Federated Optimization Algorithms: Methods and Results Analysis

Ravada Satyadev
ravada22398@iiitd.ac.in

November 18, 2025

1 Introduction

Federated Learning (FL) enables the collaborative training of a global model across multiple clients while keeping raw data local. Optimization choices, applied at the client and/or server level, critically affect convergence speed, stability under heterogeneous data distributions, communication efficiency, and final model quality. This document summarizes several prominent FL optimization strategies and analyzes their empirical behavior observed in an experimental study that tracked test accuracy, training loss, and test loss across communication rounds.

2 Algorithms

2.1 FedAvg

Federated Averaging (FedAvg) is the canonical optimization approach in FL. Each client k performs multiple local steps of stochastic gradient descent (SGD) on its private objective $F_k(\theta)$, producing an updated model $\theta_{t+1}^{(k)}$. A common local update step is

$$\theta_{t+1}^{(k)} = \theta_t - \eta \nabla F_k(\theta_t), \tag{1}$$

where η denotes the local learning rate. The server aggregates the client updates using a weighted average (weights often proportional to client dataset sizes n_k):

$$\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \theta_{t+1}^{(k)}, \qquad n = \sum_{k=1}^{K} n_k. \tag{2}$$

FedAvg is simple and communication-efficient but can be sensitive to non-IID client data, which may lead to client drift and slower or unstable convergence.

2.2 FedAdam

FedAdam adapts the centralized Adam optimizer to the server-side aggregation. Let g_t denote the aggregated model update (for example, the weighted average of client parameter deltas). The server maintains exponential moving averages of the first and second moments:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t, \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2. \tag{4}$$

The global parameter update uses these moments:

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}, \tag{5}$$

where α is the server learning rate and ϵ is a small constant for numerical stability. FedAdam provides per-parameter adaptive updates that can mitigate the effects of heterogeneous client update scales.

2.3 FedYogi

FedYogi employs the Yogi update rule on the server to avoid uncontrolled growth of the second-moment accumulator. The second-moment update is modified as

$$v_t = v_{t-1} - (1 - \beta_2)\,\mathrm{sign}(v_{t-1} - g_t^2)\,g_t^2, \tag{6}$$

with the first-moment update similar to Adam (m_t as above). The parameter update follows the Adam-like formula:

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}. \tag{7}$$

Yogi's conservative second-moment adjustments yield greater stability under high heterogeneity and noisy updates.

2.4 FedAMSGrad

FedAMSGrad applies the AMSGrad variant of Adam at the server by enforcing a non-decreasing second-moment sequence. After computing the standard second-moment estimate

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\,g_t^2, \tag{8}$$

one computes the running maximum

$$\widehat{v}_t = \max(\widehat{v}_{t-1}, v_t). \tag{9}$$

The global update uses $\widehat{v}_t$:

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{\widehat{v}_t} + \epsilon}. \tag{10}$$

AMSGrad has better theoretical convergence properties in certain non-convex settings and can stabilize optimization when updates fluctuate.

2.5 FedAdamW

FedAdamW incorporates the AdamW decoupled weight decay into the server optimization. Weight decay is applied independently from the adaptive gradient step:

$$\theta_t \leftarrow \theta_t - \alpha\lambda\theta_t \tag{11}$$

followed by the adaptive update (as in FedAdam):

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}. \tag{12}$$

Decoupled weight decay often improves generalization in deep models and can reduce overfitting when client datasets are small or biased.

2.6 ClientAdam with Server Averaging

In the hybrid approach, each client runs the Adam optimizer locally while the server aggregates client models via simple averaging (as in FedAvg). On client k, local adaptive updates use client-specific moment estimates:

$$m_t^{(k)} = \beta_1 m_{t-1}^{(k)} + (1-\beta_1)\, g_t^{(k)}, \tag{13}$$

$$v_t^{(k)} = \beta_2 v_{t-1}^{(k)} + (1-\beta_2)\left(g_t^{(k)}\right)^2, \tag{14}$$

$$\theta_{t+1}^{(k)} = \theta_t - \eta \frac{m_t^{(k)}}{\sqrt{v_t^{(k)}} + \epsilon}. \tag{15}$$

The server then computes

$$\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n}\, \theta_{t+1}^{(k)}. \tag{16}$$

This hybrid scheme leverages local adaptivity to accelerate per-client convergence while retaining the simple and robust global aggregation offered by averaging.

3 Experimental Setup

The experiments were conducted on a 25,000-sample subset of the FEMNIST dataset, a widely used benchmark for handwritten character recognition in federated learning. FEMNIST is naturally partitioned by writer identity, allowing each client to receive data that reflects personalized handwriting styles. To introduce controlled statistical heterogeneity, we employed a Dirichlet distribution with concentration parameter $\alpha = 0.5$, resulting in moderately non-IID local data distributions across clients.

A total of 10 clients participated in every communication round. Each client performed 5 local training epochs per round using a batch size of 32. The global model was trained for 100 communication rounds. During each round, clients updated their local models on their private data and transmitted their parameters to the central server, where the selected federated optimization algorithm was applied to compute the global update.

All algorithms were evaluated under identical hyperparameter and training conditions to ensure a fair comparison. The metrics recorded throughout training included test accuracy, training loss, and test loss.

4 Results

Table 1: Summary of Final Performance Metrics for Federated Optimization Algorithms

Algorithm	**Final Acc (%)**	**Best Acc (%)**	**Train Loss**	**Test Loss**
FedAvg (Baseline)	81.97	82.29	0.3346	0.6094
FedAdam	82.56	82.77	0.2418	0.6034
FedYogi	82.29	82.51	0.2482	0.5954
FedAMSGrad	82.19	82.37	0.2775	0.6013
FedAdamW	82.08	82.64	0.2432	0.5983
ClientAdam + ServerAvg	81.12	82.64	0.2973	0.6655

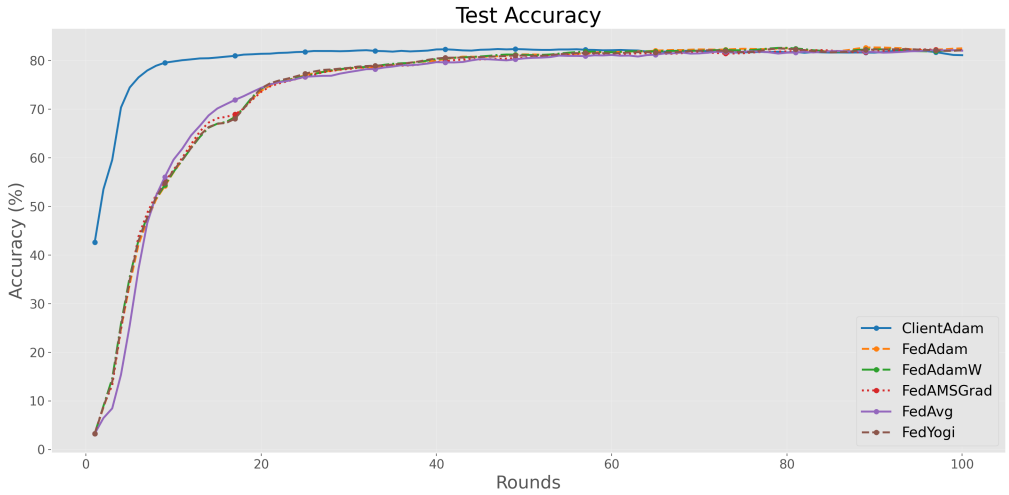

Figure 1: Test Accuracy vs. Communication Rounds.

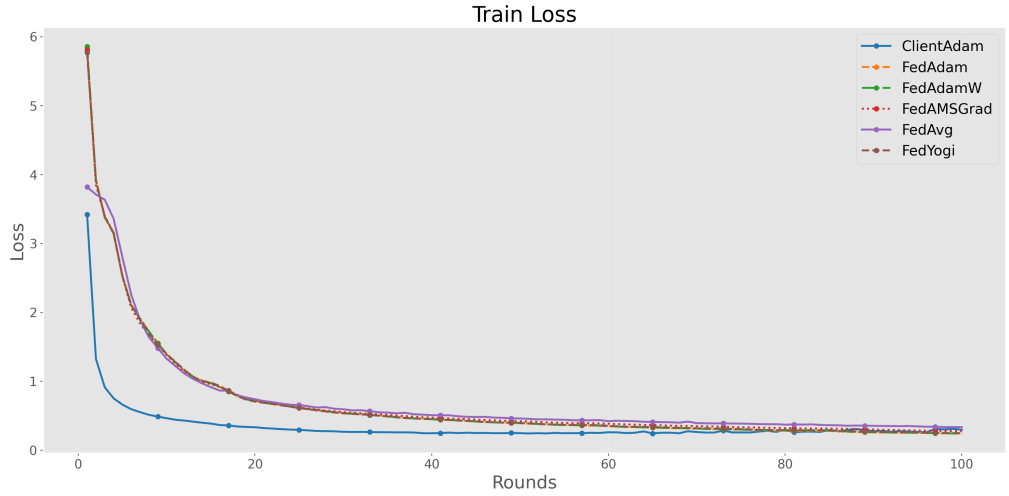

Figure 2: Training Loss vs. Communication Rounds.

Figure 3: Test Loss vs. Communication Rounds.

5 Results Analysis

- The ClientAdam + Server Averaging method exhibited the fastest early improvement in both test accuracy and test loss. This is primarily due to client-side adaptivity, where local Adam optimizers allow each client to make more effective updates during local training.
- Faster initial convergence in this method reduces the number of communication rounds required to achieve competitive performance, making it well-suited for bandwidth-constrained federated learning scenarios.
- Server-side adaptive methods such as FedAdam, FedYogi, FedAMSGrad, and FedAdamW performed similarly to the FedAvg baseline. This suggests that under the moderate heterogeneity induced by the Dirichlet distribution with $\alpha = 0.5$, the advantages of server-side adaptivity are not strongly expressed.
- Although these adaptive optimizers have theoretical advantages in highly non-IID or noisy environments, their curves closely align with FedAvg in this setting, indicating similar learning dynamics.
- All algorithms converged to a similar final accuracy (approximately 82%), with only marginal differences in test loss and training loss. This implies that the choice of optimizer affects convergence speed more than final model quality.
- FedAdam and FedAdamW achieved the lowest final training losses, suggesting more aggressive optimization. However, the generalization gap (test loss) was not significantly smaller than FedAvg, indicating limited gain in generalization performance.
- The ClientAdam + ServerAvg method, while effective early on, showed a slightly higher final test loss, which could suggest mild overfitting or instability in later rounds due to the lack of server-side correction mechanisms.
- FedYogi and FedAMSGrad exhibited stable and smooth convergence patterns, likely due to their conservative second-moment update rules, which are designed to prevent abrupt changes in learning dynamics.
- In practical deployments, ClientAdam + ServerAvg is recommended when rapid early convergence is desired. FedAMSGrad or FedYogi are preferable in settings where stability or theoretical convergence guarantees are essential.
- Despite being the simplest method, FedAvg remains a strong baseline, particularly when the client data distribution is not severely skewed.

6 Conclusion

The combined analysis demonstrates that client-side adaptivity (ClientAdam with Server Averaging) yields the most pronounced practical advantage in terms of convergence speed and early generalization improvements. Server-side adaptive schemes provide stability and theoretical guarantees that may be necessary in highly heterogeneous or adversarial federated environments, but in moderate settings their empirical advantage over FedAvg appears limited. Consequently, designers of federated systems should prioritize local optimization strategies when communication cost is restrictive, and reserve server-side adaptivity for scenarios characterized by severe heterogeneity or where stability guarantees are critical.