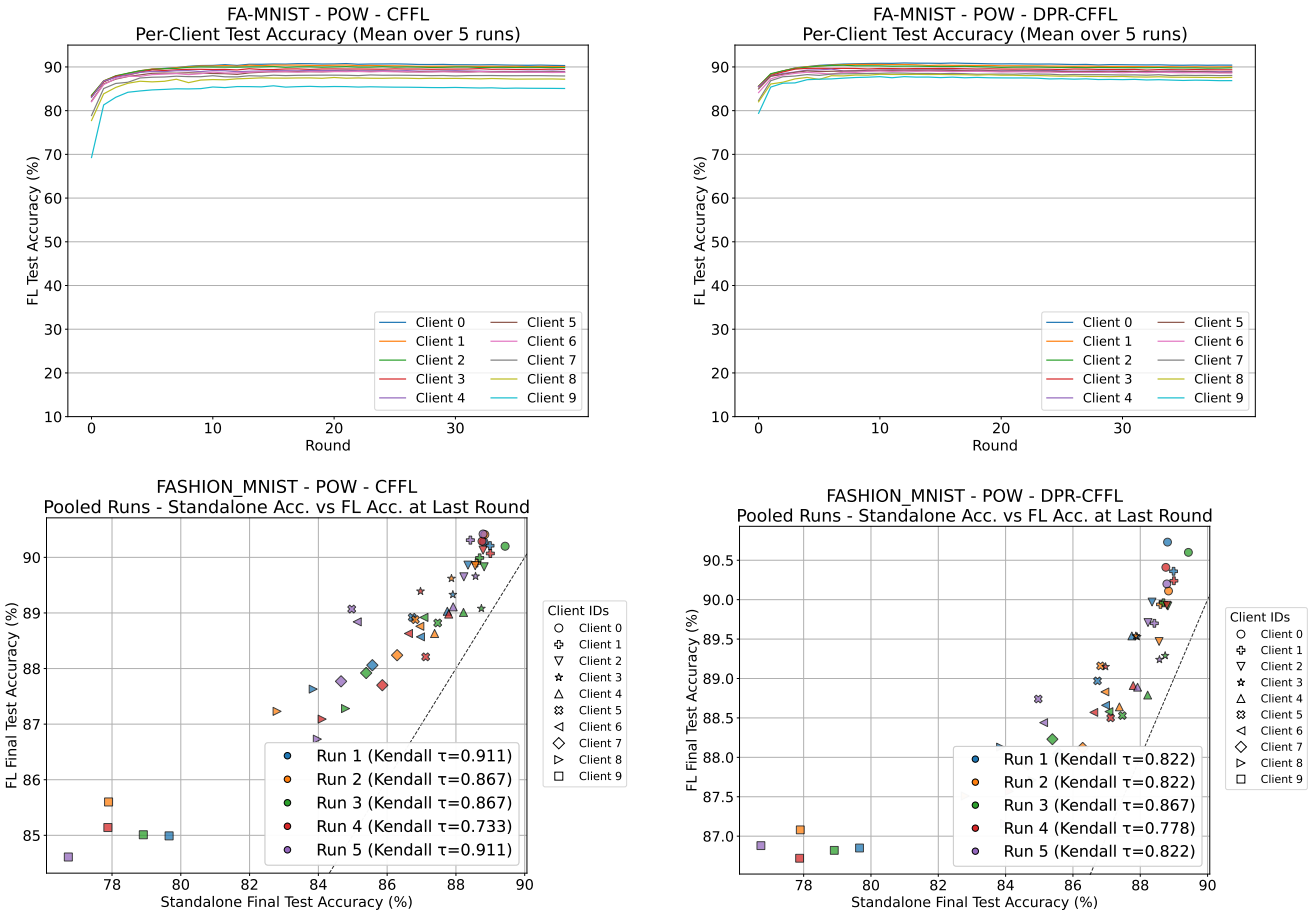


Figure 5.21: Per-client results for the CFFL framework and the DPR-CFFL variant on FA-MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

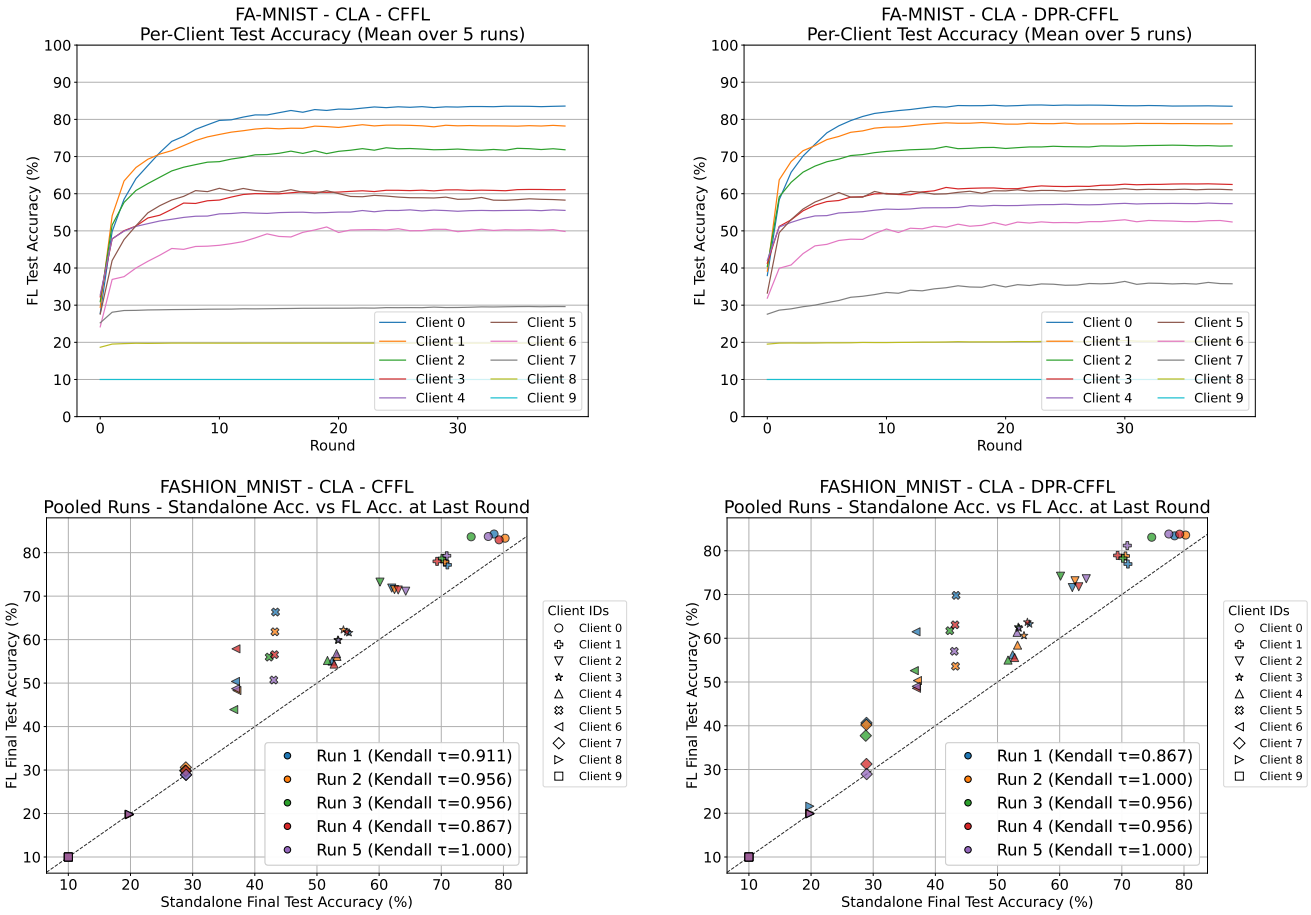


Figure 5.22: Per-client results for the CFFL framework and the DPR-CFFL variant on FA-MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

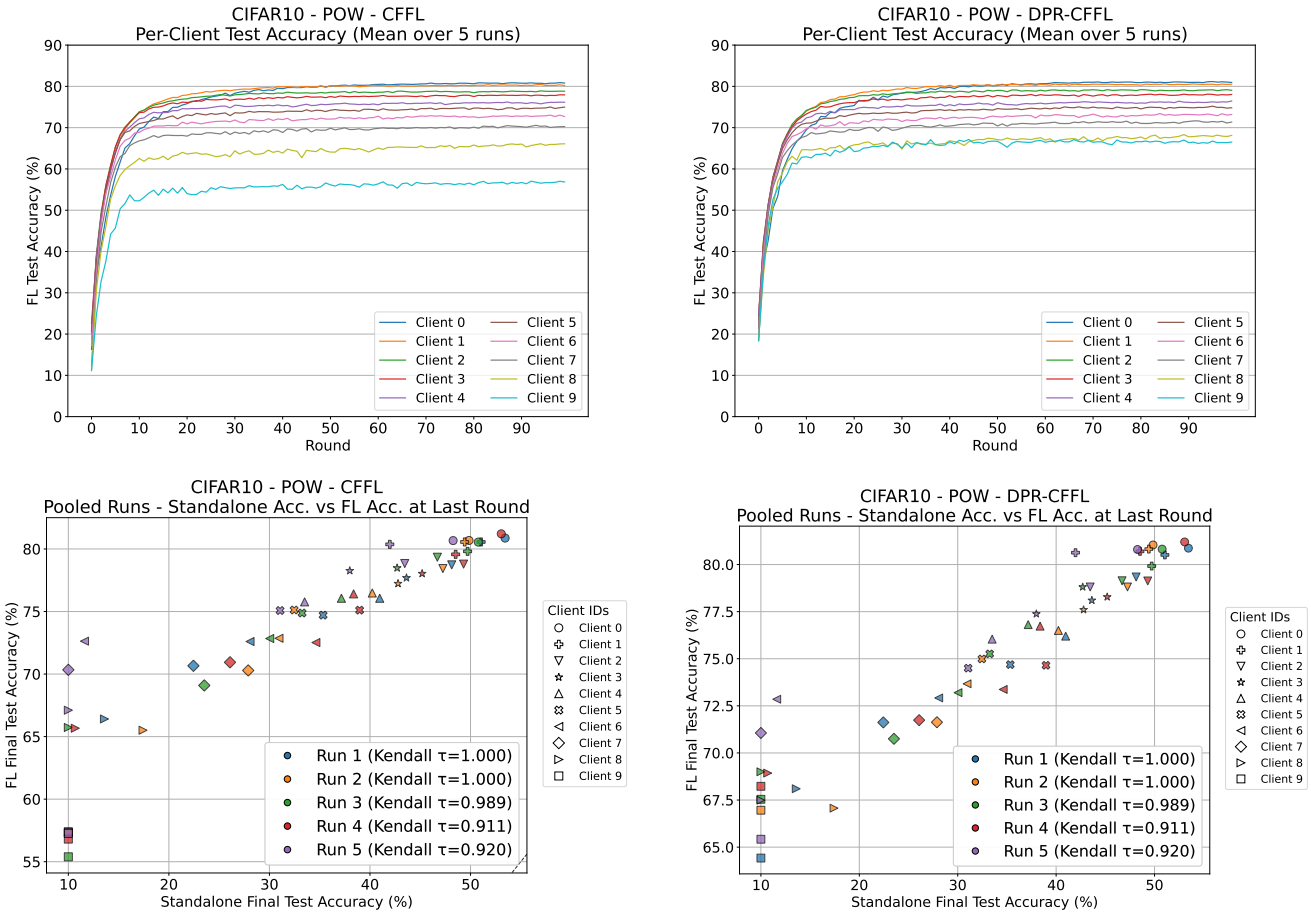


Figure 5.23: Per-client results for the CFFL framework and the DPR-CFFL variant on CIFAR10 under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

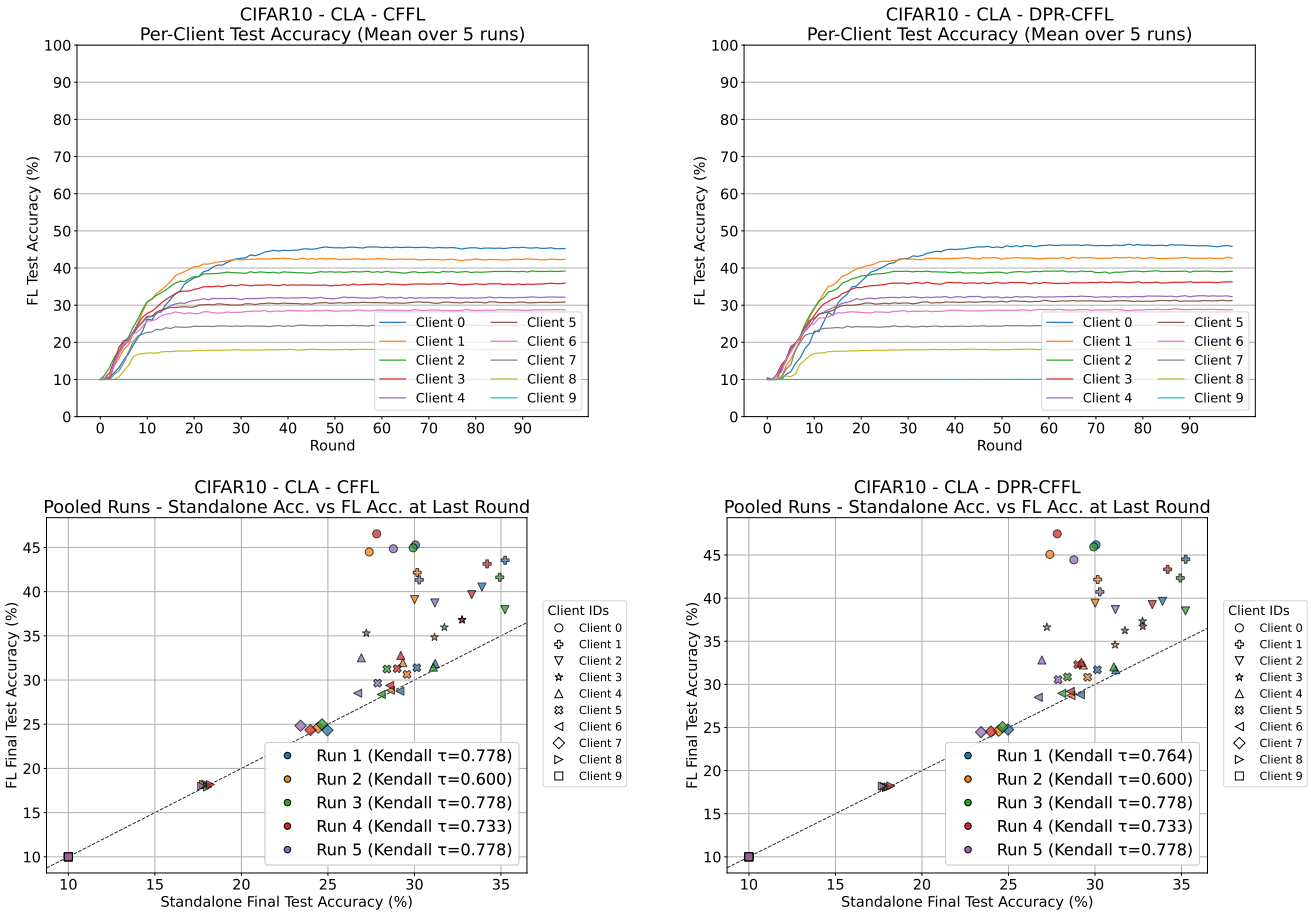


Figure 5.24: Per-client results for the CFFL framework and the DPR-CFFL variant on CIFAR10 under CLA data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

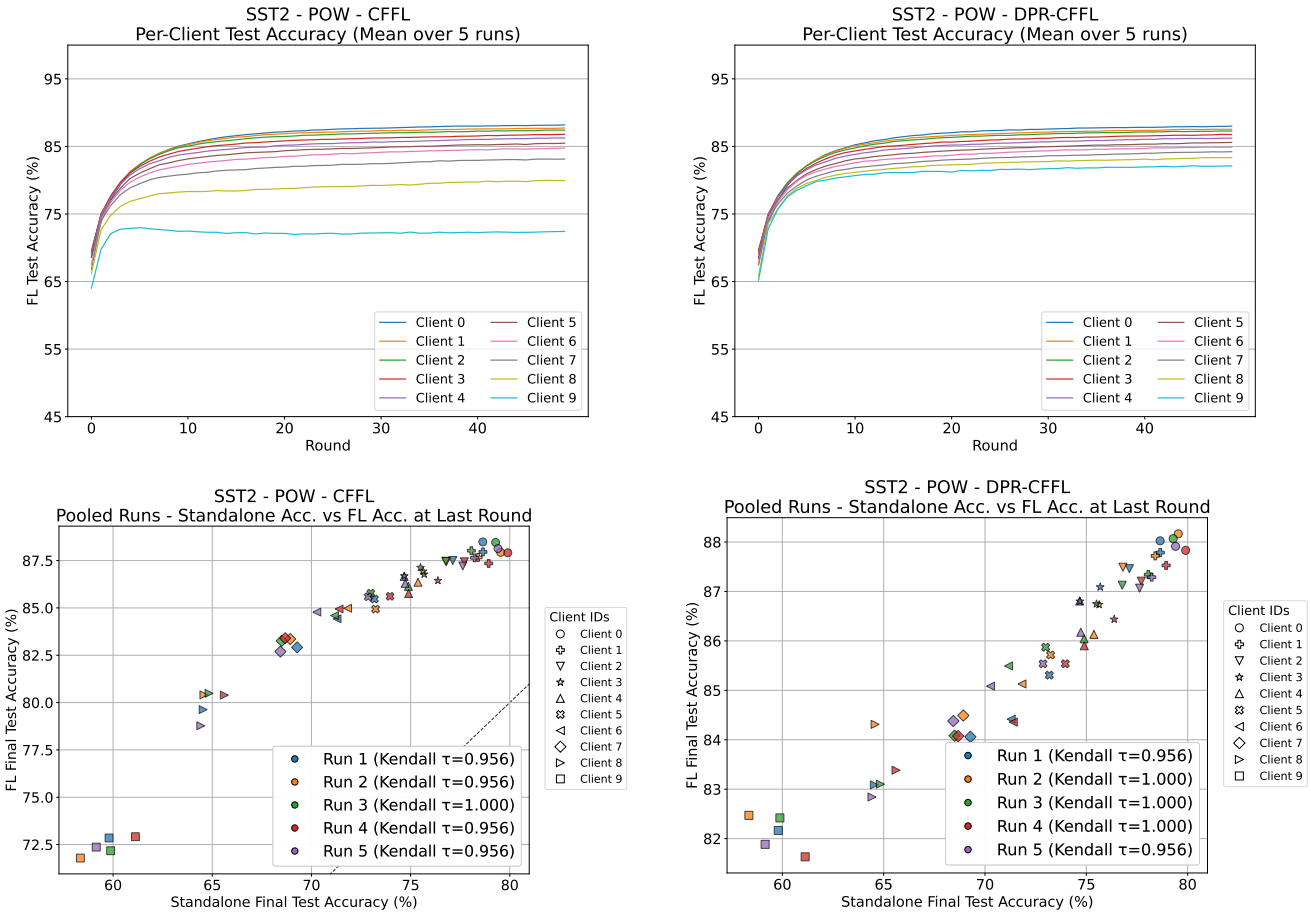


Figure 5.25: Per-client results for the CFFL framework and the DPR-CFFL variant on SST2 under POW data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

Consistent with the previous experiments, Tables 5.14 and 5.15 report the correlation values obtained using each client's Shapley Value, instead of standalone test accuracy.[8] The corresponding scatter plots are shown in Figures 5.26 and 5.27. Overall, replacing standalone accuracy with Shapley Values as the reference contribution signal does not fundamentally change the interpretation of DPR-CFFL.

[8]The per-run Pearson's r and Kendall's τ are reported in Tables D.24 and D.25.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (POW)	CFFL	0.958	**0.915**	**1.26**	3.20
	DPR-CFFL	**0.964**	0.897	0.55	**3.59**
FA-MNIST (POW)	CFFL	**0.983**	**0.938**	**1.77**	2.35
	DPR-CFFL	0.965	0.920	1.34	**2.56**
CIFAR-10 (POW)	CFFL	**0.991**	**1.000**	**8.31**	40.25
	DPR-CFFL	0.986	**1.000**	6.03	**41.67**
SST-2 (POW)	CFFL	0.922	0.982	**4.99**	11.89
	DPR-CFFL	**0.992**	**1.000**	2.31	**13.28**

Table 5.14: Comparison between CFFL and DPR-CFFL under POW data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, Gini Mean Difference (GMD), and Mean Paired Accuracy Difference (MPAD) across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric; tied values are both highlighted.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (CLA)	CFFL	**0.873**	**1.000**	**33.44**	7.44
	DPR-CFFL	0.855	0.991	30.90	**9.80**
FA-MNIST (CLA)	CFFL	**0.871**	0.938	**29.44**	6.14
	DPR-CFFL	0.851	**0.956**	28.89	**7.81**
CIFAR-10 (CLA)	CFFL	**0.954**	**0.947**	12.95	4.25
	DPR-CFFL	**0.954**	0.944	**13.09**	**4.44**

Table 5.15: Comparison between CFFL and DPR-CFFL under CLA data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, Gini Mean Difference (GMD), and Mean Paired Accuracy Difference (MPAD) across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric; tied values are both highlighted.

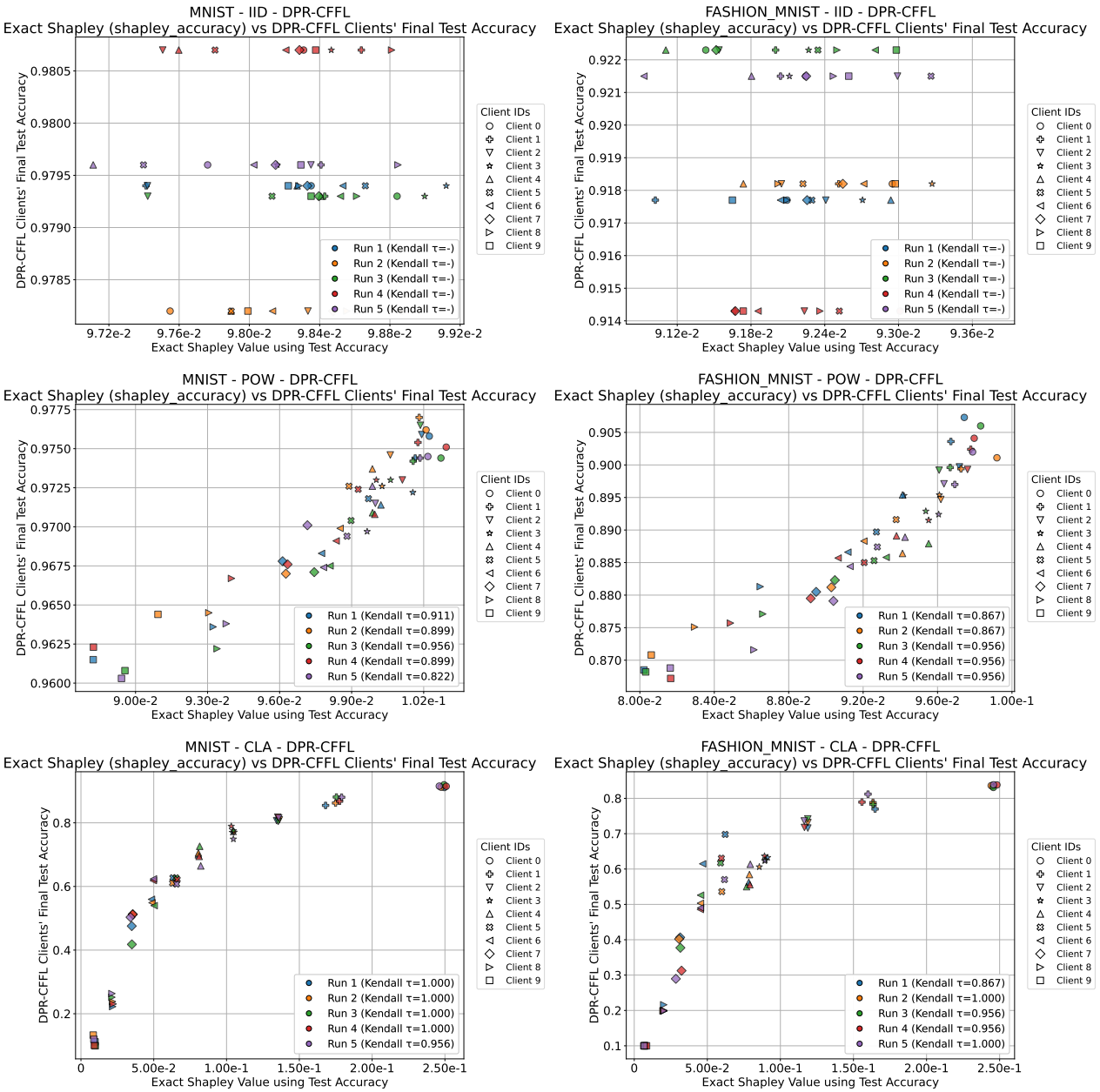


Figure 5.26: Shapley-value-based fairness analysis for DPR-CFFL on MNIST and FA-MNIST under IID, POW, and CLA data partitioning. The left column reports the DPR-CFFL results on the MNIST dataset, while the right column reports the results on the FA-MNIST dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

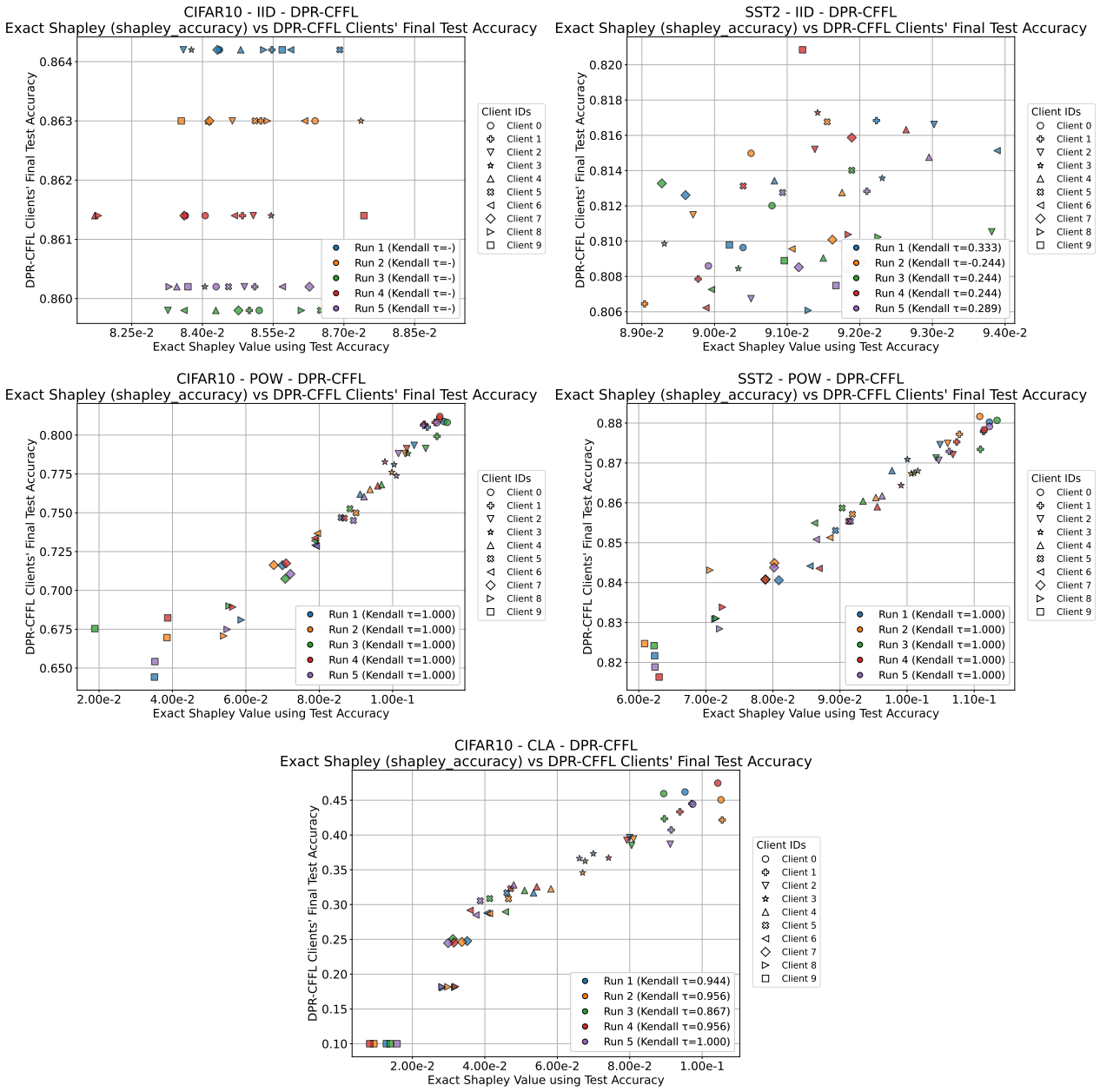


Figure 5.27: Shapley-value-based fairness analysis for DPR-CFFL on CIFAR10 under IID, POW, and CLA data partitioning, and SST under IID and POW data partitioning. The left column reports the DPR-CFFL results on the CIFAR10 dataset, while the right column reports the results on the SST2 dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

5.8 Ablation study results: DFR-ACGSV Variant

As in Section 5.5, before interpreting the effect of the ablation, I first describe how the experimental results are presented and organized. Figures 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, and 5.34 compare the results obtained with the original ACGSV framework and the DFR-ACGSV variant across the evaluated datasets under POW and CLA partitioning strategies. Each figure is organized in the same comparison grid as stated Section 5.5. Tables 5.16 and 5.17 report the mean Pearson's r, Kendall's τ, GMD, and MPAD values for all frameworks and framework variants evaluated in this thesis.[9]

As discussed in Subsection 5.4.4, the DFR-ACGSV variant was designed to test whether adding local-gradient retention to ACGSV produces similar model differentiation and contribution rank preservation patterns observed in DPR-CFFL. The key distinction between the two variants is how much of the locally accumulated gradient is retained during the client model update.

In DPR-CFFL, each client retains only a dataset-dependent portion of its locally accumulated gradient, which is determined by federation-level statistics such as the number of local samples or local classes. In DFR-ACGSV, however, this restriction is removed and each client fully retains its locally accumulated gradient. Although both DPR-CFFL and DFR-ACGSV implement this retention directly on the client side, the same operation can be executed equivalently on the server side by adding the corresponding client-specific accumulated gradient to the reward vector after the server has computed the client reward. Furthermore, the GMD values obtained with DFR-ACGSV are also substantially higher than those obtained by ACGSV and UR-ACGSV in the same settings. The reasoning presented in Section 5.7 to explain model differentiation also applies to DFR-ACGSV.

The first observation drawn from the Kendall's τ and GMD values reported in Tables 5.16 and 5.17 is that DFR-ACGSV generally achieves stronger contribution-rank preservation and model differentiation than ACGSV. More specifically, DFR-ACGSV produces consistently high Kendall's τ values across all settings except FA-MNIST under both POW and CLA partitioning, indicating that it preserves the reference contribution ranking more effectively than ACGSV. In addition, DFR-ACGSV achieves higher GMD values, demonstrating a greater degree of differentiation among the final client models.

Therefore, DFR-ACGSV experiments provide additional empirical evidence that retaining the locally accumulated gradient can induce client-level model differentiation, high contribution rank preservation, and high linear association. Furthermore, these results

[9]The Kendall's τ, GMD and MPAD values computed with standalone test accuracy for each run are with the DFR-ACGSV reported in Tables D.26, D.27, D.28, and D.29, respectively.

show that the effect of local gradient retention is not specific to CFFL or to its partial-retention rule; instead, it is a broader mechanism capable of causing the client models to drift away from each other.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (POW)	ACGSV	0.959	0.828	0.11	**4.02**
	DFR-ACGSV	**0.990**	**0.973**	**2.48**	1.61
FA-MNIST (POW)	ACGSV	**0.884**	**0.275**	0.09	**4.76**
	DFR-ACGSV	0.657	0.093	**1.51**	-8.50
CIFAR-10 (POW)	ACGSV	0.708	0.526	0.26	**41.14**
	DFR-ACGSV	**0.972**	**0.964**	**12.23**	29.93
SST-2 (POW)	ACGSV	0.737	0.380	0.08	**15.52**
	DFR-ACGSV	**0.996**	**0.982**	**5.82**	5.32

Table 5.16: Comparison between ACGSV and DFR-ACGSV under POW data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

Dataset	Framework	r_{SA}	τ_{SA}	GMD	MPAD
MNIST (CLA)	ACGSV	0.747	0.326	1.16	**29.52**
	DFR-ACGSV	**1.000**	**1.000**	**31.38**	0.24
FA-MNIST (CLA)	ACGSV	0.795	0.808	5.24	**33.65**
	DFR-ACGSV	**0.998**	**1.000**	**25.49**	-1.76
CIFAR-10 (CLA)	ACGSV	0.834	0.719	4.44	**20.07**
	DFR-ACGSV	**0.943**	**0.742**	**10.55**	2.20

Table 5.17: Comparison between ACGSV and DFR-ACGSV under CLA data partitioning using the mean Pearson's r_{SA}, Kendall's τ_{SA}, GMD, and MPAD across experimental runs. The subscript SA indicates that the correlation coefficients were computed between the standalone and federated client test accuracies. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

The second observation concerns the MPAD values reported in Tables 5.16 and 5.17. In ACGSV, the very low GMD values indicate that the client models generally converge to similar levels of predictive performance. By contrast, DFR-ACGSV introduces substantially stronger model differentiation, but this improvement is consistently accompanied by a reduction in MPAD.

Under POW partitioning, DFR-ACGSV increases GMD while reducing MPAD in every setting, and the same pattern is observed under CLA partitioning. These results provide strong evidence of a trade-off between contribution fairness and average accuracy gain: maintaining stronger differences among client performance levels improves model differentiation and, in most settings, reference contribution-rank preservation, but reduces the average benefit obtained from federated training. This behavior suggests that complete local-update retention preserves client-specific optimization trajectories and thereby achieves "contribution-aligned differentiation". However, these persistent client-specific updates may also move some client models away from parameter regions associated with better generalization, thereby reducing their final accuracy gains.

This effect is particularly evident in the FA-MNIST settings, where the negative MPAD values indicate that, on average, the federated client models perform worse than their corresponding standalone models. The convergence curves in Figures 5.30 and 5.31 show a late-round decline in client test accuracy, supporting the interpretation that continued training degraded generalization performance. As the final accuracy differences became small and varied across experimental seeds, numerous pairwise rank inversions emerged, which explains the lower Kendall's τ_{SA} obtained in these two settings.

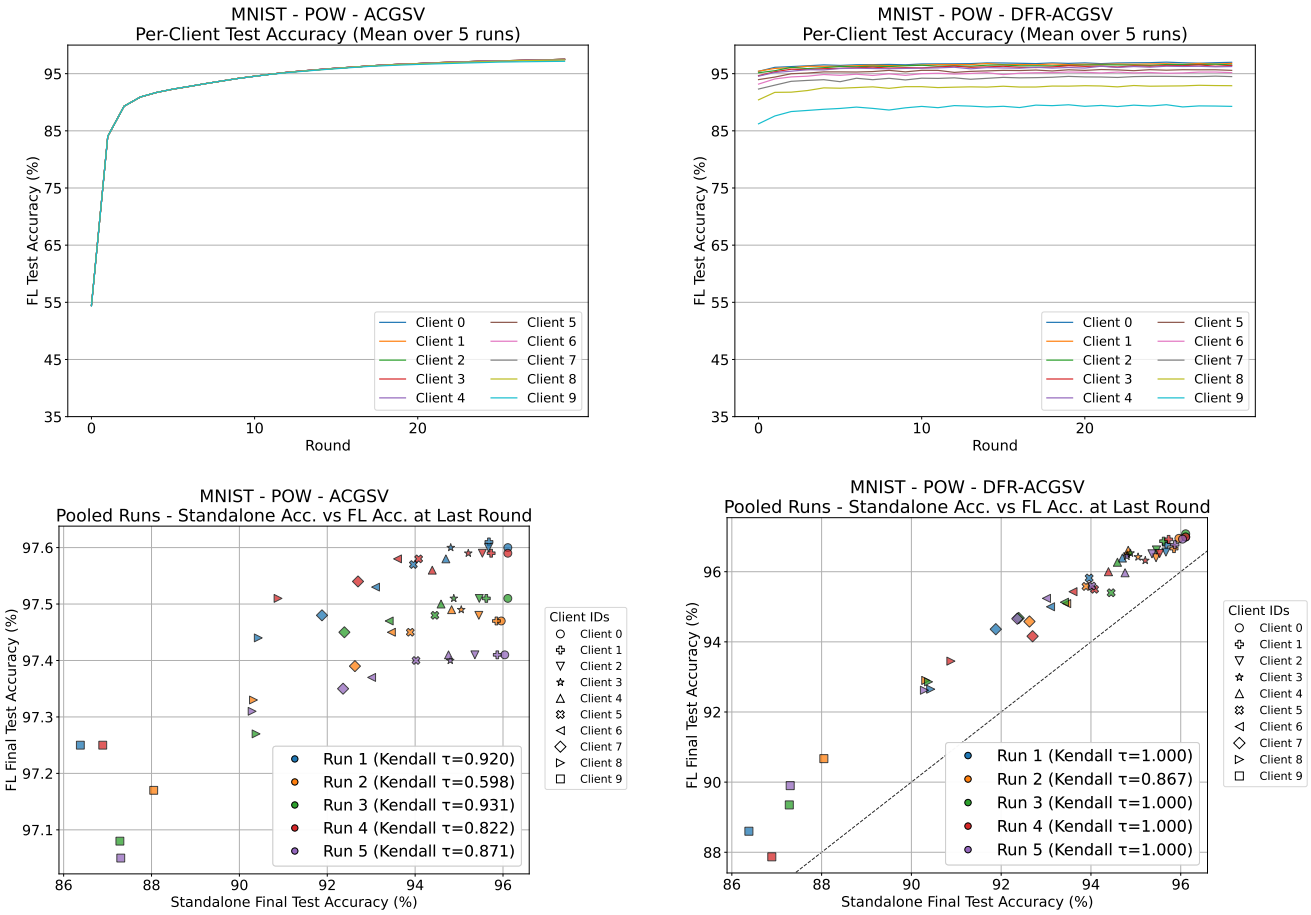


Figure 5.28: Per-client results for the ACGSV framework and the DFR-ACGSV variant on MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

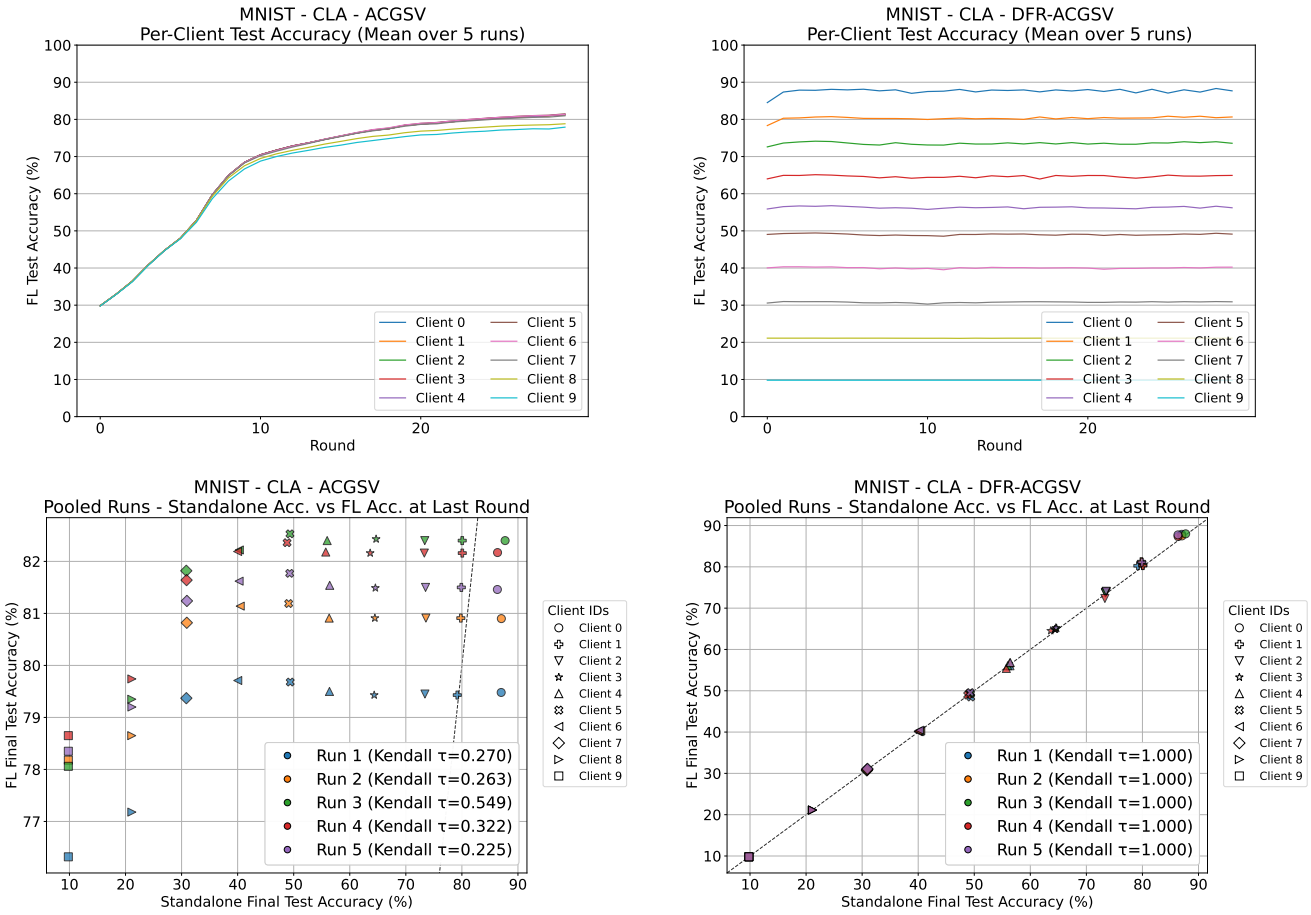


Figure 5.29: Per-client results for the ACGSV framework and the DFR-ACGSV variant on MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

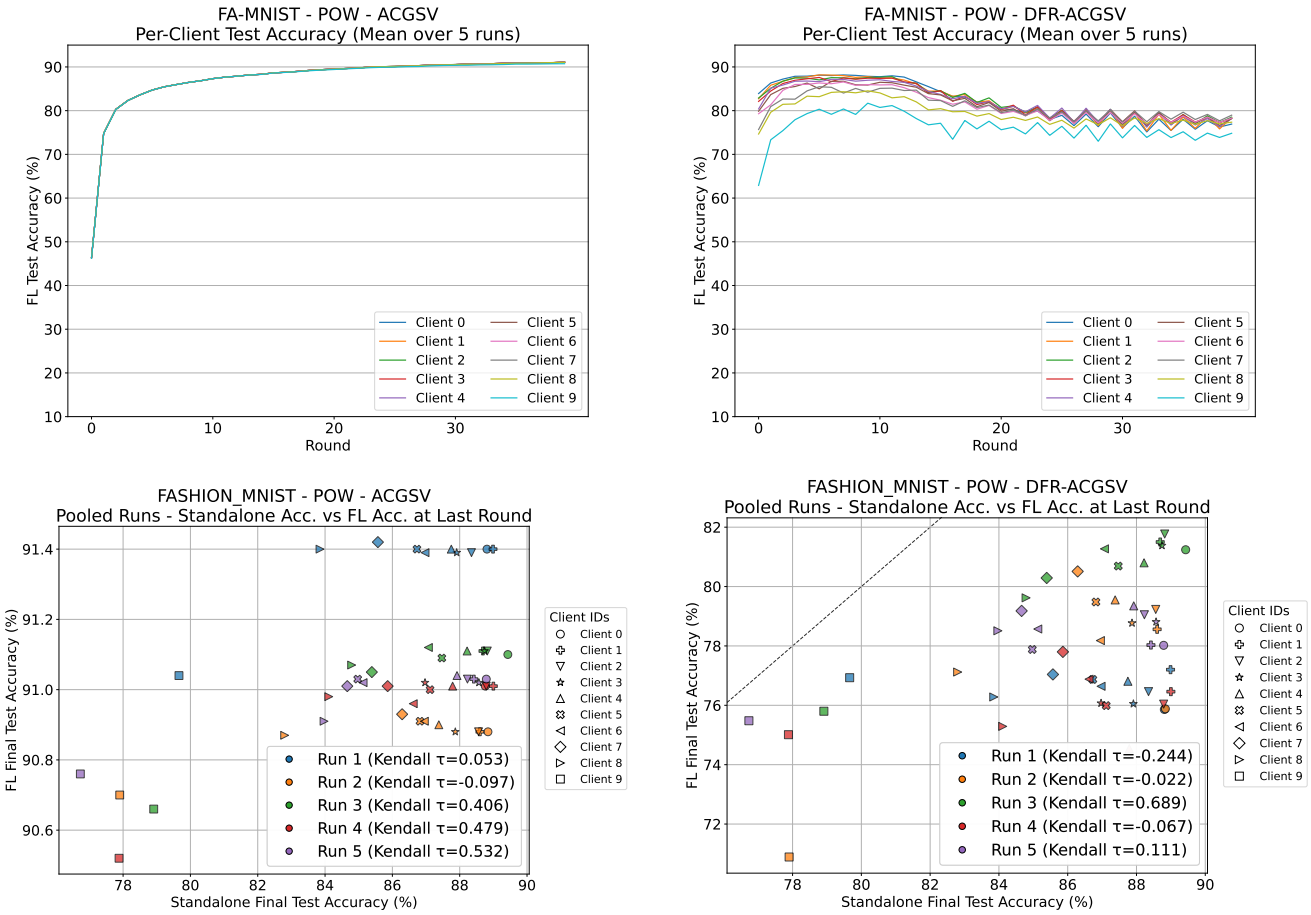


Figure 5.30: Per-client results for the ACGSV framework and the DFR-ACGSV variant on FA-MNIST under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

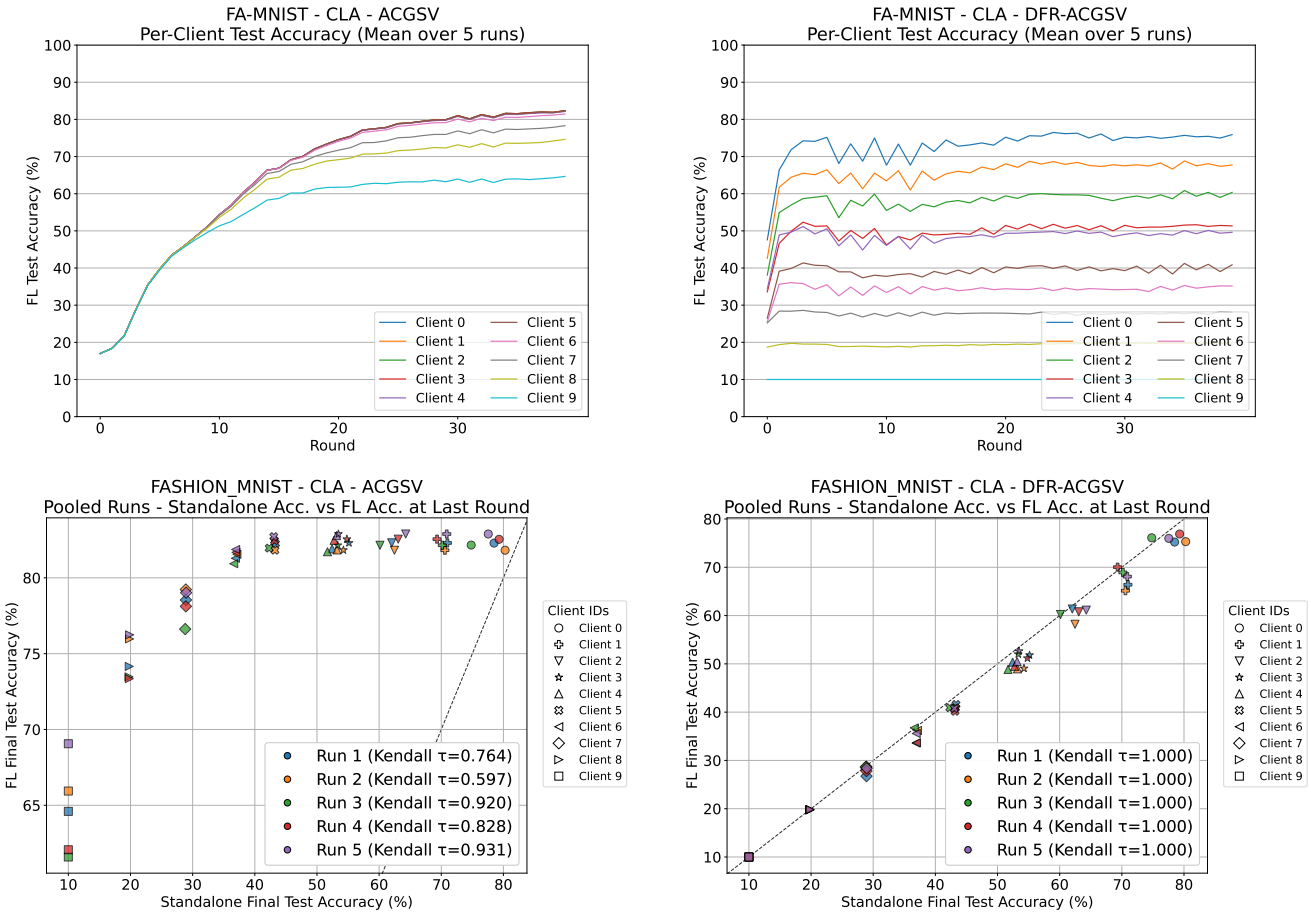


Figure 5.31: Per-client results for the ACGSV framework and the DFR-ACGSV variant on FA-MNIST under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

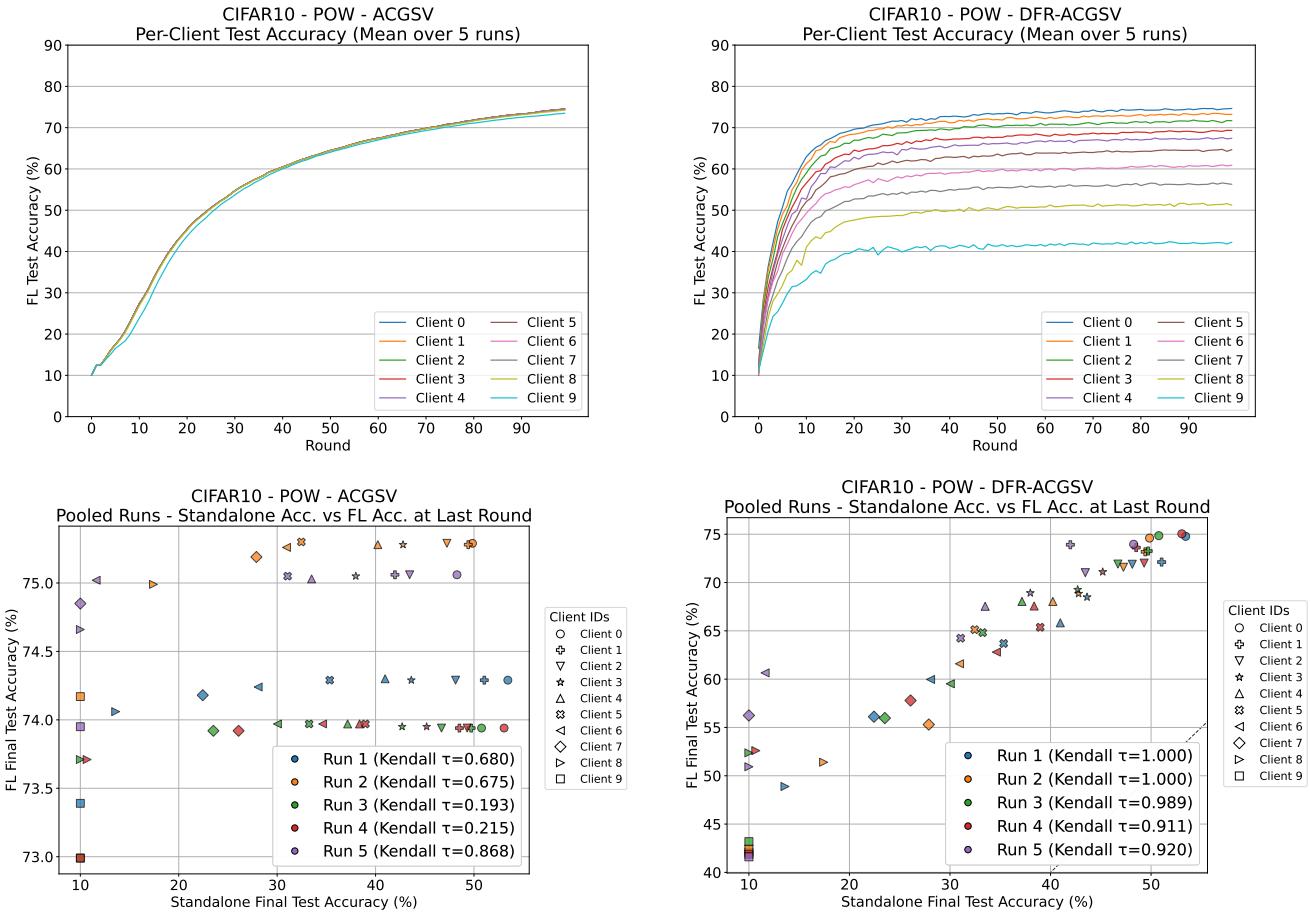


Figure 5.32: Per-client results for the ACGSV framework and the DFR-ACGSV variant on CIFAR10 under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

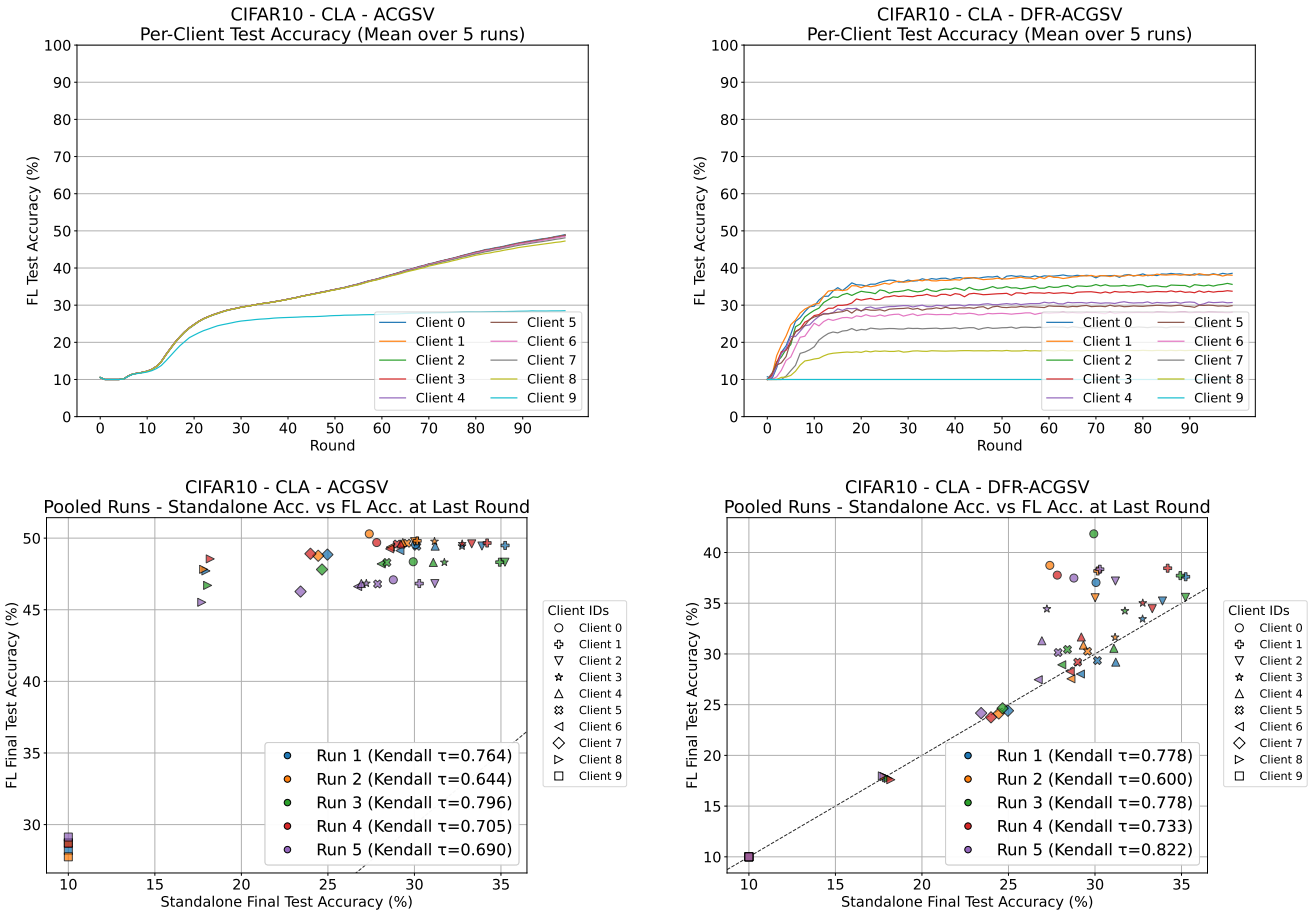


Figure 5.33: Per-client results for the ACGSV framework and the DFR-ACGSV variant on CIFAR10 under CLA data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

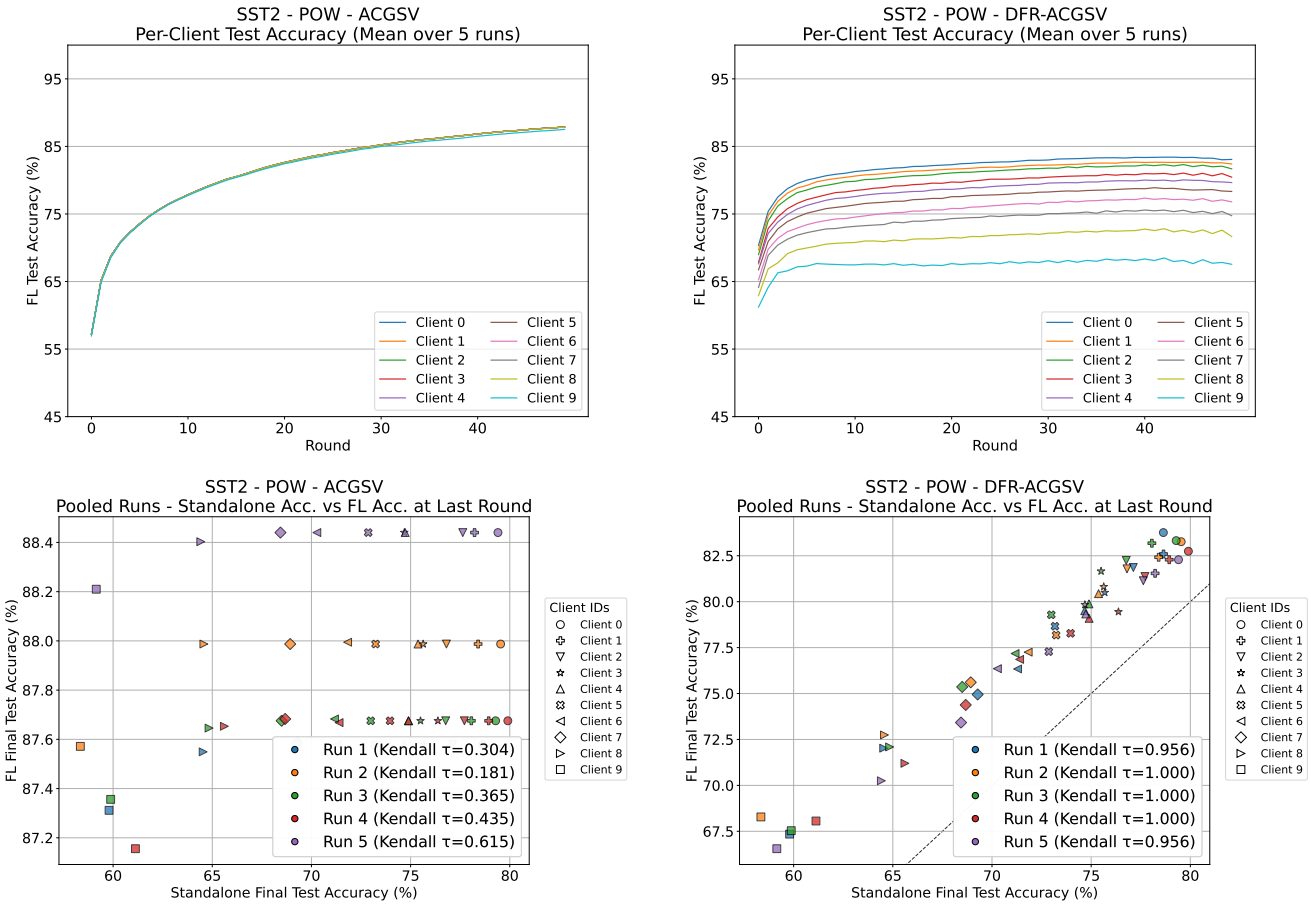


Figure 5.34: Per-client results for the ACGSV framework and the DFR-ACGSV variant on SST2 under POW data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

Consistent with the previous experiments, Tables 5.18 and 5.19 report the correlation values obtained using each client's Shapley Value, instead of standalone test accuracy.[10] The corresponding scatter plots are shown in Figures 5.35 and 5.36. Overall, replacing standalone accuracy with Shapley Values as the reference contribution signal does not fundamentally change the interpretation of DFR-ACGSV.

[10]The per-run Pearson's r and Kendall's τ are reported in Tables D.30 and D.31.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (POW)	ACGSV	0.949	0.829	0.11	**4.02**
	DFR-ACGSV	**0.984**	**0.947**	**2.48**	1.61
FA-MNIST (POW)	ACGSV	**0.777**	**0.333**	0.09	**4.76**
	DFR-ACGSV	0.484	0.084	**1.51**	-8.50
CIFAR-10 (POW)	ACGSV	0.834	0.539	0.26	**41.14**
	DFR-ACGSV	**0.995**	**1.000**	**12.23**	29.93
SST-2 (POW)	ACGSV	0.657	0.380	0.08	**15.52**
	DFR-ACGSV	**0.987**	**1.000**	**5.82**	5.32

Table 5.18: Comparison between ACGSV and DFR-ACGSV under POW data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, Gini Mean Difference (GMD), and Mean Paired Accuracy Difference (MPAD) across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

Dataset	Framework	r_{SV}	τ_{SV}	GMD	MPAD
MNIST (CLA)	ACGSV	0.537	0.326	1.16	**29.52**
	DFR-ACGSV	**0.934**	**1.000**	**31.38**	0.24
FA-MNIST (CLA)	ACGSV	0.570	0.808	5.24	**33.65**
	DFR-ACGSV	**0.932**	**1.000**	**25.49**	-1.76
CIFAR-10 (CLA)	ACGSV	0.565	0.900	4.44	**20.07**
	DFR-ACGSV	**0.919**	**0.920**	**10.55**	2.20

Table 5.19: Comparison between ACGSV and DFR-ACGSV under CLA data partitioning using the mean Pearson's r_{SV}, Kendall's τ_{SV}, Gini Mean Difference (GMD), and Mean Paired Accuracy Difference (MPAD) across experimental runs. The subscript SV indicates that the correlation coefficients were computed between each client's Shapley Value, using test accuracy as the coalition utility, and its final federated test accuracy. GMD and MPAD are reported in percentage points. Bold values indicate the highest value obtained for each dataset and metric.

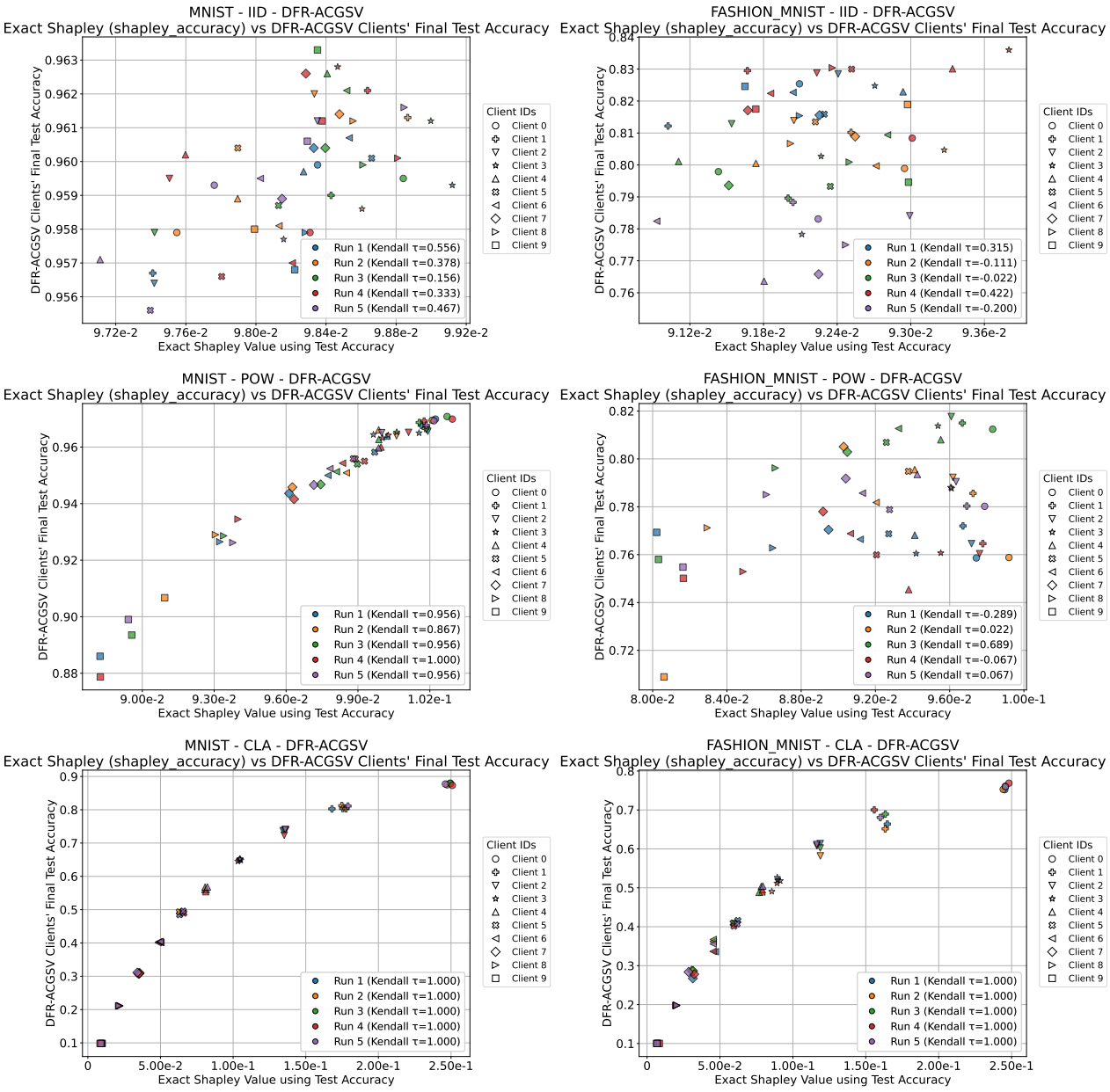


Figure 5.35: Shapley-value-based fairness analysis for DFR-ACGSV on MNIST and FA-MNIST under IID, POW, and CLA data partitioning. The left column reports the DFR-ACGSV results on the MNIST dataset, while the right column reports the results on the FA-MNIST dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

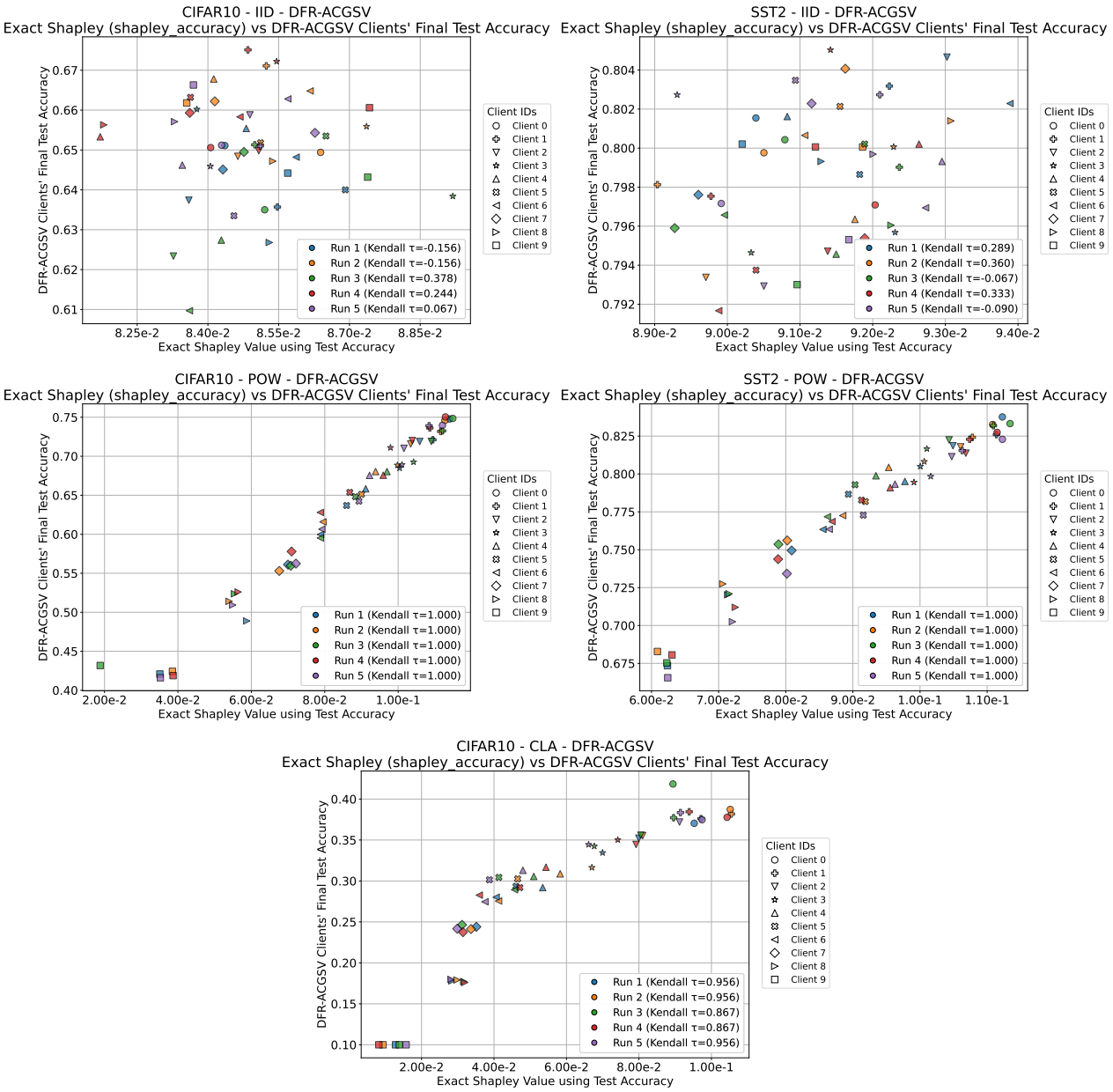


Figure 5.36: Shapley-value-based fairness analysis for DFR-ACGSV on CIFAR10 under IID, POW, and CLA data partitioning, and SST under IID and POW data partitioning. The left column reports the DFR-ACGSV results on the CIFAR10 dataset, while the right column reports the results on the SST2 dataset. Each row corresponds to one data partitioning strategy. In each scatter plot, the x-axis represents the client Shapley Value computed using test accuracy as the coalition utility, and the y-axis represents the final federated test accuracy achieved by the same client. Each marker represents one client in one experimental run, and marker shapes identify the client IDs.

5.9 Chapter Conclusion

The experiments presented in this chapter address two research gaps in the contribution- and incentive-aware federated learning literature: the utilization of Top-K sparsification as a contribution-aware reward mechanism and the effect of local gradient retention. Furthermore, the ablation study also contain results that strengthen the limitations of the literature's method to evaluate contribution fairness. Together, these ablation studies constitute the main contributions of my thesis and expose limitations in the way existing frameworks are commonly designed and evaluated [7, 14, 21, 22, 23, 25, 39, 40, 41, 42].

The first contribution concerns the role of Top-K sparsification. The experiments with ACGSV, UR-ACGSV, and OTOPK-CFFL show that Top-K sparsification of the aggregated gradient does not provide a linear mapping between reward sparsity and model convergence. Sections 5.5 and 5.6 showed that clients that consistently received rewards with sparsity approximately at 70% or lower converged to models with nearly the same test accuracy. The results do not advocate against the usage of Top-K sparsified rewards as a method to promote collaborative fairness, but it brings evidence that, in order to achieve contribution fairness, the reward-generation pipeline needs to map estimated contributions to sufficiently distinct sparsity levels. Otherwise, it either has little effect on model differentiation and rank preservation, or act primarily as a coarse performance-degradation mechanism rather than a fine-grained method for translating contribution differences into gradient rewards of distinct quality.

Although the experiments did not exhaustively evaluate sparsity levels beyond those computed by the frameworks and their variants, extending the rewards to even higher sparsity is not necessary to establish the central conclusion of this ablation study. These clients already exhibit a consistent pattern in which increasing reward sparsity above 70% is accompanied by greater performance degradation and more pronounced model differentiation. Therefore, the main finding is not that Top-K sparsification is incapable of generating differentiated rewards; rather, meaningful differentiation emerges only when the rewards become sufficiently sparse.

The second contribution concerns the effect of *local gradient retention*. In this thesis, local gradient retention refers to any setting in which a client's final personalized update preserves information from its own locally accumulated gradient. This can occur either because the client directly retains all or part of its own locally accumulated gradient, or because the server explicitly adds it into the reward that's returned to the client. In both cases, the final personalized model is not determined by the FLCE algorithm or the estimated-contribution-based reward. The local gradient retention contains a component derived from the client's own local optimization trajectory which is independent of the FL framework's estimated contribution. The experiments presented in Sections 5.7 and 5.8 show that locally accumulated gradients are not a secondary implementation detail, and,

despite not being part of the contribution estimation and reward generation pipeline, they are the main driver of high fairness outcome in CFFL and the framework variants tested in this ablation study. It is important to emphasize that, this thesis does not advocate its usage as a mechanism to promote collaborative fairness.

Although the objective of this chapter is methodological rather than accuracy-driven, it is still important to discuss the trade-off between contribution fairness and the accuracy gains obtained through federated training. The analyses in Sections 5.6, 5.7, and 5.8 show that stronger reference contribution rank preservation and model differentiation can be accompanied by lower average accuracy gains across the federation. This trade-off is particularly evident when clients fully retain their locally accumulated gradients, as shown in Section 5.8, because the preserved client-specific optimization directions promote differentiation but can also reduce the average learning and generalization benefit obtained from collaboration.

Conversely, when clients do not retain their locally accumulated gradients and their models are updated only through estimated-contribution-based rewards generated by the server, as in Section 5.6, a high average accuracy gain should not be interpreted uncritically. If the reward mechanism is unable to meaningfully differentiate most clients, clients with distinct reference contributions may converge to models with the same or nearly identical test performance, while only the few clients receiving the most restrictive rewards experience substantial performance degradation. In this case, the framework-level accuracy gain is largely sustained by insufficiently differentiated clients achieving performance comparable to that of the highest contributors, rather than an even distribution of accuracy gain across the federation. Therefore, the resulting MPAD may appear favorable even though the reward mechanism has failed to translate contribution differences into proportionally differentiated client outcomes.

Finally, although not the primary objective of the ablation study, the results obtained with ACGSV, UR-ACGSV, and OTOPK-CFFL show that measuring contribution fairness only through Pearson's r correlation coefficient provides an incomplete and potentially misleading measure of the contribution fairness, regardless of whether the contribution reference values are defined by the clients' standalone test accuracy or their Shapley Values. In multiple experiments, a high Pearson's r correlation happened when there was low contribution rank preservation and negligible model differentiation across the federation. These observations support the novel evaluation protocol proposed in Chapter 4, which consists of evaluating individually the contribution rank preservation and model differentiation through Kendall's τ and GMD, respectively.

Chapter 6

Conclusion and Future Directions

Federated Learning enables independent, and often competing, data owners to train a shared model without exposing their local datasets, but incentivizing and sustaining this collaboration over time requires reward mechanisms that satisfy contribution fairness: the principle that each client should be rewarded commensurately to its contribution to the training. A prominent line of research pursues this objective through non-monetary rewards, distributing to each client a contribution-based Top-K sparsified version of the aggregated gradient, such that higher-contributing clients receive denser, and therefore more useful, updates than lower-contributing ones. Therefore, by controlling the sparsity degree of the aggregated gradient, the server could control the convergence of each client model.

This thesis started from two observations regarding the design of FL frameworks that seek to achieve contribution fairness. First, Top-K sparsification was not conceived as a reward mechanism, but as a gradient-compression technique whose original purpose is to *preserve* convergence under communication constraints, which does not make it obvious how it can also produce the client-level model differentiation that fairness demands. Second, existing frameworks entangle several moving parts, the Federated Learning Contribution Estimation (FLCE) algorithm, the gradient chosen for reward generation, the sparsification rule, and whether locally accumulated gradient retention is enabled or not. Therefore, when high-contributing clients end up with better-performing models, it is not clear which component is actually responsible for that outcome. Furthermore, the fairness of these frameworks is evaluated through a single number: the Pearson's r correlation between each client's standalone and federated test accuracy, which was verified that it can be influenced by outliers in the data distribution.

To investigate these questions, I reproduced and investigated two representative state-of-the-art frameworks that have several differences in their designs: ACGSV and CFFL. I evaluated both across four benchmark datasets, MNIST, FA-MNIST, CIFAR-10, and SST-2, under Independent and Identicall Distributed (IID), quantity-skewed (POW), and class-label-skewed (CLA) data partitions, and then designed a series of controlled ablation studies to isolate the effect of three components: the gradient used for reward generation, Top-K sparsification, and the retention of locally accumulated gradients. Throughout the analysis, both clients' standalone test accuracy and Shapley values were used as reference contribution signals to ensure that the conclusions did not depend on a particular choice of contribution reference.

Throughout the analysis, both clients' standalone test accuracy and Shapley values were used as reference contribution signals to ensure that the conclusions did not depend on a particular choice of contribution reference.

6.1 Summary of Contributions and Findings

The thesis makes the following contributions, where each exposes a gap in the way contribution-fairness frameworks are commonly built and assessed.

The evaluation methodology for FL contribution fairness is incomplete, and a more accurate protocol is needed. The first and most transferable contribution is methodological. A positive Pearson's r between standalone and federated accuracy indicates that the two performance profiles vary together. It does not show whether the contribution ranks are preserved, nor whether the final client models are meaningfully differentiated. This limitation is expected because Pearson's r was designed to measure linear association, not rank-order preservation or the spread in magnitude performance across clients. Nevertheless, both aspects are central to contribution fairness in FL, since a framework should not only align rewards with contribution, but also produce client-level outcomes that preserve the contribution ordering and generate meaningful model differentiation. The experiments show that a high Pearson's r correlation can be produced by a small group of low-performing clients that are separated from a large cluster of clients that converge to nearly identical accuracy. Two frameworks can, therefore, report similar Pearson's r values yet behave very differently once their convergence and client-level performance distributions are examined. To address this, I proposed a hierarchical evaluation protocol built on two complementary measures. Kendall's τ serves as the primary criterion,

because it verifies whether the contribution-based ranks are preserved after federated training: values closer to 1 indicate strong rank preservation, thus higher fairness, values near 0 indicate weak agreement, and negative values indicate a reversal of the expected ordering, under which a framework should not be regarded as fair regardless of how much differentiation it produces. The Gini Mean Difference (GMD) then acts as an auxiliary measure that quantifies whether a preserved ordering is accompanied by meaningful performance spread across clients.

Top-K sparsification does not provide a linear mapping between reward sparsity and model performance. The second contribution concerns the reward channel itself. Based on the results of ACGSV, UR-ACGSV, and OTOPK-CFFL variants, it is evident that Top-K sparsification of the aggregated gradient produces meaningful model differentiation only under substantial reward sparsity. In the studied setting, clients that consistently received rewards with sparsity below 70% converged to models with nearly identical test accuracy, and differentiation emerged only once the reward scheme imposed large sparsity gaps across clients. The results do not advocate against the usage of Top-K sparsified rewards as a method to promote collaborative fairness, but it brings evidence that in order to achieve contribution fairness the reward-generation pipeline needs to map estimated contributions to sufficiently distinct sparsity levels. Otherwise, it either has little effect on model differentiation and rank preservation, or acts as a coarse performance-degradation mechanism rather than a fine-grained method for translating contribution differences into gradient rewards of distinct quality. This is, on reflection, entirely consistent with the gradient-compression literature from which Top-K originates, where biased compression schemes are known to preserve convergence even at extreme sparsity levels [27, 28, 29].

Local-gradient retention is an under-acknowledged confounder that drives fairness scores. The third contribution concerns a component that the literature typically treats as a neutral implementation detail: whether a client should retain its own locally accumulated gradient or not. This local gradient retention could happen either because the clients carry their own gradients across rounds or because the server adds a client-specific local-gradient term to its reward. The DPR-CFFL and DFR-ACGSV framework variants remove Top-K sparsified rewards entirely and distribute to every client the same dense server update, so that no client-specific differentiation can originate from the contribution-based reward. Even under these conditions, retaining each client's local gradient still causes clients to converge to models that achieve distinct test accuracies and yields high Pearson's r and Kendall's τ values. A substantial part of the model differentiation and rank

preservation present in CFFL can be attributed to the clients' local gradient retention. Therefore, the model differentiation and reference contribution rank preservation is not governed by the FL framework's contribution estimation algorithm and the contribution-score-to-reward mapping pipeline, instead it was generated by the clients' own local training optimization.

The findings persist under a coalition-based contribution reference. Because the federation sizes considered in this thesis keep Shapley Value computation computationally feasible, the contribution fairness analysis was repeated using each client's Shapley Value as the reference contribution signal, with test accuracy serving as the coalition utility. Unlike standalone performance, this coalition-based reference accounts for each client's marginal contribution across all possible coalitions and therefore provides a contribution measure that is more accurate and account for coalitions of participants. The Shapley-based evaluation led to the same substantive conclusions as the standalone-accuracy-based analysis: Top-K sparsification alone remained insufficient to produce consistently calibrated model differentiation and reference contribution rank preservation, whereas local gradient retention continued to explain most of the fairness behavior observed in CFFL. Thus, the main conclusions are robust to the choice of reference contribution signal rather than being specific to standalone test accuracy.

6.2 Implications

The practical implications of these findings are threefold. First, evaluation of contribution-fairness frameworks should not rely on Pearson's r as a single summary statistic. Reporting a rank-preservation measure such as Kendall's τ together with a differentiation measure such as the GMD gives a far more faithful interpretation than Pearson's r of whether a framework actually preserves the clients' reference contribution ranks after they join the federation, when space constraints limit the display of charts showing per-client convergence and client-level accuracy distributions.

Second, local-gradient retention should not be treated as an secondary FL framework design detail. Two frameworks that appear to differ only in their contribution estimator may in fact owe most of their differentiation to whether, and how, clients retain their own gradients. Therefore, without accounting for this, comparisons across frameworks risk attributing to the contribution mechanism an effect that is not governed by the FL framework's contribution estimation algorithm and the contribution-score-to-reward mapping pipeline.

Third, framework designers should not focus exclusively on improving FLCE algorithms; they should give equal attention to the reward-generation mechanism itself. The reward mechanism must be capable of translating estimated contribution into reward quality in a calibrated way, such that its effect on client model convergence reflects the client's measured contribution and preserves its reference contribution rank.

6.3 Limitations

Several limitations bound the scope of these conclusions and should be kept in mind when interpreting them. The analysis relies on a fixed and relatively small federation size, a deliberate choice that makes Shapley Value computation tractable but that also leaves open how the observed effects scale to federations with many more clients. The empirical guidelines that emerge from the study, in particular the approximate 70% sparsity level at which Top-K begins to induce differentiation and the one-percentage-point GMD floor below which differentiation is treated as negligible, are setting-dependent rather than universal constants, and are expected to shift with the federation size, model architecture, data heterogeneity, and training hyperparameters.

The evaluation is grounded primarily on standalone test accuracy as the contribution proxy, although the Shapley-value cross-check substantially strengthens the findings. The study also examines two frameworks and four datasets which, while representative of the validation-free and validation-based branches of the literature and spanning both image and text tasks, do not exhaust the space of contribution-fairness designs or application domains. Finally, the thesis focuses on non-monetary, gradient-based rewards and does not address monetary incentive schemes, adversarial or free-riding clients, or the privacy and security dimensions that a deployed incentive mechanism would also need to satisfy.

6.4 Future Directions

A first direction is to test whether the findings hold at scale, by combining the ablation methodology with tractable Shapley Value approximations, so that the confounding role of local-gradient retention and the sparsity behavior of Top-K can be verified in federations far larger than those studied here.

A second research direction follows directly from the second contribution of this thesis: since Top-K is not a calibrated mechanism to generate contribution based rewards, future

work should focus on characterizing whether there exist ranges of K for which differences in estimated contribution can be translated into gradual and monotonic differences in client outcomes, rather than producing negligible differentiation at moderate sparsity levels and abrupt performance degradation at high sparsity levels. This analysis could support the design of an adaptive mapping from estimated contribution to K value ranges that are potentially independent on the dataset, model architecture, and stage of training. If such a mapping cannot be made sufficiently stable and well calibrated, developing alternative gradient-modification mechanisms that provide a more reliable relationship between estimated contribution and reward quality remains an important open problem.

A third direction is to broaden the empirical scope, evaluating additional FLCE algorithms, datasets, modalities, and heterogeneity regimes, and stress-testing the frameworks against adversarial and free-riding behavior.

Finally, the proposed evaluation protocol itself invites further development: formalizing how the Kendall's τ and GMD guidelines should scale with federation size, model, and hyperparameters, and complementing them with additional differentiation and fairness measures, would move the community toward a standardized and more complete methodology for assessing federated learning contribution fairness.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, 2017.

[2] Qianpiao Ma, Yang Xu, Hongli Xu, Zhida Jiang, Liusheng Huang, and He Huang. FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing. *IEEE Journal on Selected Areas in Communications*, 39(12):3654–3672, 2021.

[3] Zheng Chai, Yujing Chen, Ali Anwar, Liang Zhao, Yue Cheng, and Huzefa Rangwala. FedAT: a high-performance and communication-efficient federated learning system with asynchronous tiers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–16, 2021.

[4] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. TiFL: A Tier-based Federated Learning System. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, pages 125–136, 2020.

[5] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. PAPAYA: Practical, Private, and Scalable Federated Learning. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 4, pages 814–832, 2022.

[6] Cheng-Wei Ching, Xin Chen, Taehwan Kim, Bo Ji, Qingyang Wang, Dilma Da Silva, and Liting Hu. Totoro: A Scalable Federated Learning Engine for the Edge. In

Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys), pages 182–199, 2024.

[7] Zhaoxuan Wu, Mohammad Mohammadi Amiri, Ramesh Raskar, and Bryan Kian Hsiang Low. Incentive-aware federated learning with training-time model rewards. In *International Conference on Learning Representations (ICLR)*, 2024.

[8] Minfeng Qi, Ziyuan Wang, Shiping Chen, and Yang Xiang. A Hybrid Incentive Mechanism for Decentralized Federated Learning. *Distributed Ledger Technologies: Research and Practice*, 1(1):1–15, 2022.

[9] Hao Sun, Xiaoli Tang, Chengyi Yang, Zhenpeng Yu, Xiuli Wang, Qijie Ding, Zengxiang Li, and Han Yu. HiFi-Gas: Hierarchical Federated Learning Incentive Mechanism Enhanced Gas Usage Estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22824–22832, 2024.

[10] Yuxin Shi, Han Yu, and Cyril Leung. Towards Fairness-Aware Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):11922–11938, 2024.

[11] 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.

[12] Anton Wahrstätter, Sajjad Khan, and Davor Svetinovic. OpenFL: A scalable and secure decentralized federated learning system on the Ethereum blockchain. *Internet of Things*, 26:101174, 2024.

[13] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A Fairness-aware Incentive Scheme for Federated Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 393–399, 2020.

[14] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 16104–16117, 2021.

[15] Hao Wu, Likun Zhang, Shucheng Li, Fengyuan Xu, and Sheng Zhong. CoAst: Validation-Free Contribution Assessment for Federated Learning based on Cross-Round Valuation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1839–1847, 2024.

[16] Jingjing Guo, Jiaxing Li, Zhiquan Liu, Yupeng Xiong, Yong Ma, Athanasios V. Vasilakos, Xinghua Li, and Jianfeng Ma. LCEFL: A Lightweight Contribution Evaluation Approach for Federated Learning. *IEEE Transactions on Mobile Computing*, 24(7):6643–6657, 2025.

[17] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair Resource Allocation in Federated Learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[18] Tianshu Song, Yongxin Tong, and Shuyue Wei. Profit Allocation for Federated Learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2577–2586, 2019.

[19] Yong Wang, Kaiyu Li, Yuyu Luo, Guoliang Li, Yunyan Guo, and Zhuo Wang. Fast, Robust and Interpretable Participant Contribution Estimation for Federated Learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2298–2311, 2024.

[20] Hongyi Wu, Xiaoying Tang, Ying-Jun Angela Zhang, and Lin Gao. Incentive Mechanism for Federated Learning based on Random Client Sampling. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1640–1645, 2022.

[21] Chenglu Pan, Jiarong Xu, Yue Yu, Ziqi Yang, Qingbiao Wu, Chunping Wang, Lei Chen, and Yang Yang. Towards Fair Graph Federated Learning via Incentive Mechanisms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14499–14507, 2024.

[22] 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.

[23] Lingjuan Lyu, Xinyi Xu, Qian Wang, and Han Yu. Collaborative Fairness in Federated Learning. In Qiang Yang, Lixin Fan, and Han Yu, editors, *Federated Learning: Privacy and Incentive*, pages 189–204. Springer International Publishing, 2020.

[24] Zhuan Shi, Lan Zhang, Zhenyu Yao, Lingjuan Lyu, Cen Chen, Li Wang, Junhao Wang, and Xiang-Yang Li. FedFAIM: A Model Performance-Based Fair Incentive Mechanism for Federated Learning. *IEEE Transactions on Big Data*, 10(6):1038–1050, 2024.

[25] Bizhi Lei, Fan Chen, YuXin Xie, Shaobo Zhang, Entao Luo, and Qing Yang. FedDAR: Multidimensional Contribution Assessment and Adaptive Gradient Rewards for Collaborative Fairness in Heterogeneous Federated Learning. In *International Conference on Intelligent Computing (ICIC)*, pages 100–113, 2025.

[26] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative Machine Learning with Incentive-Aware Model Rewards. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 8927–8936, 2020.

[27] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 5977–5987, 2018.

[28] Guangyao Li. Layer-wise Top-k Gradient Sparsification for Distributed Deep Learning. In *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pages 112–115, 2023.

[29] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 4452–4463, 2018.

[30] Tianrun Yu, Jiaqi Wang, Haoyu Wang, Mingquan Lin, Han Liu, Nelson S Yee, and Fenglong Ma. Towards Collaborative Fairness in Federated Learning Under Imbalanced Covariate Shift. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Volume 2*, pages 3645–3656, 2025.

[31] Do-Van Nguyen, Anh Khoa Tran, and Koji Zettsu. FedProb: An Aggregation Method Based on Feature Probability Distribution for Federated Learning on Non-IID Data. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2875–2881, 2022.

[32] Paul Zheng, Yao Zhu, Zhengming Zhang, Yulin Hu, and Anke Schmeink. Federated Learning in Heterogeneous Networks with Unreliable Communication. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2021.

[33] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In *Proceedings of the 39th International*

Conference on Machine Learning (ICML), volume 162 of *Proceedings of Machine Learning Research*, pages 11814–11827, 2022.

[34] Paritosh Ramanan and Kiyoshi Nakayama. Baffle: Blockchain based aggregator free federated learning. In *2020 IEEE international conference on blockchain (Blockchain)*, pages 72–81, 2020.

[35] Shashi Raj Pandey, Lam Duc Nguyen, and Petar Popovski. Fedtoken: Tokenized incentives for data contribution in federated learning. In *NeurIPS 2022 Workshop on Federated Learning: Recent Advances and New Challenges (FL-NeurIPS)*, 2022.

[36] Meriem Arbaoui, Mohamed-el-Amine Brahmia, Abdellatif Rahmoun, and Mourad Zghal. Federated Learning Survey: A Multi-Level Taxonomy of Aggregation Techniques, Experimental Insights, and Future Frontiers. *ACM Transactions on Intelligent Systems and Technology*, 15(6):1–69, 2024.

[37] 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, 2022.

[38] Yiwei Chen, Kaiyu Li, Guoliang Li, and Yong Wang. Contributions Estimation in Federated Learning: A Comprehensive Experimental Evaluation. *Proceedings of the VLDB Endowment*, 17(8):2077–2090, 2024.

[39] Jingwen Zhang, Yuezhou Wu, and Rong Pan. Incentive Mechanism for Horizontal Federated Learning Based on Reputation and Reverse Auction. In *Proceedings of the Web Conference 2021*, pages 947–956, 2021.

[40] Haitian Chen, Xuebin Chen, Lulu Peng, and Yuntian Bai. Personalized Fair Split Learning for Resource-Constrained Internet of Things. *Sensors*, 24(1):88, 2024.

[41] Hui Huang, Di Xiao, Xue Tan, and Zhuyang Yu. Privacy-Preserving and Hierarchically Fair Federated Learning in IoT Based on Compressed Sensing. *IEEE Transactions on Network Science and Engineering*, 13:7527–7546, 2026.

[42] Hailin Yang, Yanhong Huang, Jianqi Shi, and Yang Yang. A Federated Framework for Edge Computing Devices with Collaborative Fairness and Adversarial Robustness. *Journal of Grid Computing*, 21(3):36, 2023.

[43] Xiaoyun Li and Ping Li. Analysis of error feedback in federated non-convex optimization with biased compression: Fast convergence and partial participation. In

Proceedings of the 40th International Conference on Machine Learning (ICML), volume 202 of *Proceedings of Machine Learning Research*, pages 19638–19688, 2023.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, 2012.

[45] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251, 2019.

[48] Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. GTG-Shapley: Efficient and Accurate Participant Contribution Evaluation in Federated Learning. *ACM Transactions on Intelligent Systems and Technology*, 13(4):1–21, 2022.

[49] Qiheng Sun, Xiang Li, Jiayao Zhang, Li Xiong, Weiran Liu, Jinfei Liu, Zhan Qin, and Kui Ren. ShapleyFL: Robust Federated Learning Based on Shapley Value. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2096–2108, 2023.

[50] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8239–8247, 2021.

[51] Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.

[52] Emily Black and Matt Fredrikson. Leave-one-out Unfairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 285–295, 2021.

[53] Lloyd S. Shapley. A Value for n-Person Games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, 1953.

[54] 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.

[55] Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Validation Free and Replication Robust Volume-based Data Valuation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 10837–10848, 2021.

[56] Yunmi Kim, Tae-Hwan Kim, and Tolga Ergün. The instability of the Pearson correlation coefficient in the presence of coincidental outliers. *Finance Research Letters*, 13:243–257, 2015.

[57] Cyril R Pernet, Rand Wilcox, and Guillaume A Rousselet. Robust correlation analyses: false positive and power validation using a new open source matlab toolbox. *Frontiers in Psychology*, 3:606, 2013.

[58] Edoardo Saccenti, Margriet HWB Hendriks, and Age K Smilde. Corruption of the Pearson correlation coefficient by measurement error and its estimation, bias, and correction under different error models. *Scientific Reports*, 10(1):438, 2020.

[59] Charles Spearman. The Proof and Measurement of Association Between Two Things. *The American Journal of Psychology*, 15(1):72–101, 1904.

[60] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[61] Maurice G Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251, 1945.

[62] Lidia Ceriani and Paolo Verme. The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini. *The Journal of Economic Inequality*, 10(3):421–443, 2012.

[63] Xinyi Xu and Lingjuan Lyu. Towards building a robust and fair federated learning system. *arXiv preprint arXiv:2011.10464*, 2020.

[64] N. Silver and William Dunlap. Averaging Correlation Coefficients: Should Fisher's z Transformation Be Used? *Journal of Applied Psychology*, 72:146–148, 1987.

[65] David M Corey, William P Dunlap, and Michael J Burke. Averaging correlations: Expected values and bias in combined Pearson rs and Fisher's z transformations. *The Journal of General Psychology*, 125(3):245–261, 1998.

[66] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. MNIST Handwritten Digit Database, 2010. Published: Online dataset.

[67] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[68] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report, University of Toronto, 2009.

[69] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, 2013.

[70] Lisandro Dalcin and Yao-Lung L. Fang. mpi4py: Status Update After 12 Years of Development. *Computing in Science & Engineering*, 23(4):47–54, 2021.

[71] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS)*, pages 929–947, 2024.

[72] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*, 2020.

[73] Shiyun Lin, Yuze Han, Xiang Li, and Zhihua Zhang. Personalized Federated Learning towards Communication Efficiency, Robustness and Fairness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 30471–30485, 2022.

[74] Marcel Wever, Maximilian Muschalik, Fabian Fumagalli, and Marius Lindauer. HyperSHAP: Shapley Values and Interactions for Explaining Hyperparameter Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 26867–26875, 2026.

[75] Hung-Mo Lin and Vernon M Chinchilli. A measure of association between two sets of multivariate responses. *Journal of Biopharmaceutical Statistics*, 11(3):139–154, 2001.

[76] Lihi Dery and Erez Shmueli. BoostLR: a boosting-based learning ensemble for label ranking tasks. *IEEE Access*, 8:176023–176032, 2020.

[77] Anton Y Kulaga, Eugen Ursu, Dmitri Toren, Vladyslava Tyshchenko, Rodrigo Guinea, Malvina Pushkova, Vadim E Fraifeld, and Robi Tacutu. Machine learning analysis of longevity-associated gene expression landscapes in mammals. *International Journal of Molecular Sciences*, 22(3):1073, 2021.

[78] A. I. McLeod. Kendall: Kendall Rank Correlation and Mann-Kendall Trend Test, 2005. Published: R package.

[79] Herold Dehling, Daniel Vogel, Martin Wendler, and Dominik Wied. Testing for changes in Kendall's tau. *Econometric Theory*, 33(6):1352–1386, 2017.

[80] Jeremy MG Taylor. Kendall's and Spearman's correlation coefficients in the presence of a blocking variable. *Biometrics*, 43(2):409–416, 1987.

[81] Xue Tan, Di Xiao, Hui Huang, Mengdi Wang, and Min Li. Hierarchically Fair and Differentially Private Federated Learning in Industrial IoT Based on Compressed Sensing With Adaptive-Thresholding Sparsification. *IEEE Transactions on Industrial Informatics*, 20(12):13731–13740, 2024.

[82] Holger R. Roth, Yan Cheng, Yuhong Wen, Isaac Yang, Ziyue Xu, Yuan-Ting Hsieh, Kristopher Kersten, Ahmed Harouni, Can Zhao, Kevin Lu, Zhihong Zhang, Wenqi Li, Andriy Myronenko, Dong Yang, Sean Yang, Nicola Rieke, Abood Quraini, Chester Chen, Daguang Xu, Nic Ma, Prerna Dogra, Mona Flores, and Andrew Feng. NVIDIA FLARE: Federated Learning from Simulation to Real-World. *IEEE Data Engineering Bulletin*, 46(1):170–184, 2023.

[83] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A Principled Approach to Data Valuation for Federated Learning. In Qiang Yang, Lixin Fan, and Han Yu, editors, *Federated Learning: Privacy and Incentive*, pages 153–167. Springer International Publishing, 2020.

[84] Oscar Dilley, Juan Marcelo Parra-Ullauri, Rasheed Hussain, and Dimitra Simeonidou. Federated fairness analytics: Quantifying fairness in federated learning. *arXiv preprint arXiv:2408.08214*, 2024.

[85] Pengchao Han, Shiqiang Wang, and Kin K. Leung. Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 300–310, 2020.

[86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[87] Mavuto M Mukaka. A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal*, 24(3):69, 2012.

[88] Lisandro Dalcin and Yao-Lung L. Fang. mpi4py: Status Update After 12 Years of Development. *Computing in Science & Engineering*, 23(4):47–54, 2021.

[89] Flower Labs. Flower: A Friendly Federated AI Framework, 2026. Published: GitHub repository.

[90] Eyal Winter. The Shapley Value. In Robert J. Aumann and Sergiu Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3 of *Handbook of Game Theory with Economic Applications*, pages 2025–2054. Elsevier, 2002.

[91] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated Learning on Non-IID Data Silos: An Experimental Study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978, 2022.

[92] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[93] Junhao Wang, Lan Zhang, Anran Li, Xuanke You, and Haoran Cheng. Efficient Participant Contribution Evaluation for Horizontal and Vertical Federated Learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 911–923, 2022.

[94] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards Efficient Data Valuation Based on the Shapley Value. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176, 2019.

[95] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2020.

[96] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Client-Edge-Cloud Hierarchical Federated Learning. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[97] Tom Yan and Ariel D Procaccia. If you like shapley then you'll love the core. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5751–5759, 2021.

[98] Tamás Solymosi and T. E. S. Raghavan. Assignment games with stable core. *International Journal of Game Theory*, 30(2):177–185, 2001.

[99] Maarten van Steen and Andrew S. Tanenbaum. *Distributed Systems*. distributed-systems.net, 4th edition, 2023.

[100] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 14668–14679, 2019.

[101] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-Efficient Federated Learning with Sketching. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 8253–8265, 2020.

[102] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 14236–14245, 2019.

[103] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient Federated Learning via Guided Participant Selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 19–35, 2021.

[104] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, and others. IBM Federated Learning: an Enterprise Framework White Paper V0. 1. *arXiv preprint arXiv:2007.10987*, 2020.

[105] Patrick Foley, Micah J Sheller, Brandon Edwards, Sarthak Pati, Walter Riviera, Mansi Sharma, Prakash Narayana Moorthy, Shi-han Wang, Jason Martin, Parsa Mirhaji, Prashant Shah, and Spyridon Bakas. OpenFL: the open federated learning library. *Physics in Medicine & Biology*, 67(21):214001, 2022.

[106] OpenMined. PySyft: Privacy-Preserving Data Science Framework, 2025. Published: GitHub repository.

[107] Caner Korkmaz, Halil Eralp Kocas, Ahmet Uysal, Ahmed Masry, Oznur Ozkasap, and Baris Akgun. Chain fl: Decentralized federated machine learning via blockchain. In *2020 Second international conference on blockchain computing and applications (BCCA)*, pages 140–146, 2020.

[108] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 1, pages 374–388, 2019.

[109] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated Learning with Buffered Asynchronous Aggregation. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pages 3581–3607, 2022.

[110] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2020.

[111] Jian Xu, Yi Yan, and Shao-Lun Huang. FedPer++: Toward improved personalized federated learning on heterogeneous and imbalanced data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[112] Andrew Hard, Chloé M. Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. Federated Learning for Mobile Keyboard Prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[113] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William Dally. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations (ICLR)*, 2018.

[114] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

[115] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. AdaComp: adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 2827–2835, 2018.

[116] Alexandros Karargyris, Renato Umeton, Micah J. Sheller, Alejandro Aristizabal, Johnu George, Anna Wuest, Sarthak Pati, Hasan Kassem, Maximilian Zenk, Ujjwal Baid, Prakash Narayana Moorthy, Alexander Chowdhury, Junyi Guo, Sahil Nalawade, Jacob Rosenthal, David Kanter, Maria Xenochristou, Daniel J. Beutel, Verena Chung, Timothy Bergquist, James Eddy, Abubakar Abid, Lewis Tunstall, Omar Sanseviero, Dimitrios Dimitriadis, Yiming Qian, Xinxing Xu, Yong Liu, Rick Siow Mong Goh, Srini Bala, Victor Bittorf, Sreekar Reddy Puchala, Biagio Ricciuti, Soujanya Samineni, Eshna Sengupta, Akshay Chaudhari, Cody Coleman, Bala Desinghu, Gregory Diamos, Debo Dutta, Diane Feddema, Grigori Fursin, Xinyuan Huang, Satyananda Kashyap, Nicholas Lane, Indranil Mallick, Pietro Mascagni, Virendra Mehta, Cassiano Ferro Moraes, Vivek Natarajan, Nikola Nikolov, Nicolas Padoy, Gennady Pekhimenko, Vijay Janapa Reddi, G. Anthony Reina, Pablo Ribalta, Abhishek Singh, Jayaraman J. Thiagarajan, Jacob Albrecht, Thomas Wolf, Geralyn Miller, Huazhu Fu, Prashant Shah, Daguang Xu, Poonam Yadav, David Talby, Mark M. Awad, Jeremy P. Howard, Michael Rosenthal, Luigi Marchionni,

Massimo Loda, Jason M. Johnson, Spyridon Bakas, Peter Mattson, FeTS Consortium, BraTS-2020 Consortium, and AI4SafeChole Consortium. Federated benchmarking of medical artificial intelligence with MedPerf. *Nature Machine Intelligence*, 5(7):799–810, 2023.

[117] Wouter Heyndrickx, Lewis Mervin, Tobias Morawietz, Noé Sturm, Lukas Friedrich, Adam Zalewski, Anastasia Pentina, Lina Humbeck, Martijn Oldenhof, Ritsuya Niwayama, Peter Schmidtke, Nikolas Fechner, Jaak Simm, Adam Arany, Nicolas Drizard, Rama Jabal, Arina Afanasyeva, Regis Loeb, Shlok Verma, Simon Harnqvist, Matthew Holmes, Balazs Pejo, Maria Telenczuk, Nicholas Holway, Arne Dieckmann, Nicola Rieke, Friederike Zumsande, Djork-Arné Clevert, Michael Krug, Christopher Luscombe, Darren Green, Peter Ertl, Peter Antal, David Marcus, Nicolas Do Huu, Hideyoshi Fuji, Stephen Pickett, Gergely Acs, Eric Boniface, Bernd Beck, Yax Sun, Arnaud Gohier, Friedrich Rippmann, Ola Engkvist, Andreas H. Göller, Yves Moreau, Mathieu N. Galtier, Ansgar Schuffenhauer, and Hugo Ceulemans. MELLODDY: Cross-pharma Federated Learning at Unprecedented Scale Unlocks Benefits in QSAR without Compromising Proprietary Information. *Journal of Chemical Information and Modeling*, 64(7):2331–2344, 2024.

[118] Abdelrhman Gaber, Hassan Abdeltwab, and Tamer Elbatt. FedCVD: Towards a Scalable, Privacy-Preserving Federated Learning Model for Cardiovascular Diseases Prediction. In *Proceedings of the 2024 8th International Conference on Machine Learning and Soft Computing*, pages 7–11, 2024.

APPENDICES

Appendix A

Notation Tables

Notation	Definition
$\mathcal{N}$	The grand coalition of clients in the federation
$\mathcal{S}$	A coalition of clients in the federation. A subset of $\mathcal{N}$
$\mathcal{D}_i$	The dataset of the i-th client
$\lvert\mathcal{D}_i\rvert$	The dataset size of the i-th client
$\lvert\mathcal{D}_\mathcal{S}\rvert$	The dataset size of the coalition $\mathcal{S}$
$\lvert\mathcal{D}_\mathcal{N}\rvert$	The dataset size of the grand coalition
$\lvert\mathcal{C}_i\rvert$	The amount of classes available in the dataset of the i-th client
$\lvert\mathcal{C}_\mathcal{S}\rvert$	The amount of classes available in the dataset of the coalition $\mathcal{S}$
$\lvert\mathcal{C}_\mathcal{N}\rvert$	The amount of classes available in the dataset of the grand coalition
$\vec{w}_{0,0}$	The initial global model
$\vec{w}_{\mathcal{N},t}$	The global model at the t-th round
$\vec{w}_{i,t}$	The model of the i-th client at the t-th round
$\Delta\vec{w}_{i,t}$	The model parameter update of the i-th client at the t-th round
$\Delta\vec{w}_{\mathcal{N},t}$	The aggregated model parameter update at the t-th round
$\vec{u}_{i,t}$	The L2-normalized $\Delta\vec{w}_{i,t}$
$\vec{u}_{\mathcal{N},t}$	The vector representing the aggregation of $\vec{u}_{i,t}$
r	Pearson Correlation Coefficient
ρ	Spearman's Correlation Coefficient
τ	Kendall's Correlation Coefficient

Table A.1: Notations and definitions

Appendix B

Shapley Value Properties

The properties of the Shapley value presented here follow from the original axiomatic formulation proposed by Shapley [53]. For a cooperative game with player set $\mathcal{N}$ and characteristic function $v : 2^{\mathcal{N}} \to \mathbb{R}$, the Shapley value of player i is defined as

$$\phi_i(v) = \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}} \frac{|\mathcal{S}|!(|\mathcal{N}| - |\mathcal{S}| - 1)!}{|\mathcal{N}|!} \Big(v(\mathcal{S} \cup \{i\}) - v(\mathcal{S}) \Big).$$

This expression assigns to each player its average marginal contribution over all possible coalition formation orders. In the original formulation of Shapley, the value is uniquely characterized by three axioms: symmetry, efficiency, and additivity. Additional properties, such as the dummy-player condition, follow as corollaries.

1. **Symmetry.** If two players i and j contribute equally to every coalition that excludes them, then they must receive the same value. Formally, if

   $$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S} \cup \{j\}), \qquad \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i, j\},$$

   then

   $$\phi_i(v) = \phi_j(v).$$

   This property ensures that players who are indistinguishable in terms of their contribution to the game are treated identically.

2. **Efficiency.** The total value generated by the grand coalition is fully distributed among the players:
$$\sum_{i \in \mathcal{N}} \phi_i(v) = v(\mathcal{N}).$$
Thus, the Shapley value allocates the entire cooperative surplus without loss or excess.

3. **Additivity.** If two games with characteristic functions v and w are combined into a new game $(v + w)$, then the value assigned to each player in the combined game is the sum of the values assigned in the individual games:
$$\phi_i(v + w) = \phi_i(v) + \phi_i(w).$$
This property is also commonly referred to as linearity.

4. **Dummy Player Property.** If player i contributes only its standalone worth to every coalition, then its Shapley value equals that standalone worth. Formally, if
$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S}) + v(\{i\}), \qquad \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{i\},$$
then
$$\phi_i(v) = v(\{i\}).$$
As a special case, if $v(\{i\}) = 0$ and player i does not change the value of any coalition, then $\phi_i(v) = 0$. This special case is often referred to as the *null-player property*.

Appendix C

ACGSV Longer Training

Since the ACGSV presented a lower convergence than CFFL and other framework variants tested in this thesis, I also allowed the training to run for more rounds to verify whether the results would be different. However, there was no change in the outcome of the fairness metrics and model differentiation. Nevertheless, I put the experimental results here for completeness of the study.

Dataset	η	γ	B	E	T	α	β	Γ
MNIST	0.015	0.977	32	5	100	0.95	1.0	0.5
FA-MNIST	0.005	0.977	64	5	150	0.95	1.0	0.5
CIFAR10	0.02	0.977	64	5	200	0.95	1.0	0.5
SST2	$1e^{-4}$	0.977	256	5	150	0.95	1.0	1.0

Table C.1: Training hyperparameters used in the simulations with the ACGSV frameworks. These values were applied to the IID, POW, and CLA partitioning scenarios. The parameters α, β, and Γ are exclusive to ACGSV, while θ is exclusive to CFFL.

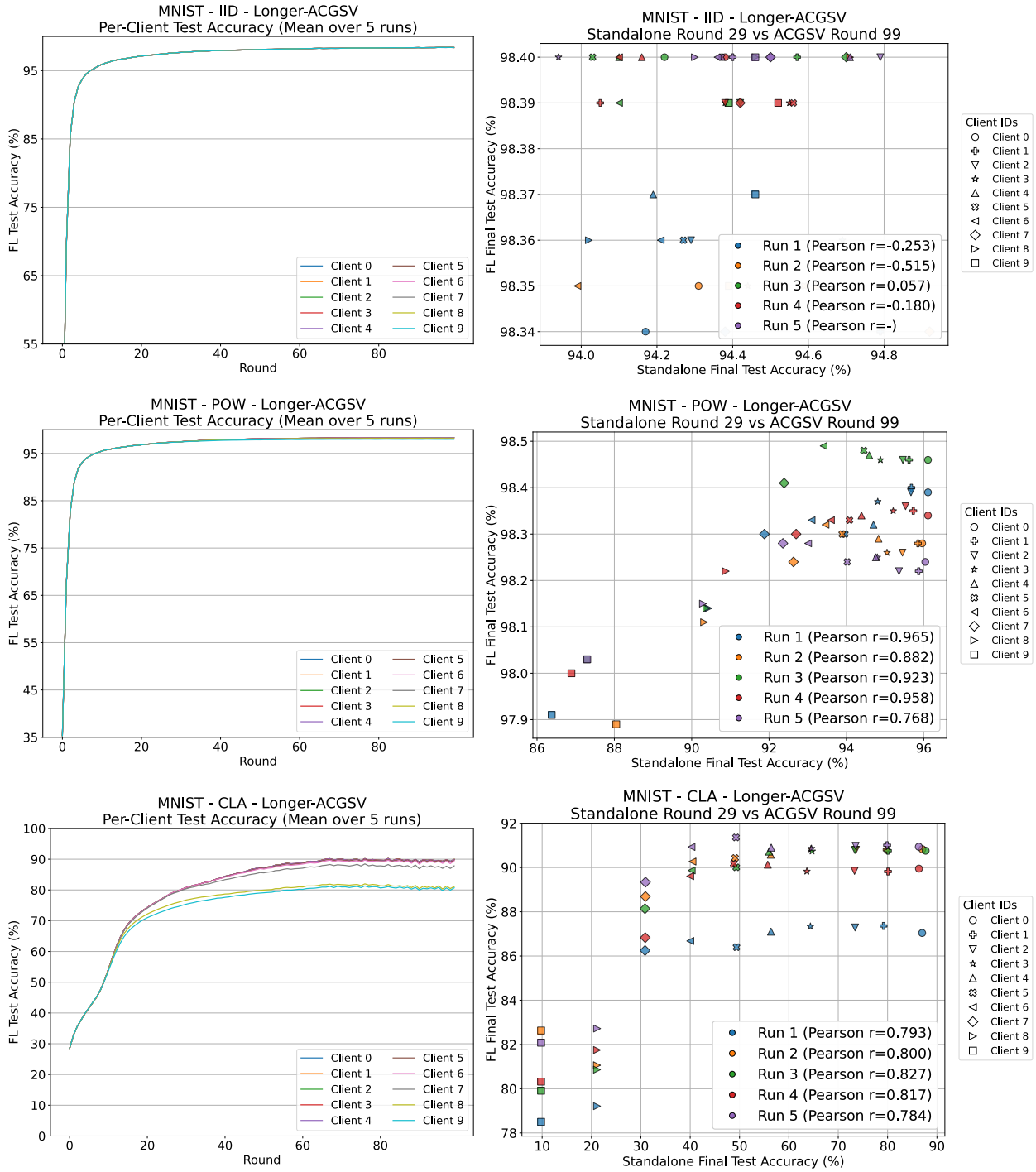


Figure C.1: Per-client test accuracy results for a longer training process with the ACGSV framework on MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

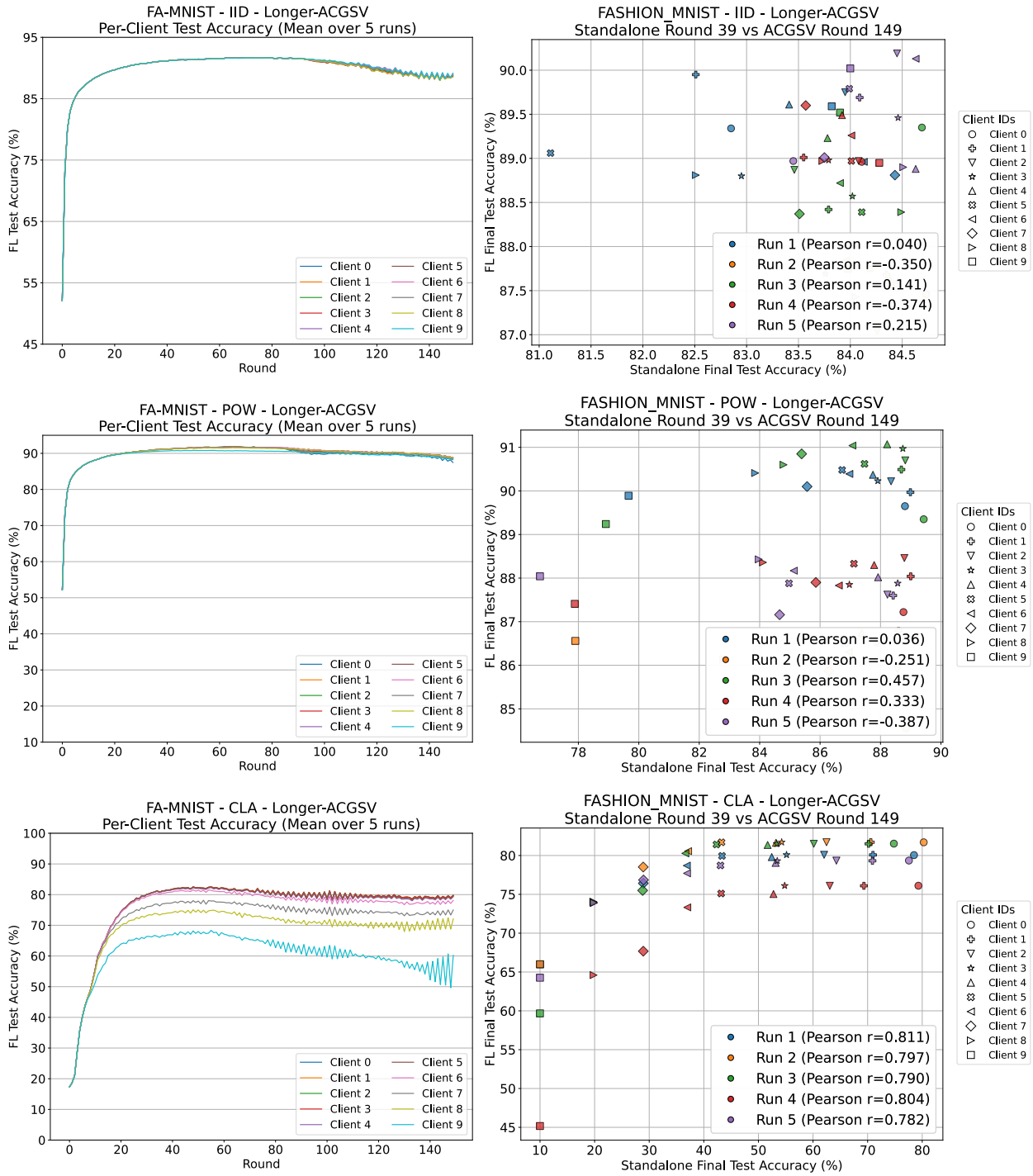


Figure C.2: Per-client test accuracy results for a longer training process with the ACGSV framework on FA-MNIST under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

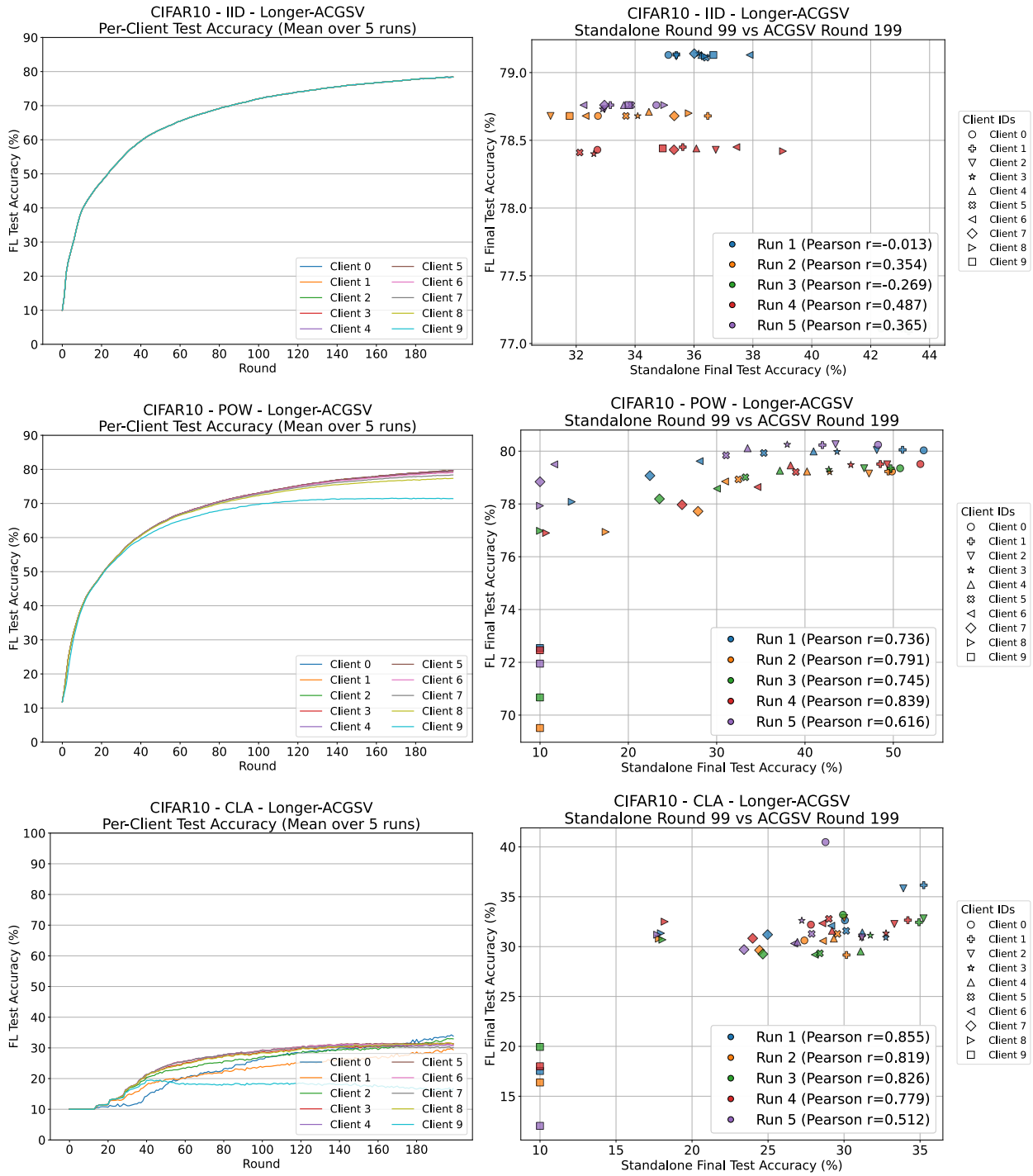


Figure C.3: Per-client test accuracy results for a longer training process with the ACGSV framework on CIFAR10 under IID, POW, and CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

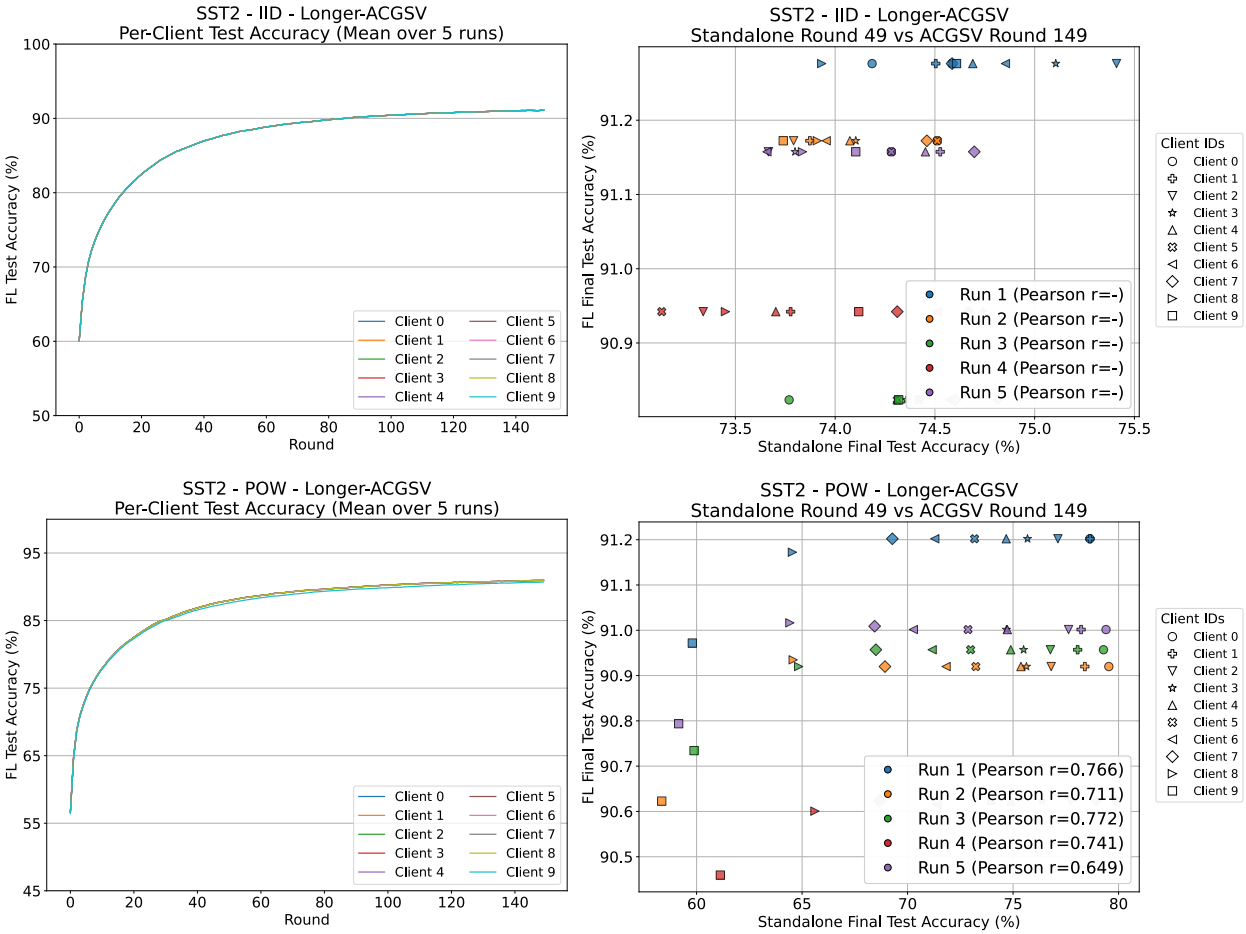


Figure C.4: Per-client test accuracy results for a longer training process with the ACGSV framework on SST2 under IID and POW data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

Appendix D

Run-by-Run Results

D.1 CFFL

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.953	0.872	0.045	0.986	0.855	0.284	0.990	0.968	0.529	0.932
r_2	-	0.956	0.872	-0.008	0.973	0.865	0.685	0.985	0.957	-0.287	0.921
r_3	-	0.951	0.873	0.292	0.986	0.889	0.519	0.997	0.948	0.381	0.905
r_4	-	0.970	0.870	-0.176	0.980	0.845	0.135	0.985	0.959	0.212	0.908
r_5	-	0.958	0.877	-0.160	0.985	0.894	0.250	0.989	0.925	-0.215	0.936

Table D.1: Per-run Pearson's r correlation coefficients obtained with the CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	-	0.943	1.000	0.072	0.956	0.911	0.182	1.000	0.956	0.378	1.000
τ_2	-	0.911	1.000	0.097	0.911	0.956	0.545	1.000	0.956	-0.316	0.956
τ_3	-	0.867	1.000	0.230	0.956	0.956	0.367	1.000	0.867	0.430	1.000
τ_4	-	0.867	1.000	-0.138	0.911	0.867	-	1.000	0.956	0.105	0.956
τ_5	-	0.989	1.000	-0.047	0.956	1.000	0.263	1.000	1.000	-0.149	1.000

Table D.2: Per-run Kendall's τ correlation coefficients obtained with the CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Run	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.000	1.416	33.342	0.035	1.712	29.768	0.013	8.163	13.349	0.008	5.116
GMD_2	0.000	1.184	33.490	0.019	1.625	29.318	0.009	8.234	12.666	0.010	4.978
GMD_3	0.000	1.303	33.791	0.024	1.691	29.782	0.015	8.763	12.636	0.005	5.074
GMD_4	0.000	1.133	33.182	0.034	1.853	28.799	0.014	8.311	13.419	0.007	4.638
GMD_5	0.000	1.276	33.407	0.028	1.984	29.563	0.016	8.080	12.659	0.013	5.179

Table D.3: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the CFFL framework under IID, POW, and CLA data partitioning.

Run	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	3.453	3.278	6.811	8.480	2.229	6.737	50.080	38.902	3.534	15.579	11.982
$MPAD_2$	3.332	3.181	7.605	7.607	2.511	6.168	52.056	38.625	4.636	15.888	11.914
$MPAD_3$	3.683	3.109	7.444	7.691	1.853	6.262	43.131	39.833	3.235	15.976	12.163
$MPAD_4$	3.677	3.192	7.791	7.655	2.273	6.432	50.738	38.025	4.507	16.097	11.357
$MPAD_5$	3.587	3.215	7.526	7.436	2.878	5.088	52.466	45.847	5.360	16.340	12.037

Table D.4: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the CFFL framework under IID, POW, and CLA data partitioning.

D.2 ACGSV

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-0.015	0.984	0.506	-	0.740	0.578	0.319	0.837	0.574	-	0.655
r_2	-0.591	0.931	0.508	0.334	0.616	0.533	-	0.800	0.592	-	0.644
r_3	-0.625	0.956	0.578	0.705	0.843	0.580	-	0.848	0.576	-	0.641
r_4	-0.104	0.912	0.562	-	0.704	0.560	-	0.779	0.580	-	0.637
r_5	0.075	0.921	0.530	-0.086	0.889	0.597	-0.045	0.887	0.500	-	0.704

Table D.5: Per-run Pearson's r correlation coefficients obtained with the ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	0.050	0.874	0.270	-	0.053	0.764	0.258	0.680	0.943	-	0.304
τ_2	-0.447	0.598	0.263	0.293	-0.097	0.597	-	0.675	0.911	-	0.181
τ_3	-0.422	0.930	0.549	0.596	0.453	0.920	-	0.215	0.841	-	0.365
τ_4	-0.138	0.822	0.322	-	0.579	0.828	-	0.215	0.932	-	0.435
τ_5	0.050	0.919	0.225	-0.149	0.677	0.931	0.033	0.907	0.874	-	0.615

Table D.6: Per-run Kendall's τ correlation coefficients obtained with the ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.002	0.115	1.085	0.000	0.079	5.300	0.007	0.233	4.636	0.000	0.060
GMD_2	0.002	0.101	1.000	0.009	0.057	4.397	0.000	0.285	4.957	0.000	0.085
GMD_3	0.005	0.133	1.450	0.004	0.107	6.165	0.000	0.243	4.238	0.000	0.070
GMD_4	0.007	0.088	1.184	0.000	0.112	6.077	0.000	0.243	4.473	0.000	0.110
GMD_5	0.002	0.097	1.097	0.002	0.078	4.300	0.005	0.310	3.874	0.000	0.052

Table D.7: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the ACGSV framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	2.992	4.254	27.803	7.823	4.904	33.393	37.684	39.496	19.531	13.081	15.265
$MPAD_2$	2.963	3.870	29.188	7.036	4.673	33.374	38.726	40.306	21.418	14.020	15.680
$MPAD_3$	3.226	3.972	30.280	7.087	4.300	33.726	30.372	40.446	18.924	13.745	15.454
$MPAD_4$	3.160	4.026	30.568	6.815	4.663	33.221	37.380	38.349	20.607	14.012	14.762
$MPAD_5$	2.948	3.971	29.750	6.885	5.249	34.515	40.836	47.093	19.865	14.169	16.432

Table D.8: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the ACGSV framework under IID, POW, and CLA data partitioning.

D.3 Framework Variants Plots and Tables

D.3.1 Framework Variant: UR-ACGSV

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
Run 1	0.224	-0.205	0.270	-	0.856	0.396	-0.024	-0.072	0.778	-	0.415
Run 2	0.248	0.316	0.156	0.248	0.163	0.920	0.285	-0.023	0.600	-	-0.093
Run 3	-	0.304	-0.114	0.151	0.072	0.328	-0.149	-0.341	0.778	-	0.615
Run 4	-	0.102	0.378	0.452	-0.285	0.920	-0.526	-0.097	0.733	-	0.028
Run 5	-0.232	-0.327	-0.138	0.188	0.298	0.215	-0.204	0.434	0.778	-	0.054

Table D.9: Per-run Kendall's τ correlation coefficients obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.004	0.056	0.330	0.000	0.104	1.249	0.024	0.415	5.104	0.000	0.067
GMD_2	0.002	0.039	0.246	0.002	0.089	0.558	0.011	12.788	5.395	0.000	0.068
GMD_3	0.000	0.071	0.216	0.002	0.107	2.008	0.004	0.881	4.360	0.000	0.073
GMD_4	0.000	0.023	0.244	0.002	0.063	1.650	0.013	1.280	4.762	0.000	0.044
GMD_5	0.007	0.038	0.253	0.004	0.087	1.168	0.013	1.083	6.320	0.000	0.008

Table D.10: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	3.595	4.755	21.305	6.923	3.939	16.386	50.003	51.690	19.599	15.160	18.223
$MPAD_2$	3.363	4.339	21.031	5.559	4.120	12.489	52.241	44.648	24.227	16.151	18.050
$MPAD_3$	3.753	4.545	22.990	6.214	3.363	19.981	42.834	52.322	18.525	15.839	18.168
$MPAD_4$	3.667	4.409	22.268	5.904	4.063	17.459	50.434	50.373	20.108	16.054	17.581
$MPAD_5$	3.412	4.585	24.264	5.956	4.458	27.169	52.877	58.261	22.917	16.166	18.448

Table D.11: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-0.447	0.712	0.525	-	0.928	0.520	0.055	0.756	0.652	-	0.621
r_2	-0.290	0.680	0.485	-0.339	0.672	0.546	0.214	0.659	0.591	0.0	0.631
r_3	-	0.882	-0.064	0.0278	0.739	0.456	-0.373	0.752	0.675	-	0.643
r_4	-	0.698	0.715	0.352	0.379	0.461	0.670	0.728	0.710	-	0.594
r_5	-0.066	0.554	-0.211	-0.058	0.814	0.495	-0.372	0.767	0.560	0.0	0.264

Table D.12: Per-run Pearson's r correlation coefficients obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	-0.447	-0.153	0.270	-	0.856	0.396	0.097	-0.072	0.956	-	0.415
τ_2	-0.348	0.316	0.156	-0.348	0.109	0.920	0.130	-0.023	0.956	-	-0.093
τ_3	-	0.304	-0.114	0.050	0.167	0.328	-0.348	-0.315	0.867	-	0.615
τ_4	-	0.102	0.378	0.447	-0.389	0.920	0.415	-0.097	0.956	-	0.028
τ_5	-0.086	-0.327	-0.138	-0.075	0.348	0.215	-0.126	0.442	1.000	-	0.054

Table D.13: Per-run Kendall's τ correlation coefficients obtained with the UR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

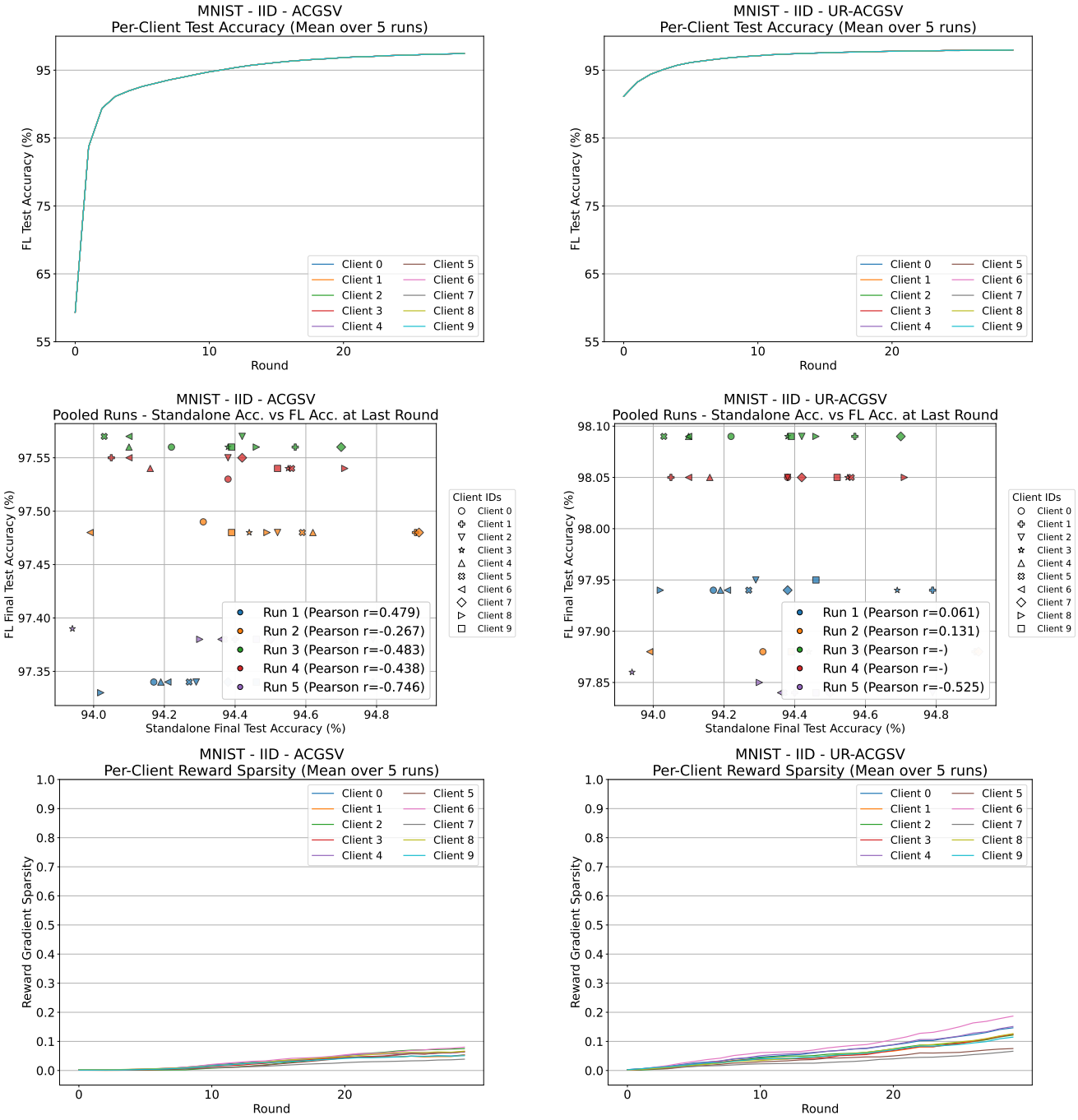


Figure D.1: Per-client results for the ACGSV framework and the UR-ACGSV variant on MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

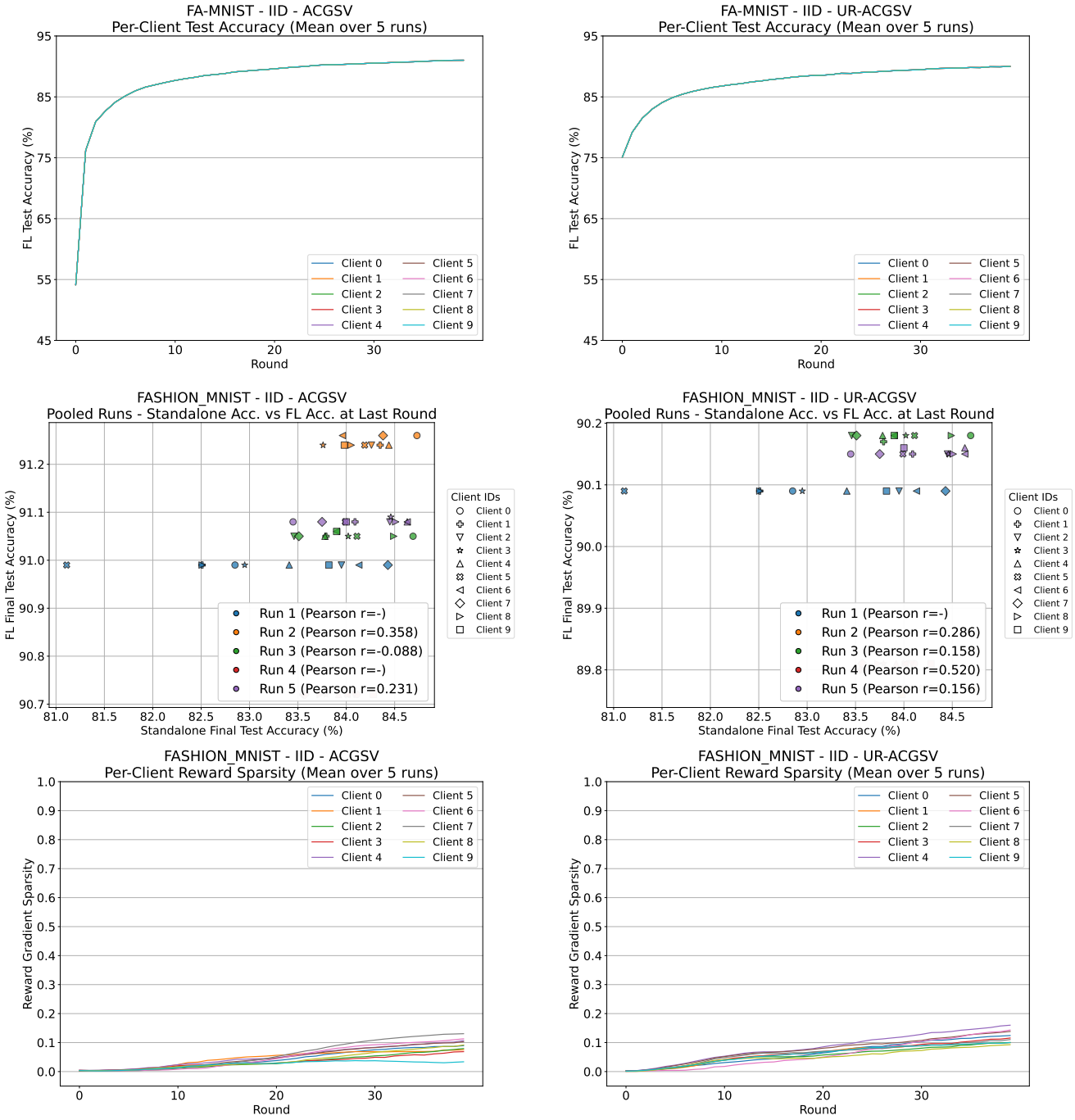


Figure D.2: Per-client results for the ACGSV framework and the UR-ACGSV variant on FA-MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

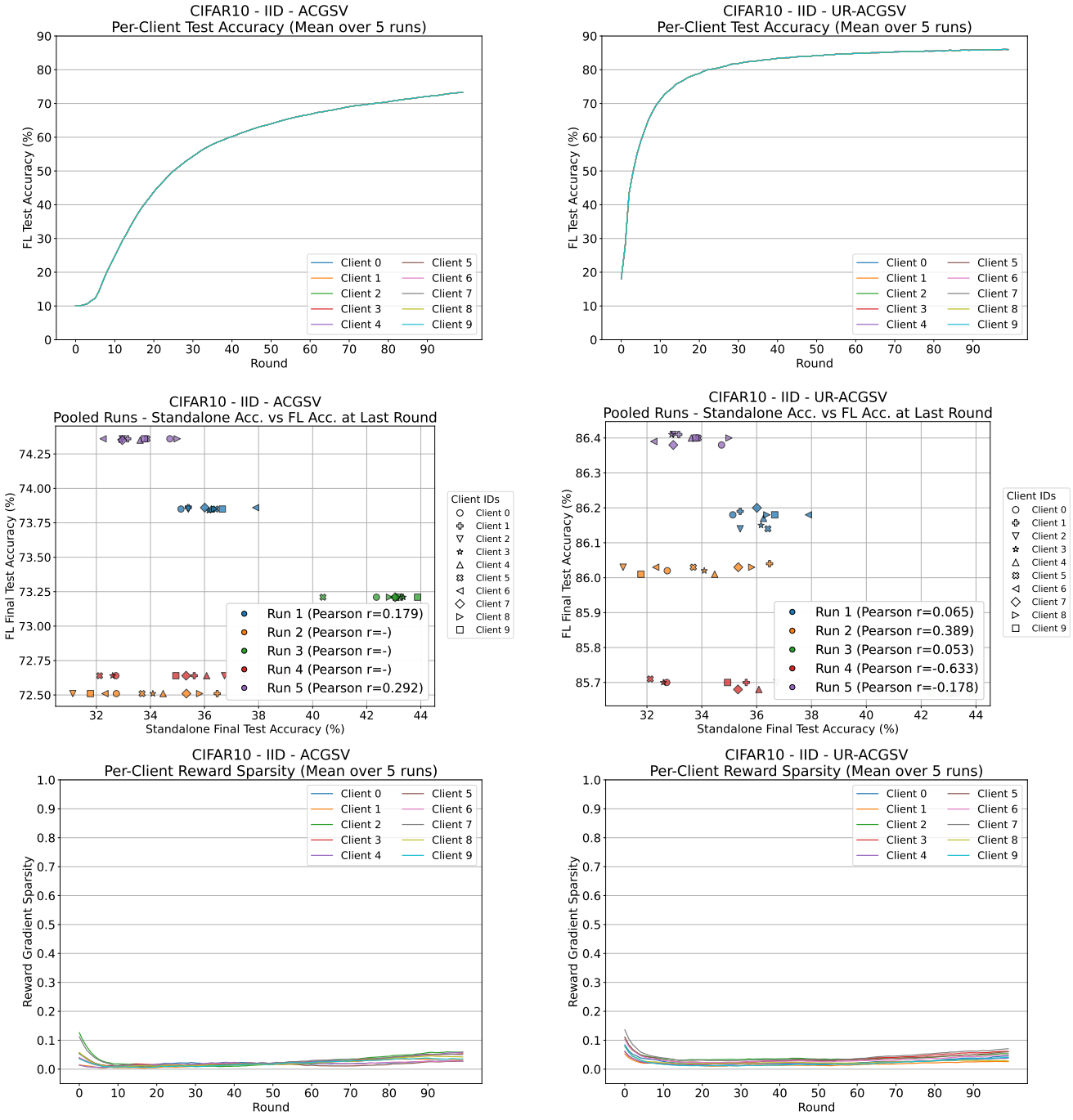


Figure D.3: Per-client results for the ACGSV framework and the UR-ACGSV variant on CIFAR10 under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

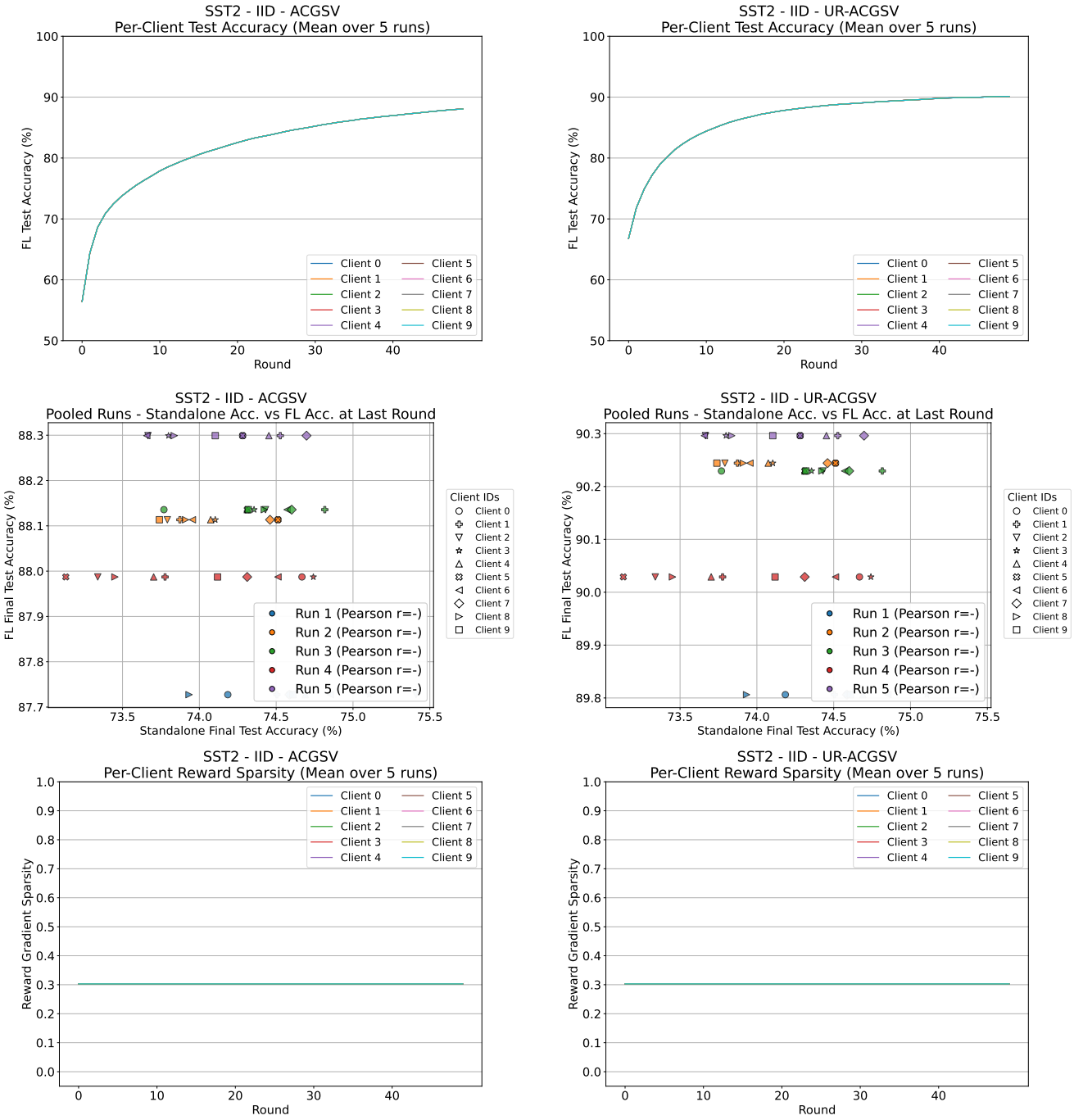


Figure D.4: Per-client results for the ACGSV framework and the UR-ACGSV variant on SST2 under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the UR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

D.3.2 Framework Variant: OTOPK-CFFL

Runs	MNIST			FA-MNIST			CIFAR10			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.900	0.793	-0.432	0.894	0.803	0.266	0.776	0.970	-	0.825
r_2	-	0.873	0.786	-	0.911	0.763	0.315	0.735	0.987	-	0.831
r_3	-	0.866	0.776	-0.411	0.904	0.789	-0.598	0.698	0.967	-	0.825
r_4	-	0.954	0.672	-	0.923	0.761	0.245	0.754	0.961	-	0.808
r_5	-	0.868	0.681	-	0.899	0.725	0.085	0.485	0.979	-	0.776

Table D.14: Per-run Pearson's r correlation coefficients obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.867	0.867	-0.362	0.659	0.989	0.269	0.494	0.778	-	0.740
r_2	-	0.899	0.778	-	0.809	0.944	0.270	0.719	0.644	-	0.532
r_3	-	0.841	1.000	-0.362	-0.023	0.911	-0.337	0.449	0.911	-	0.740
r_4	-	0.899	0.225	-	0.092	0.899	0.119	0.467	0.764	-	0.828
r_5	-	0.764	0.289	-	0.796	0.899	0.050	0.552	0.867	-	0.883

Table D.15: Per-run Kendall's τ correlation coefficients obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.000	1.320	6.413	0.006	1.388	11.024	0.016	18.551	8.266	0.000	2.820
GMD_2	0.000	1.247	4.778	0.000	1.709	9.196	0.013	15.436	10.032	0.000	3.122
GMD_3	0.000	1.158	6.328	0.005	1.360	8.248	0.010	18.791	9.852	0.000	2.676
GMD_4	0.000	0.896	5.151	0.000	1.289	9.372	0.023	17.613	9.634	0.000	2.463
GMD_5	0.000	1.528	5.543	0.000	1.569	6.258	0.013	15.721	8.321	0.000	2.832

Table D.16: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	3.693	3.937	33.048	8.508	4.682	31.362	50.037	41.121	13.384	15.650	16.632
$MPAD_2$	3.382	3.782	35.127	7.330	4.670	30.431	52.336	43.213	15.067	15.973	16.415
$MPAD_3$	3.523	3.847	35.022	7.792	3.905	32.067	42.917	42.876	12.152	15.787	17.124
$MPAD_4$	3.777	4.029	36.303	7.855	4.940	31.104	50.757	41.138	13.235	16.091	16.031
$MPAD_5$	3.547	3.747	35.923	7.644	5.297	32.765	51.992	50.052	14.169	16.315	16.482

Table D.17: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.880	0.586	0.367	0.826	0.583	0.274	0.859	0.742	-	0.734
r_2	-	0.830	0.587	-	0.747	0.543	-0.121	0.752	0.742	-	0.749
r_3	-	0.840	0.583	0.136	0.830	0.559	0.381	0.866	0.774	-	0.732
r_4	-	0.936	0.465	-	0.708	0.532	0.447	0.809	0.739	-	0.756
r_5	-	0.874	0.476	-	0.799	0.508	-0.037	0.789	0.733	-	0.718

Table D.18: Per-run Pearson's r correlation coefficients obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	-	0.822	0.867	0.302	0.705	0.989	0.145	0.494	0.956	-	0.740
τ_2	-	0.899	0.778	-	0.809	0.944	-0.074	0.719	1.000	-	0.532
τ_3	-	0.887	1.000	0.163	0.068	0.911	0.285	0.467	0.822	-	0.740
τ_4	-	0.899	0.225	-	0.000	0.899	0.310	0.556	0.944	-	0.828
τ_5	-	0.809	0.289	-	0.796	0.899	-0.050	0.644	0.911	-	0.931

Table D.19: Per-run Kendall's τ correlation coefficients obtained with the OTOPK-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

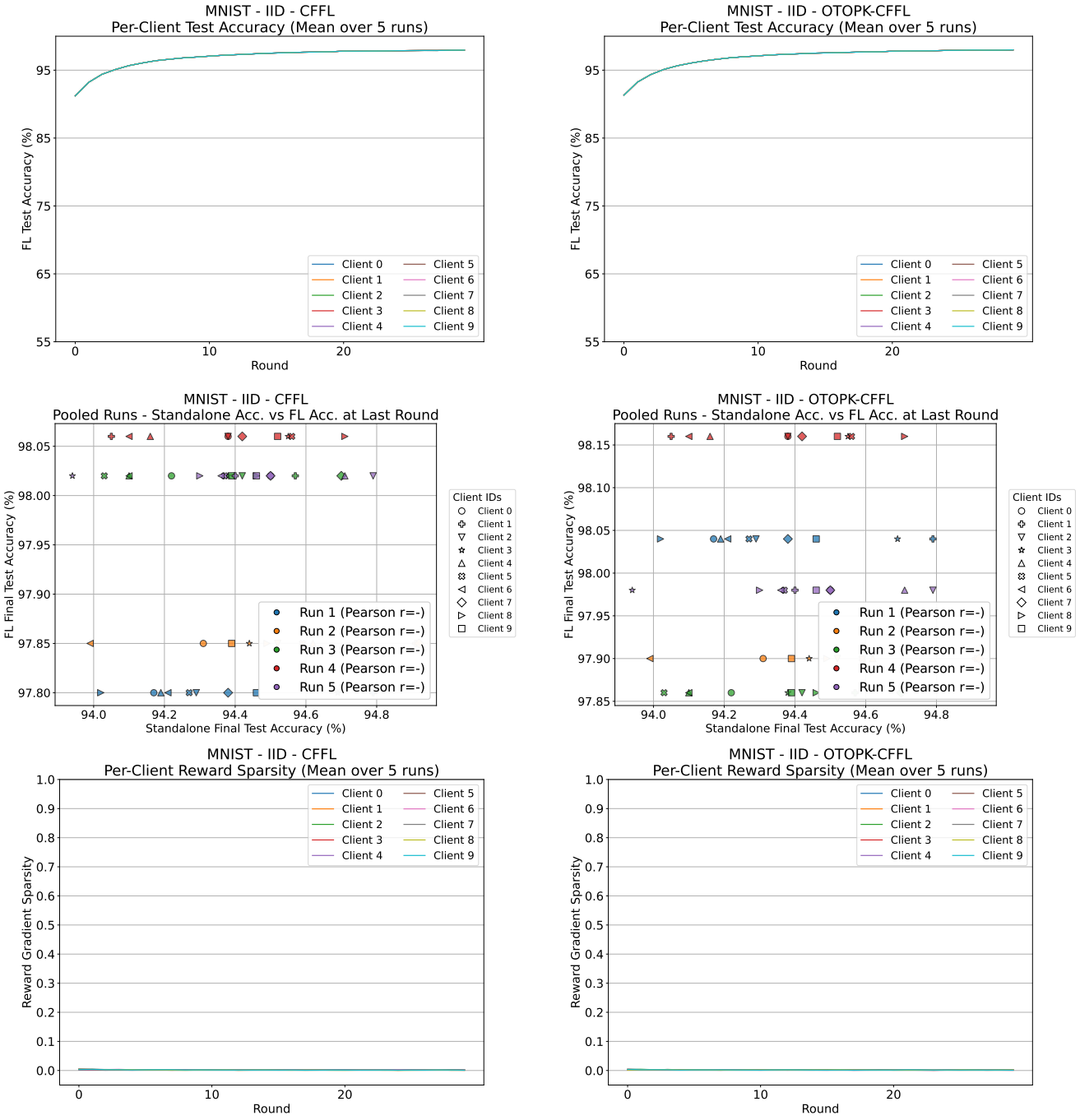


Figure D.5: Per-client results for the CFFL framework and the OTOPK-CFFL variant on MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

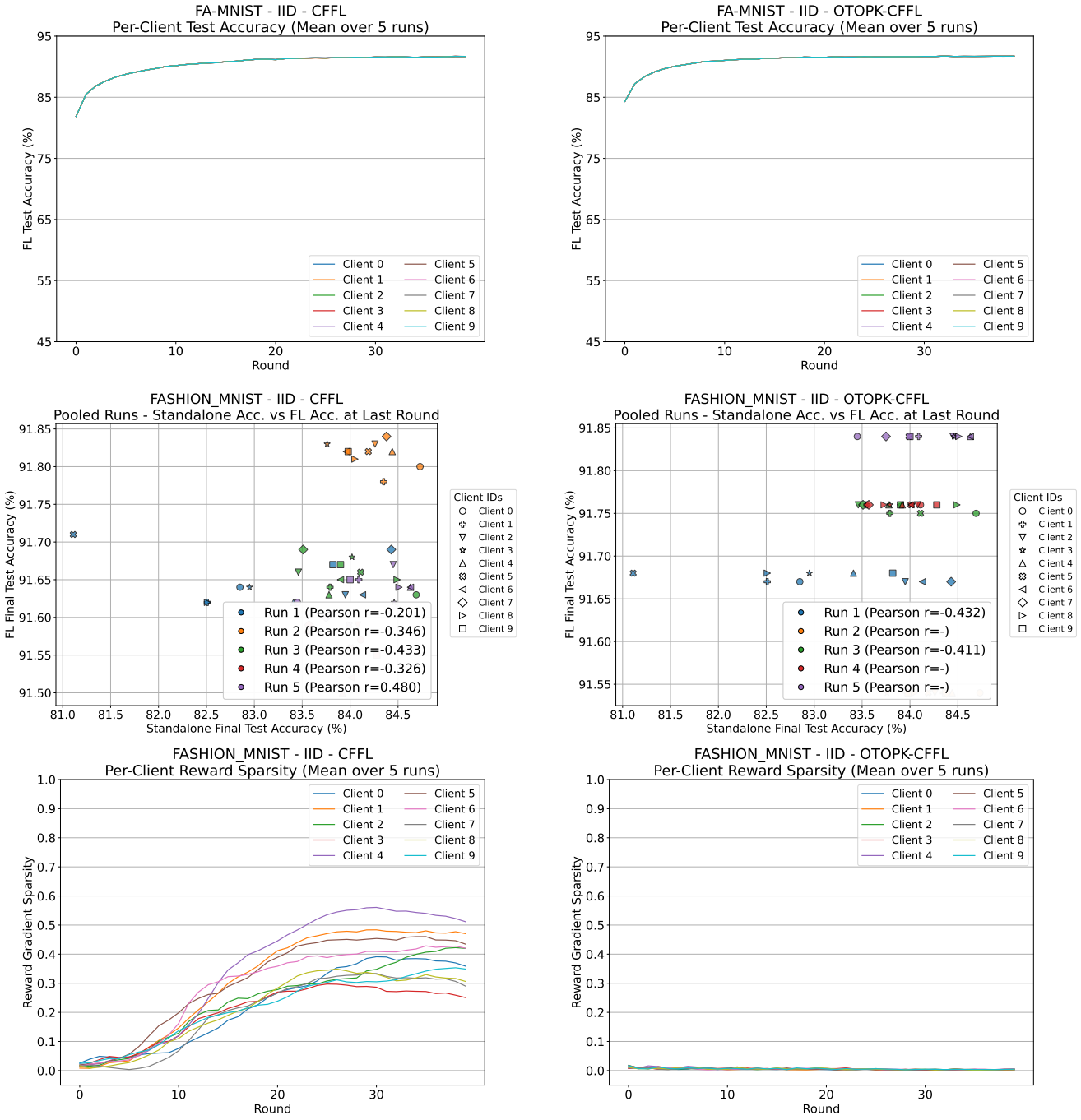


Figure D.6: Per-client results for the CFFL framework and the OTOPK-CFFL variant on FA-MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

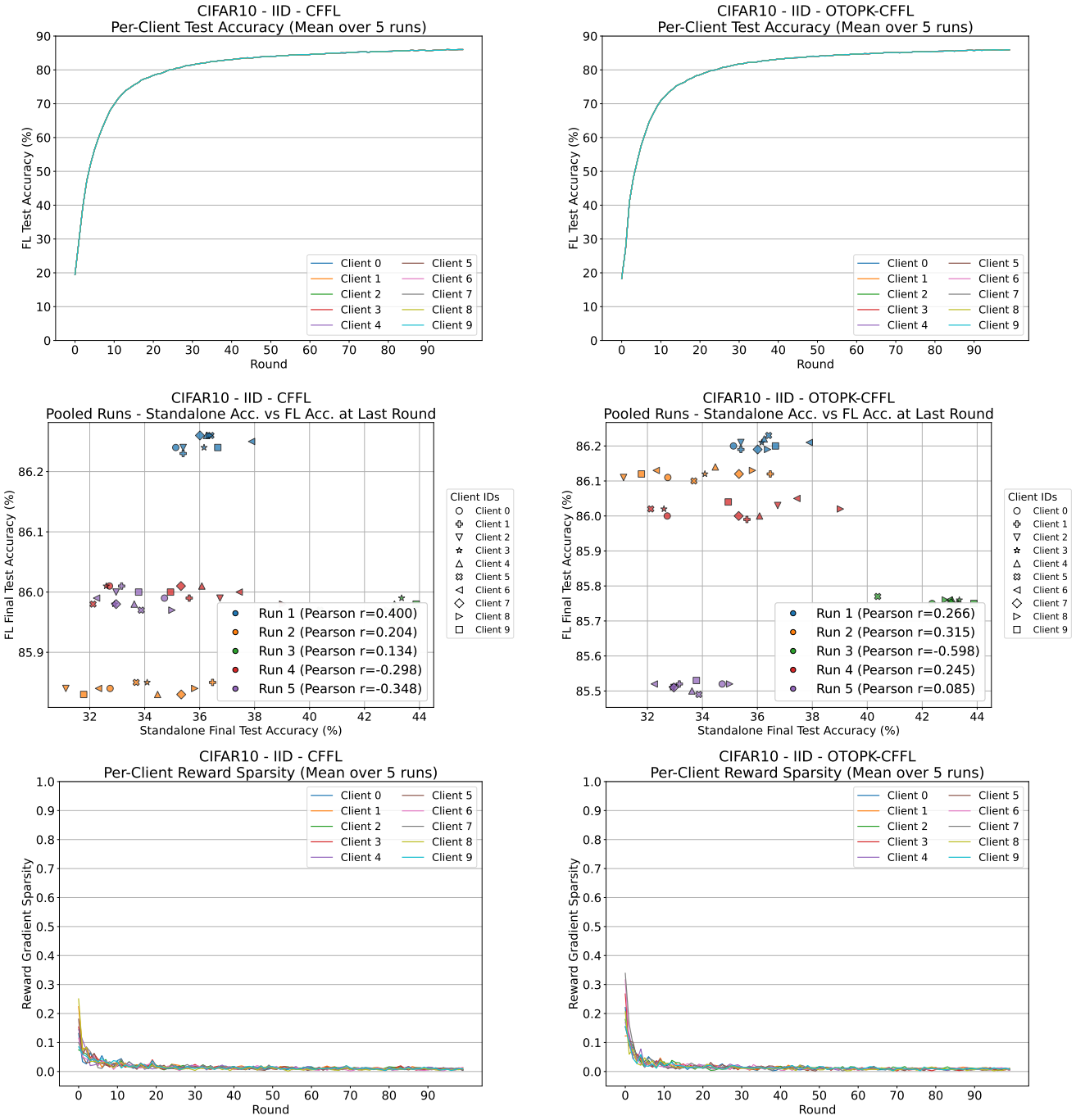


Figure D.7: Per-client results for the CFFL framework and the OTOPK-CFFL variant on CIFAR10 under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

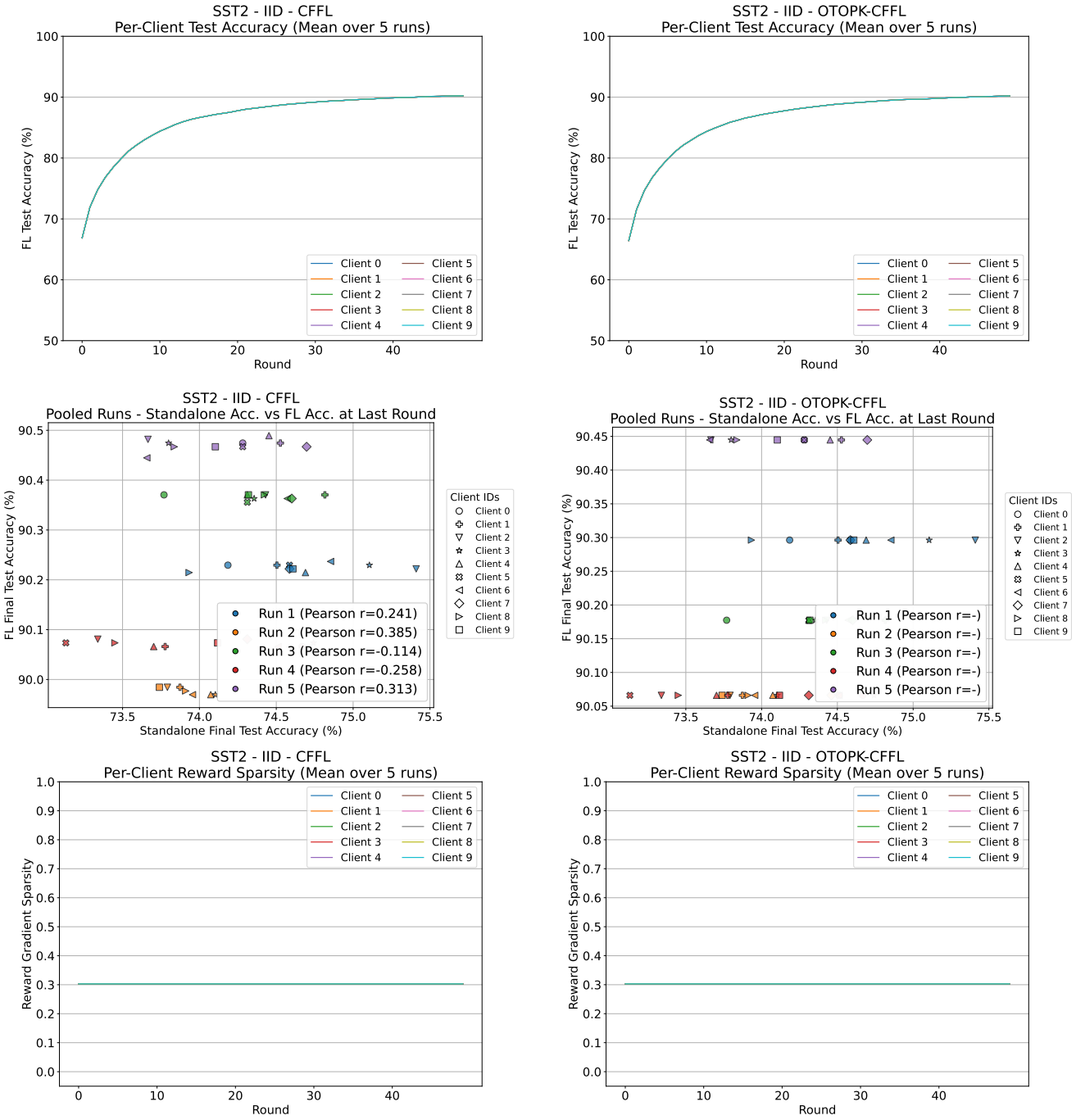


Figure D.8: Per-client results for the CFFL framework and the OTOPK-CFFL variant on SST2 under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the OTOPK-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training, the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis, and the third row shows the mean per-client reward-gradient sparsity during FL training.

D.3.3 Framework Variant: DPR-CFFL

Runs	MNIST			FA-MNIST			CIFAR10			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.957	0.975	-	0.929	0.927	-	0.989	0.908	0.720	0.970
r_2	-	0.939	0.975	-	0.893	0.982	-	0.987	0.852	0.665	0.969
r_3	-	0.945	0.982	-	0.886	0.969	-	0.988	0.914	-0.249	0.987
r_4	-	0.957	0.962	-	0.881	0.973	-	0.973	0.877	-0.050	0.991
r_5	-	0.953	0.971	-	0.870	0.986	-	0.928	0.928	0.502	0.992

Table D.20: Per-run Pearson's r correlation coefficients obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations. Mean GMD reports the average pairwise difference between clients' final FL test accuracies, averaged across runs and measured in percentage points.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
Run 1	-	0.867	1.000	-	0.822	0.867	-	1.000	0.764	0.539	0.956
Run 2	-	0.899	1.000	-	0.822	1.000	-	1.000	0.600	0.494	1.000
Run 3	-	0.911	1.000	-	0.867	0.956	-	0.989	0.778	-0.135	1.000
Run 4	-	0.899	1.000	-	0.778	0.956	-	0.911	0.733	0.022	1.000
Run 5	-	0.778	0.956	-	0.822	1.000	-	0.920	0.778	0.270	0.956

Table D.21: Per-run Kendall's τ correlation coefficients obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.000	0.578	31.070	0.000	1.411	27.677	0.000	6.457	13.577	0.432	2.492
GMD_2	0.000	0.544	30.274	0.000	1.204	28.336	0.000	6.048	12.812	0.411	2.118
GMD_3	0.000	0.620	32.281	0.000	1.349	28.573	0.000	5.695	12.969	0.256	2.215
GMD_4	0.000	0.482	30.897	0.000	1.424	29.602	0.000	5.620	13.544	0.565	2.379
GMD_5	0.000	0.522	30.012	0.000	1.316	30.249	0.000	6.317	12.565	0.300	2.325

Table D.22: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	3.593	3.755	9.234	8.603	2.620	9.697	50.252	40.011	3.726	6.574	13.335
$MPAD_2$	3.302	3.576	9.761	7.610	2.639	6.873	52.516	40.082	4.733	6.961	13.569
$MPAD_3$	3.593	3.513	9.103	8.265	2.090	8.741	43.142	41.741	3.595	6.642	13.442
$MPAD_4$	3.687	3.542	10.705	7.525	2.505	6.854	50.880	39.815	4.649	7.344	12.530
$MPAD_5$	3.527	3.556	10.203	7.954	2.948	6.905	52.499	46.712	5.491	6.867	13.515

Table D.23: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning.

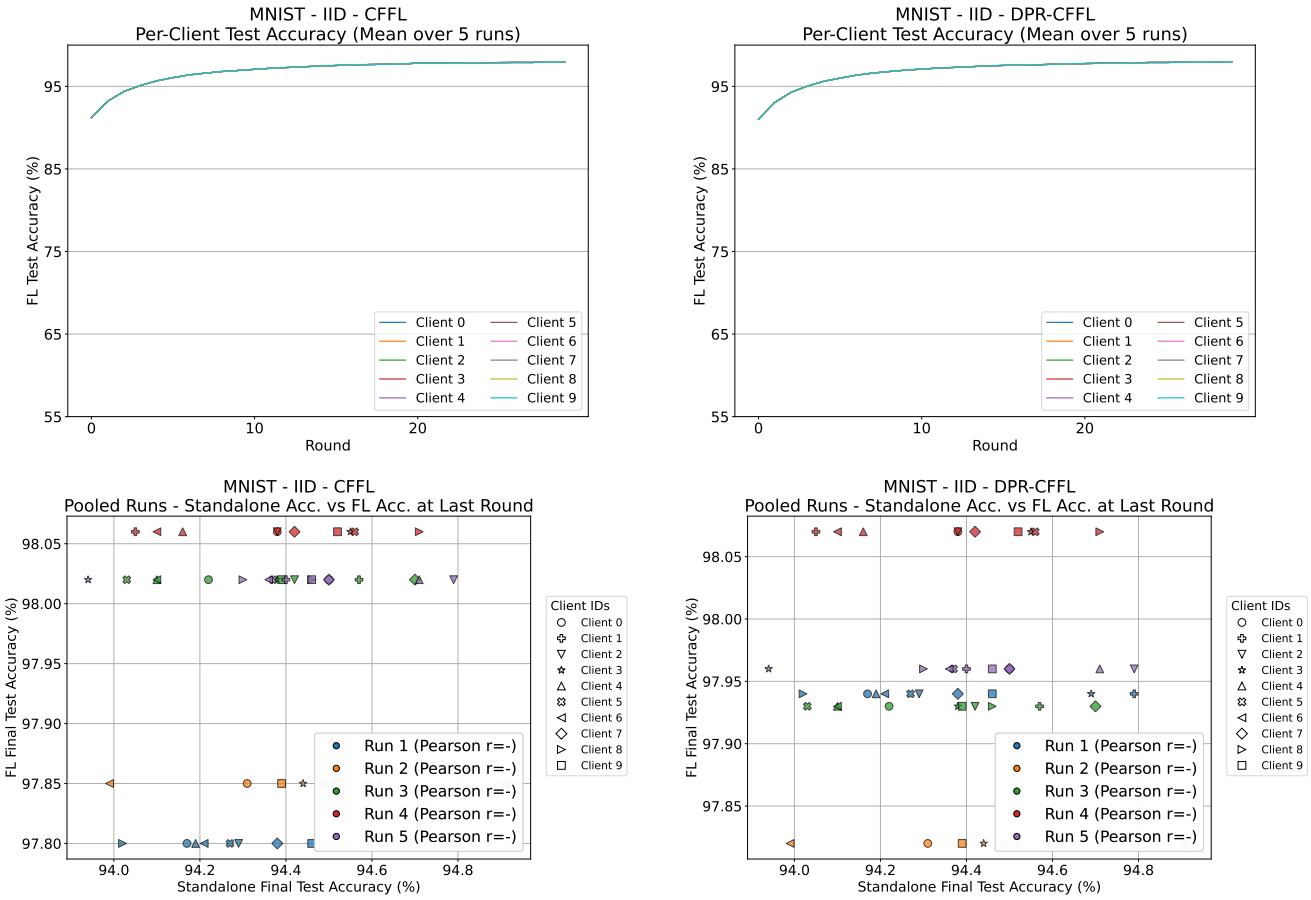


Figure D.9: Per-client results for the CFFL framework and the DPR-CFFL variant on MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

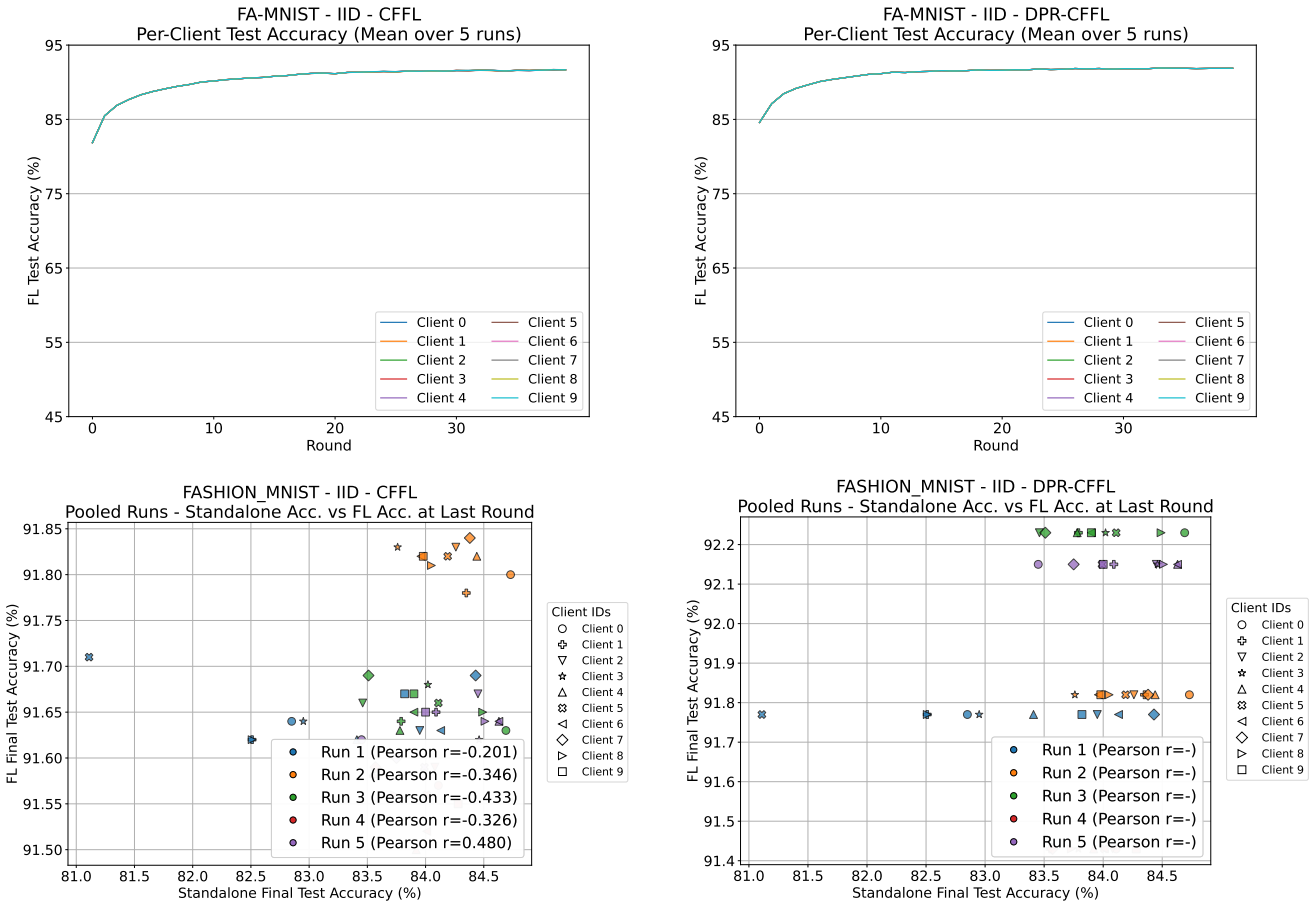


Figure D.10: Per-client results for the CFFL framework and the DPR-CFFL variant on FA-MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

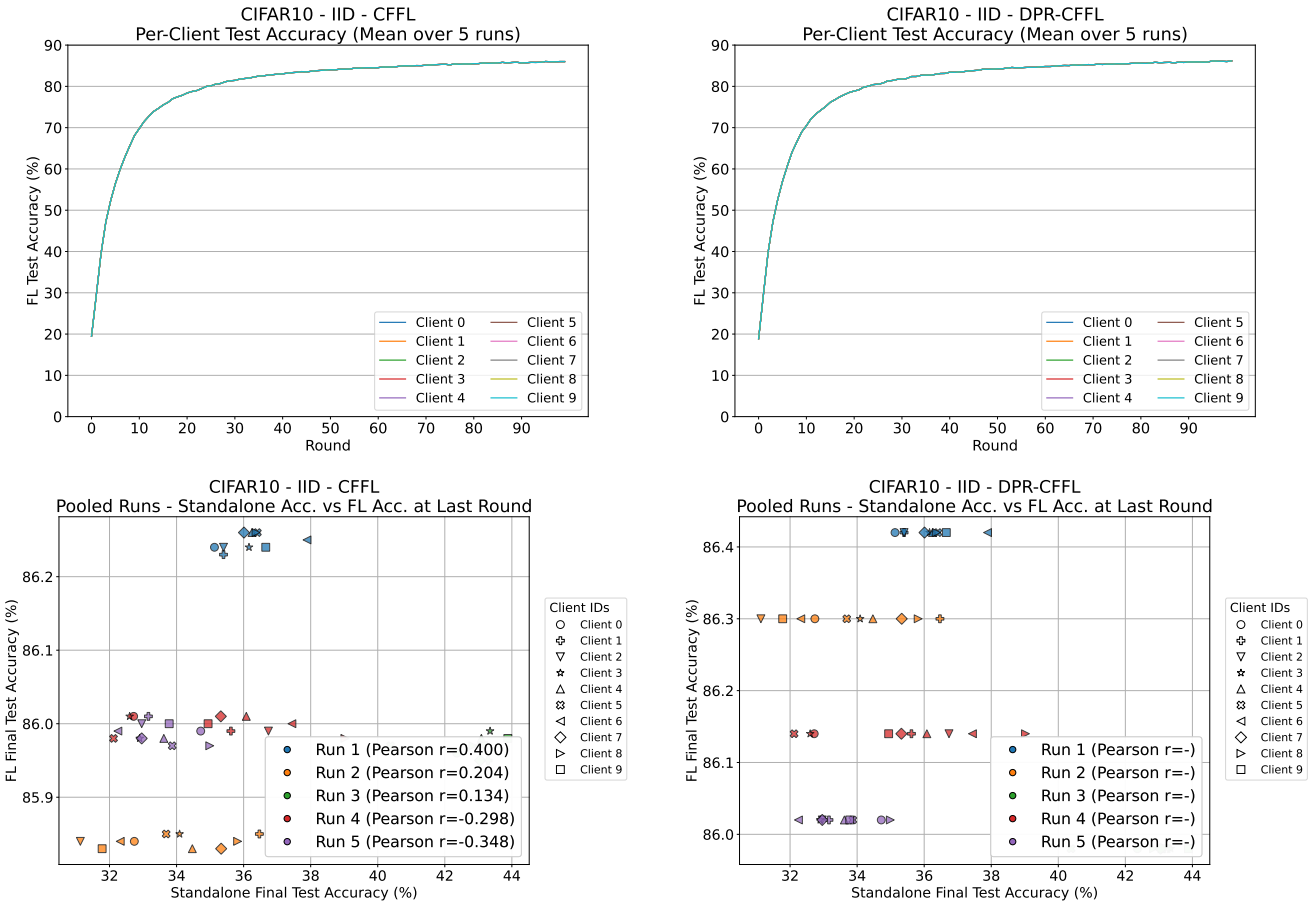


Figure D.11: Per-client results for the CFFL framework and the DPR-CFFL variant on CIFAR10 under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

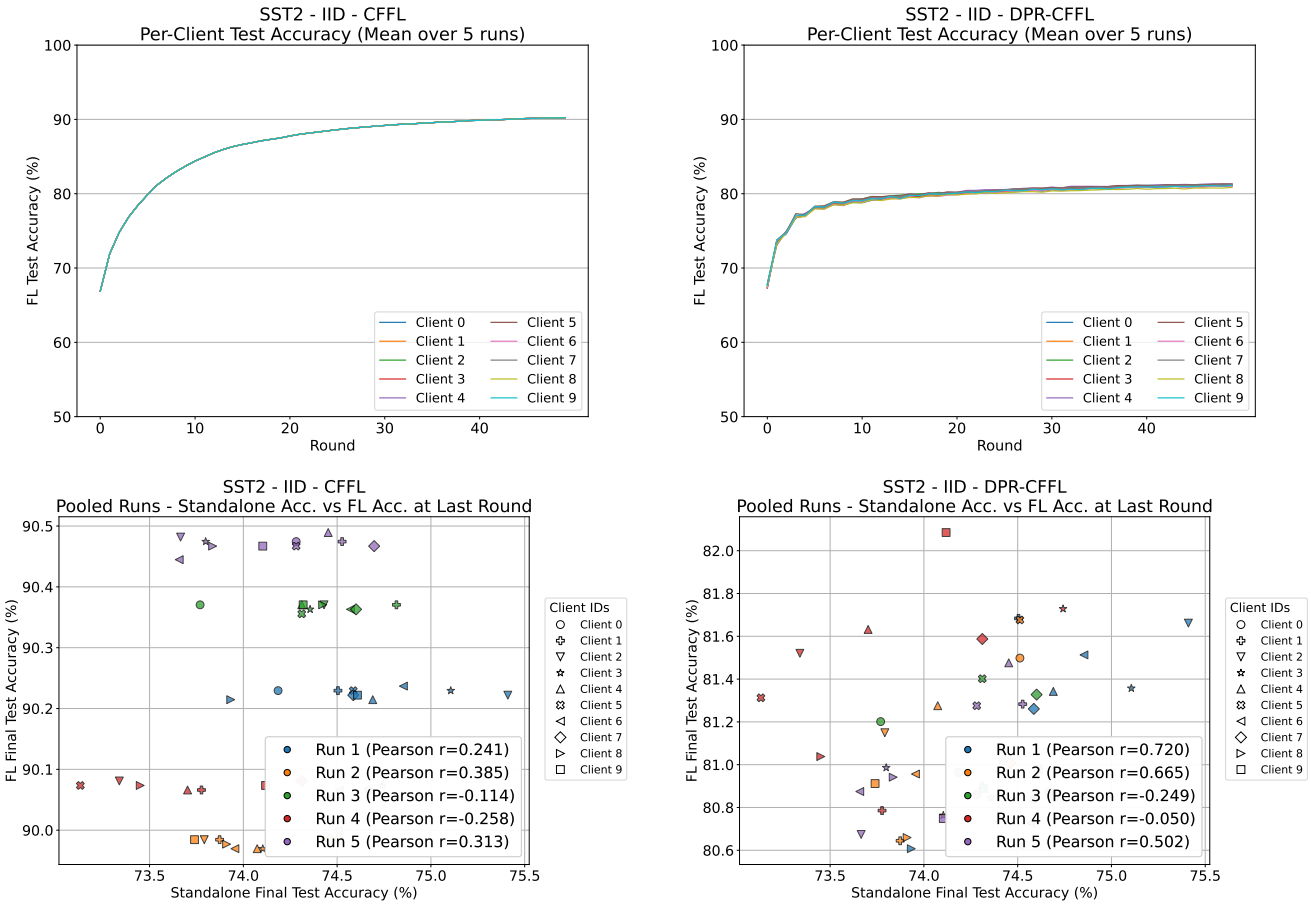


Figure D.12: Per-client results for the CFFL framework and the DPR-CFFL variant on SST2 under IID data partitioning. Columns distinguish the framework: the left column reports the original CFFL results, while the right column reports the DPR-CFFL results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-	0.963	0.850	-	0.965	0.801	-	0.997	0.970	0.517	0.990
r_2	-	0.961	0.861	-	0.968	0.874	-	0.987	0.958	-0.101	0.986
r_3	-	0.961	0.865	-	0.933	0.848	-	0.957	0.953	0.043	0.989
r_4	-	0.965	0.835	-	0.975	0.853	-	0.979	0.960	0.471	0.993
r_5	-	0.967	0.863	-	0.971	0.870	-	0.987	0.917	0.395	0.997

Table D.24: Per-run Pearson's r correlation coefficients obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	-	0.911	1.000	-	0.867	0.867	-	1.000	0.944	0.333	1.000
τ_2	-	0.899	1.000	-	0.867	1.000	-	1.000	0.956	-0.244	1.000
τ_3	-	0.956	1.000	-	0.956	0.956	-	1.000	0.867	0.244	1.000
τ_4	-	0.899	1.000	-	0.956	0.956	-	1.000	0.956	0.244	1.000
τ_5	-	0.822	0.956	-	0.956	1.000	-	1.000	1.000	0.289	1.000

Table D.25: Per-run Kendall's τ correlation coefficients obtained with the DPR-CFFL framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

D.3.4 Framework Variant: DFR-ACGSV

Runs	MNIST			FA-MNIST			CIFAR10			SST2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	-0.332	0.990	1.000	0.432	-0.240	1.000	0.108	0.988	0.959	0.162	0.996
r_2	0.687	0.993	1.000	-0.266	0.769	1.000	0.312	0.984	0.899	0.540	0.998
r_3	-0.199	0.989	1.000	-0.160	0.969	1.000	-0.279	0.972	0.928	-0.220	0.996
r_4	0.062	0.983	1.000	-0.352	0.076	1.000	-0.303	0.970	0.944	0.396	0.997
r_5	0.207	0.992	1.000	0.077	0.769	1.000	-0.158	0.898	0.964	0.519	0.995

Table D.26: Per-run Pearson's r correlation coefficients obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between the standalone test accuracy and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
Run 1	-0.200	1.000	1.000	0.135	-0.244	1.000	0.045	1.000	0.778	0.180	0.956
Run 2	0.600	0.867	1.000	-0.200	-0.022	1.000	0.289	1.000	0.600	0.295	1.000
Run 3	-0.090	1.000	1.000	-0.135	0.689	1.000	0.022	0.989	0.778	-	1.000
Run 4	0.045	1.000	1.000	-0.360	-0.067	1.000	-0.333	0.911	0.733	0.289	1.000
Run 5	-	1.000	1.000	-	0.111	1.000	-0.090	0.920	0.822	0.341	0.956

Table D.27: Per-run Kendall's τ correlation coefficients obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
GMD_1	0.194	2.676	31.393	0.642	0.525	25.544	1.183	12.520	10.083	0.325	5.992
GMD_2	0.191	2.173	31.469	0.807	2.832	24.433	1.033	12.180	10.453	0.346	5.475
GMD_3	0.216	2.498	31.456	0.868	1.656	25.580	1.706	12.076	11.050	0.314	5.962
GMD_4	0.269	2.696	31.085	0.951	1.412	26.235	0.983	12.084	10.548	0.460	5.629
GMD_5	0.234	2.346	31.508	1.266	1.144	25.692	1.106	12.312	10.641	0.426	6.039

Table D.28: Per-run Gini Mean Difference (GMD) values in percentage points obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
$MPAD_1$	1.532	1.692	0.163	-1.087	-9.845	-2.117	28.273	27.720	0.671	5.401	5.470
$MPAD_2$	1.460	1.637	0.325	-3.451	-8.384	-2.766	32.020	28.382	2.632	5.867	5.813
$MPAD_3$	1.709	1.622	0.099	-4.005	-6.317	-0.550	20.716	29.934	1.956	5.249	5.987
$MPAD_4$	1.617	1.404	0.116	-1.403	-10.519	-1.834	30.727	28.497	1.912	5.718	4.513
$MPAD_5$	1.454	1.682	0.488	-6.729	-7.451	-1.530	31.750	35.116	3.840	5.796	4.818

Table D.29: Per-run Mean Paired Accuracy Difference (MPAD) values in percentage points obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning.

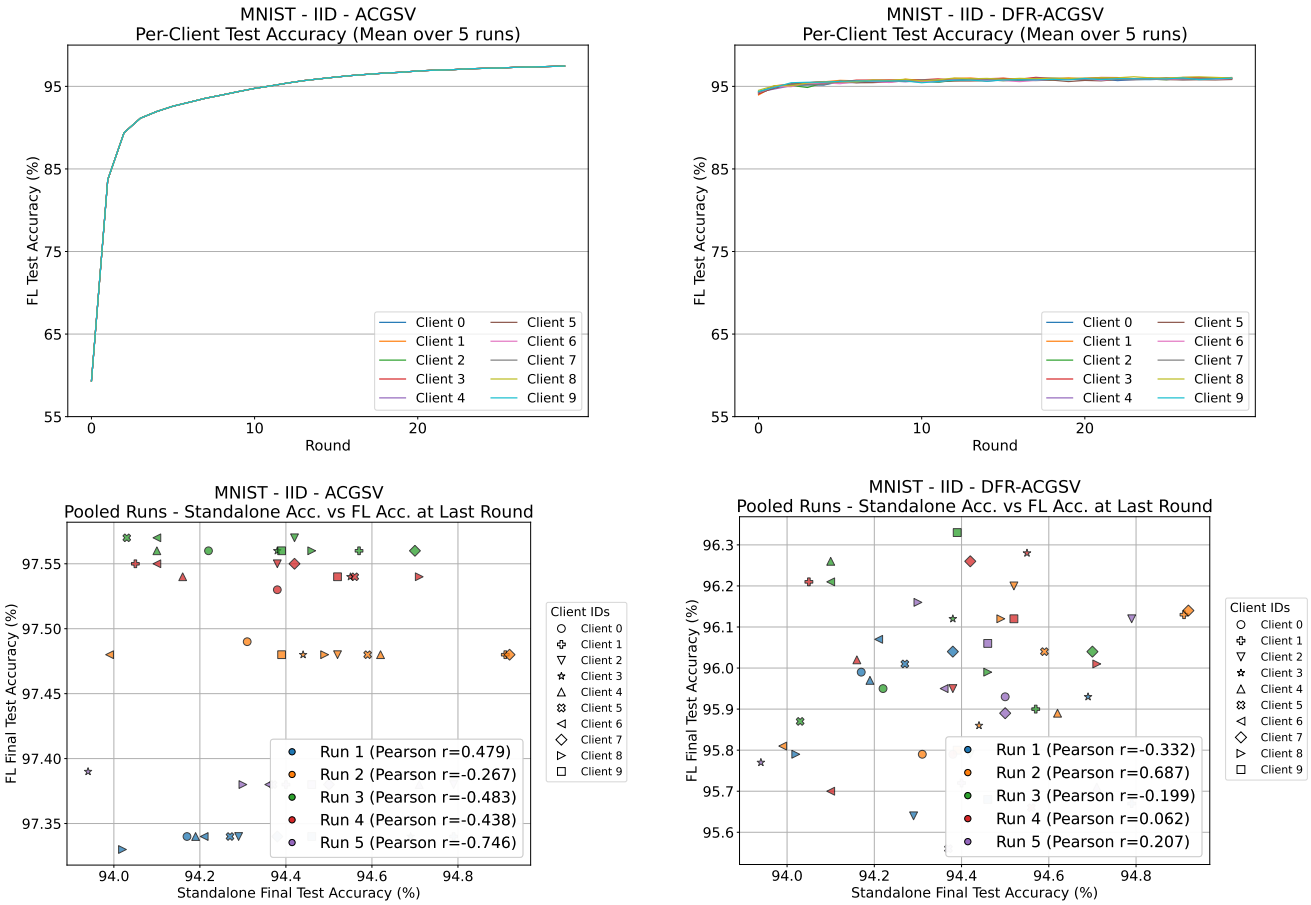


Figure D.13: Per-client results for the ACGSV framework and the DFR-ACGSV variant on MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

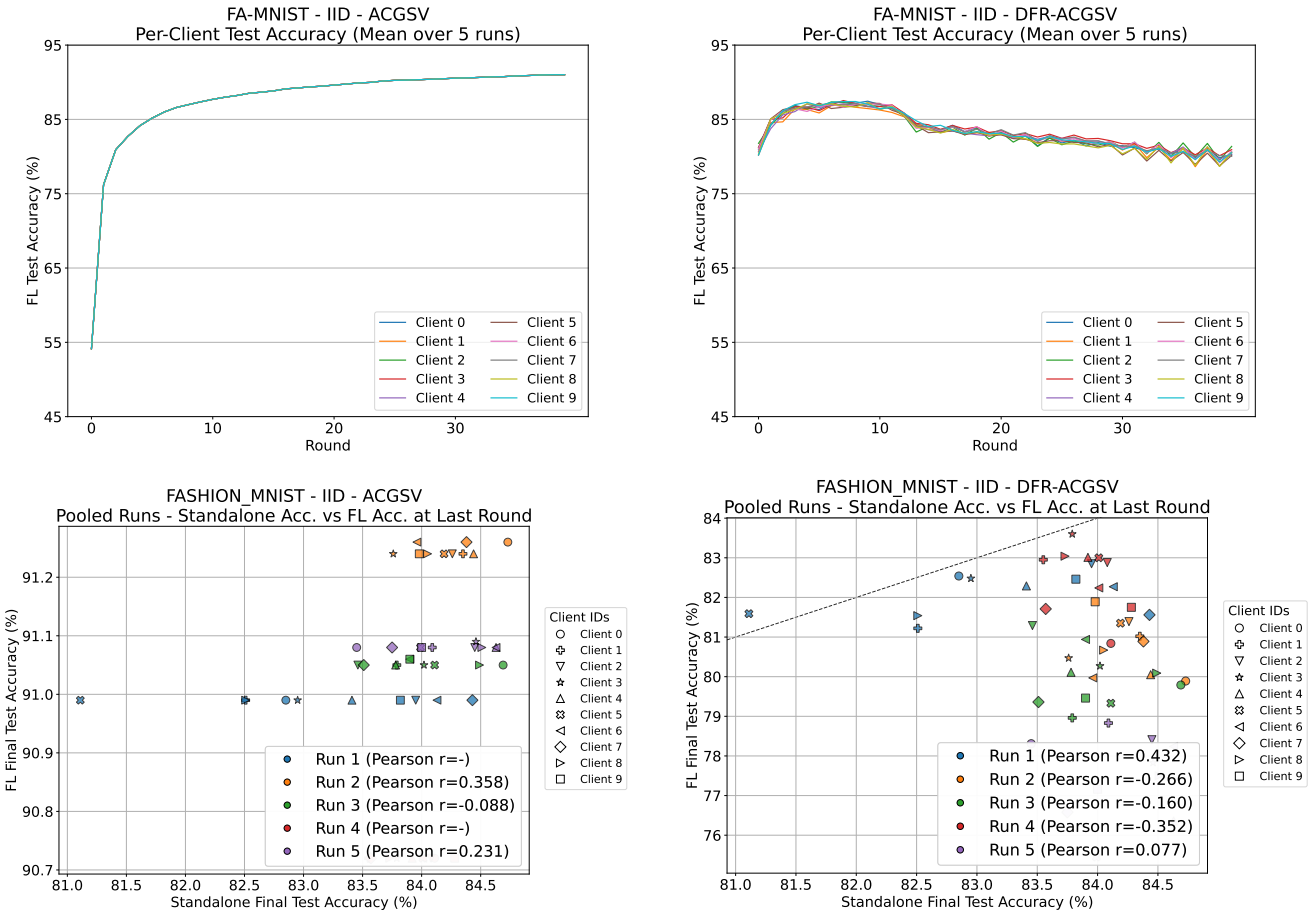


Figure D.14: Per-client results for the ACGSV framework and the DFR-ACGSV variant on FA-MNIST under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

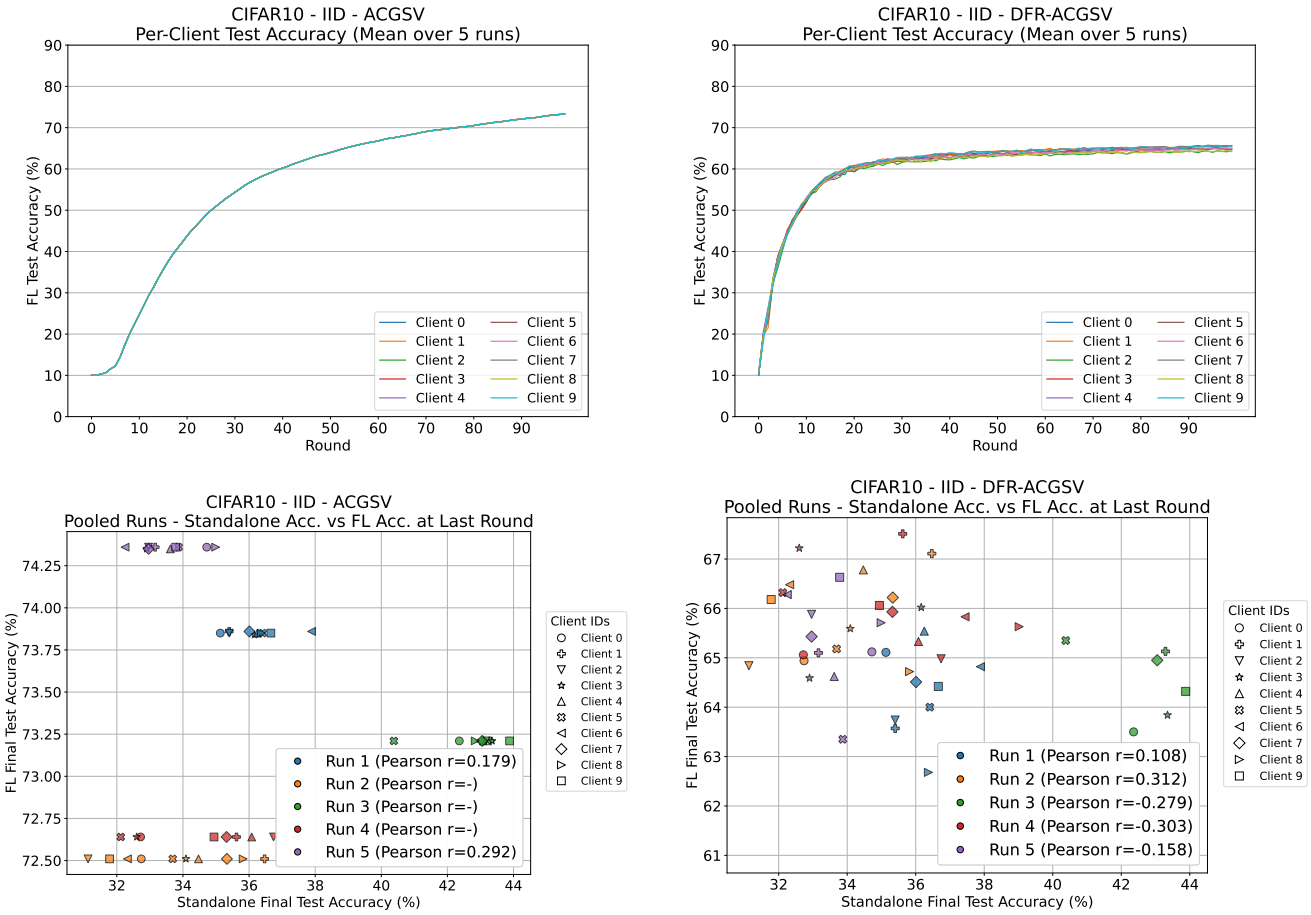


Figure D.15: Per-client results for the ACGSV framework and the DFR-ACGSV variant on CIFAR10 under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

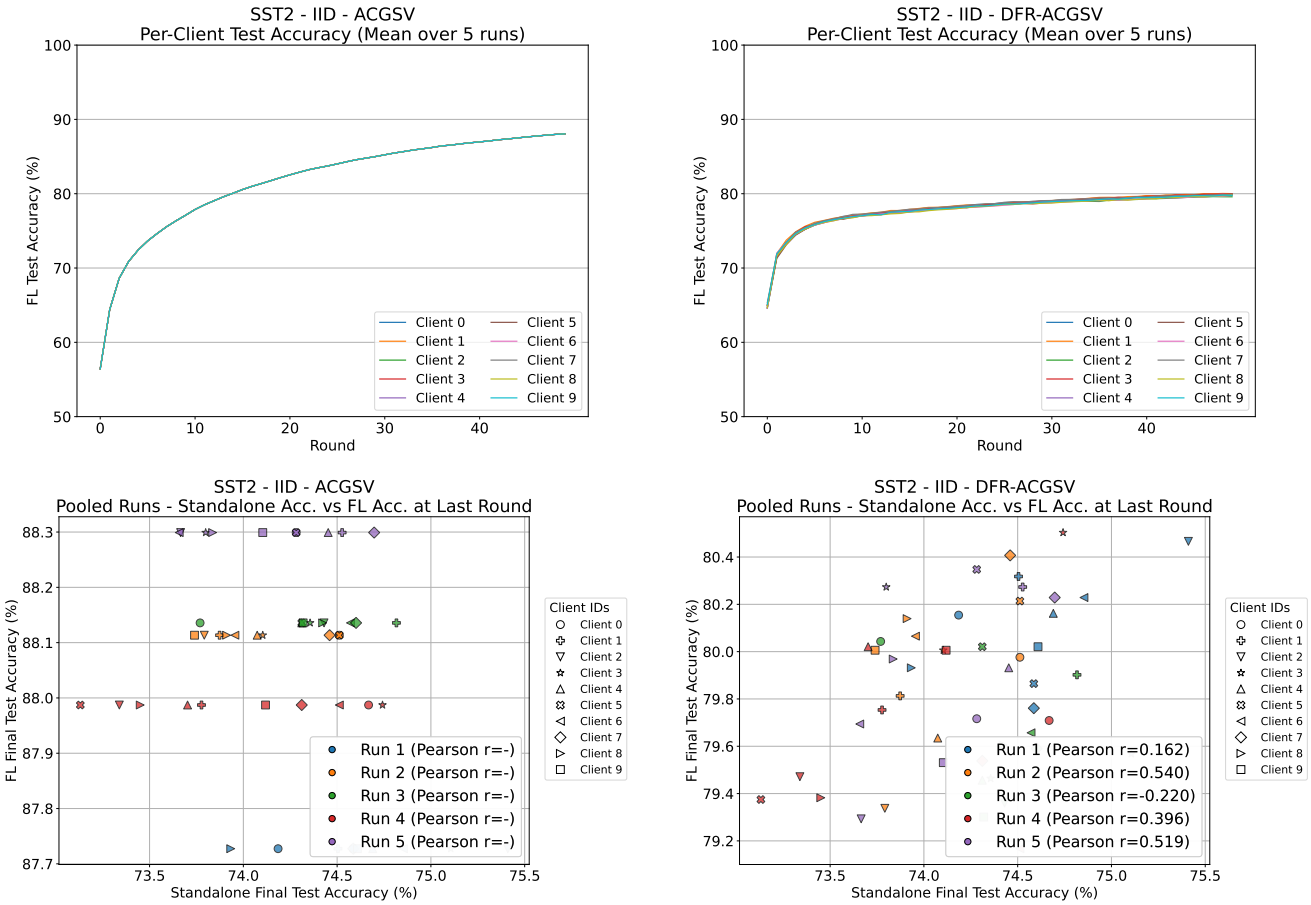


Figure D.16: Per-client results for the ACGSV framework and the DFR-ACGSV variant on SST2 under IID data partitioning. Columns distinguish the framework: the left column reports the original ACGSV results, while the right column reports the DFR-ACGSV results. Rows distinguish the chart type: the first row shows the mean per-client test accuracy during FL training and the second row shows scatter plots comparing each client's final Standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
r_1	0.712	0.986	0.932	0.462	-0.267	0.930	-0.338	0.996	0.930	0.373	0.985
r_2	0.601	0.979	0.934	-0.036	0.573	0.938	-0.273	0.996	0.929	0.527	0.990
r_3	0.423	0.985	0.935	-0.030	0.929	0.932	0.445	0.990	0.925	-0.035	0.977
r_4	0.397	0.975	0.935	0.328	-0.072	0.932	0.331	0.994	0.917	0.349	0.990
r_5	0.696	0.989	0.937	-0.371	0.594	0.928	0.110	0.996	0.887	-0.066	0.990

Table D.30: Per-run Pearson's r correlation coefficients obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Runs	MNIST			FA-MNIST			CIFAR-10			SST-2	
	IID	POW	CLA	IID	POW	CLA	IID	POW	CLA	IID	POW
τ_1	0.556	0.956	1.000	0.315	-0.289	1.000	-0.156	1.000	0.956	0.289	1.000
τ_2	0.378	0.867	1.000	-0.111	0.022	1.000	-0.156	1.000	0.956	0.360	1.000
τ_3	0.156	0.956	1.000	-0.022	0.689	1.000	0.378	1.000	0.867	-0.067	1.000
τ_4	0.333	1.000	1.000	0.422	-0.067	1.000	0.244	1.000	0.867	0.333	1.000
τ_5	0.467	0.956	1.000	-0.200	0.067	1.000	0.067	1.000	0.956	-0.090	1.000

Table D.31: Per-run Kendall's τ correlation coefficients obtained with the DFR-ACGSV framework under IID, POW, and CLA data partitioning. The correlation values were computed between each client's Shapley Values using test accuracy as data utility metric and the FL test accuracy values. Values reported as **-** indicate undefined correlations.

Appendix E

CFFL original CLA Aggregation Experiments

Dataset	η	γ	B	E	T	θ
MNIST	0.015	0.977	32	5	30	5
FA-MNIST	0.005	0.977	64	5	40	5
CIFAR10	0.01	0.977	64	1	100	5

Table E.1: Training hyperparameters used in the simulations with the CFFL frameworks without modifying the CLA equation. These values were applied only to the CLA partitioning scenarios. The parameters θ is exclusive to CFFL.

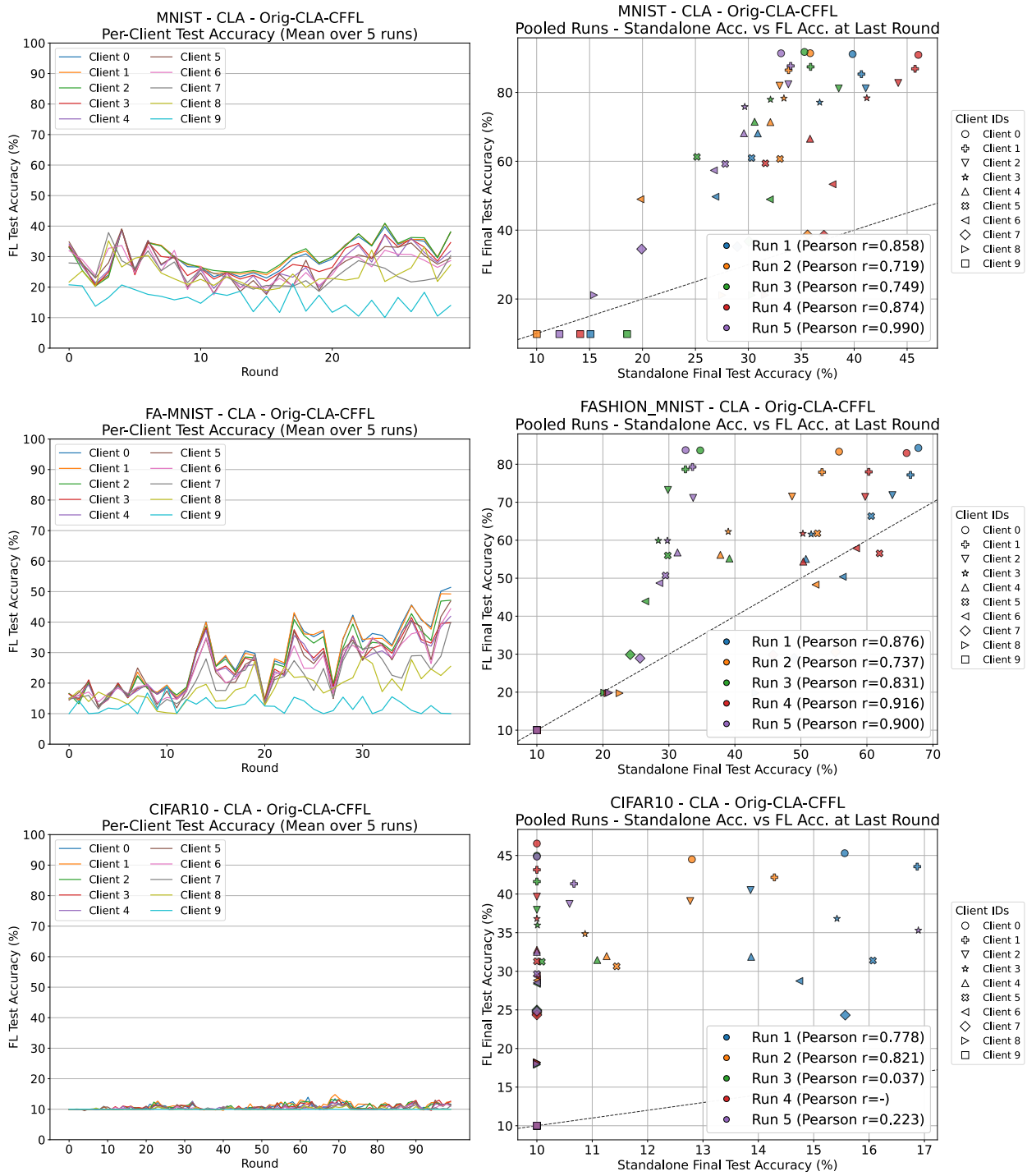


Figure E.1: Per-client test accuracy results for the CFFL framework on MNIST, FA-MNIST and CIFAR10 under CLA data partitioning. The left column shows the mean client test accuracies during FL training, while the right column compares each client's final standalone test accuracy on the x-axis with its final FL test accuracy on the y-axis.