Advanced Machine Learning Progress Evaluation Report

Enhancing Federated Averaging (FedAvg) with Local-Global Knowledge Regularization

M.A.S.N. Aththanayake

210055J

Table of Content

1) Summary

The proposed solution is a minimal, client-side enhancement to FedAvg that adds a lightweight knowledge-distillation regularizer during local training. The global model acts as a teacher and anchors each client's predictions, reducing client drift on non-IID data. The method keeps communication unchanged, adds manageable compute per client, and targets fewer rounds and higher non-IID accuracy than FedAvg on baseline test data sets. This report provides background, problem framing, methodology, experimental plan, timeline, risks, resources, and reproducibility controls for the progress evaluation.

2) Background and problem description

Federated learning (FL) trains a single global model across many clients without centralizing raw data. The FedAvg algorithm reduces communication by letting clients perform local SGD for a few epochs before the server averages their models. While simple and effective, FedAvg is fragile under statistical heterogeneity (non-IID data) and partial participation. In label-skew scenarios, each client's local update moves toward its own data distribution; when averaged, these updates can conflict, causing slow convergence, oscillations, and degraded global accuracy. Larger local epochs exacerbate this drift, and uniform aggregation can overweight noisy or idiosyncratic clients. Practitioners need fixes that are small, privacy-preserving, and do not alter the communication protocol especially in constrained settings where added server logic or extra messages are infeasible. The problem, therefore, is to improve FedAvg's stability and accuracy on non-IID data with a change that is (i) manageable in code and overhead, (ii) robust across datasets and client participation rates, and (iii) demonstrably beneficial within the course timeline.

3) Goals and research questions

The objective is to strengthen FedAvg on heterogeneous, non-IID data while preserving its simplicity. In practice, that means reaching strong accuracy with fewer communication rounds, keeping the wire protocol unchanged, and adding a well-justified amount of client computation.

Goals

1. Reduce rounds-to-target accuracy on non-IID data without increasing communication.
2. Improve final global accuracy and early-round stability relative to FedAvg.
3. Keep extra client overhead negligible and make zero changes to the protocol/server

Research questions

1. RQ1: Does guiding each client with the global model's predictions reduce harmful drift compared to FedAvg under label-skew?
2. RQ2: What are effective values for regularization weight (λ) and temperature (T) across datasets and participation rates?
3. RQ3: How does the method behave under varying local epoch counts and fractional participation?
4. RQ4: Can confidence-based masking and λ warm-up further improve stability and accuracy?

4) Methodology Overview

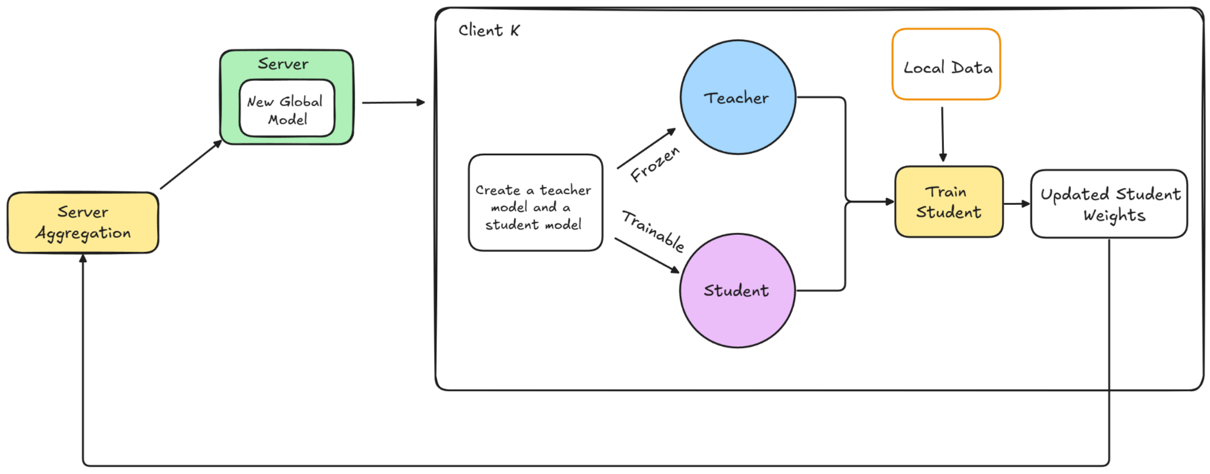

Figure 1 : High level Overview of Proposed Methodology

This methodology keeps FedAvg exactly as-is on the server. On each client it clones the received global model into a frozen "teacher" and a trainable "student," then trains the student with a loss that balances fitting the true labels and staying close to the teacher's probabilities (scaled by lambda and temperature T). Optionally it can use a confidence threshold for the teacher term and a short warm-up for lambda in early rounds. Communication and privacy stay unchanged; the only extra cost is one teacher forward pass per batch, which helps reduce client drift and speeds convergence on non-IID data.

5) Methodology design

The design is centered on making incremental improvements to FedAvg algorithm while directly addressing the issue of client drift under non-IID data. The server remains unchanged, so the communication rounds and aggregation logic are identical to baseline FedAvg. All adjustments are confined to the client's local training loop.

Server role

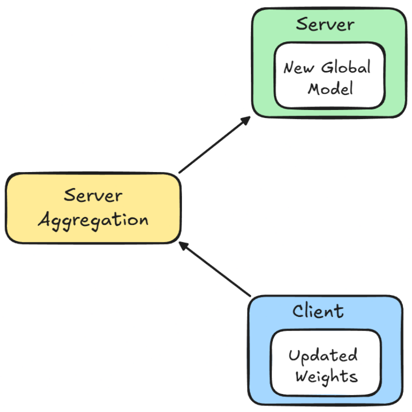

Figure 2 : Server Aggregation of updated weights

At each round, the server sends the current global model to a subset of participating clients. After receiving updates, it performs weighted averaging of client weights (standard FedAvg aggregation).

Client role:

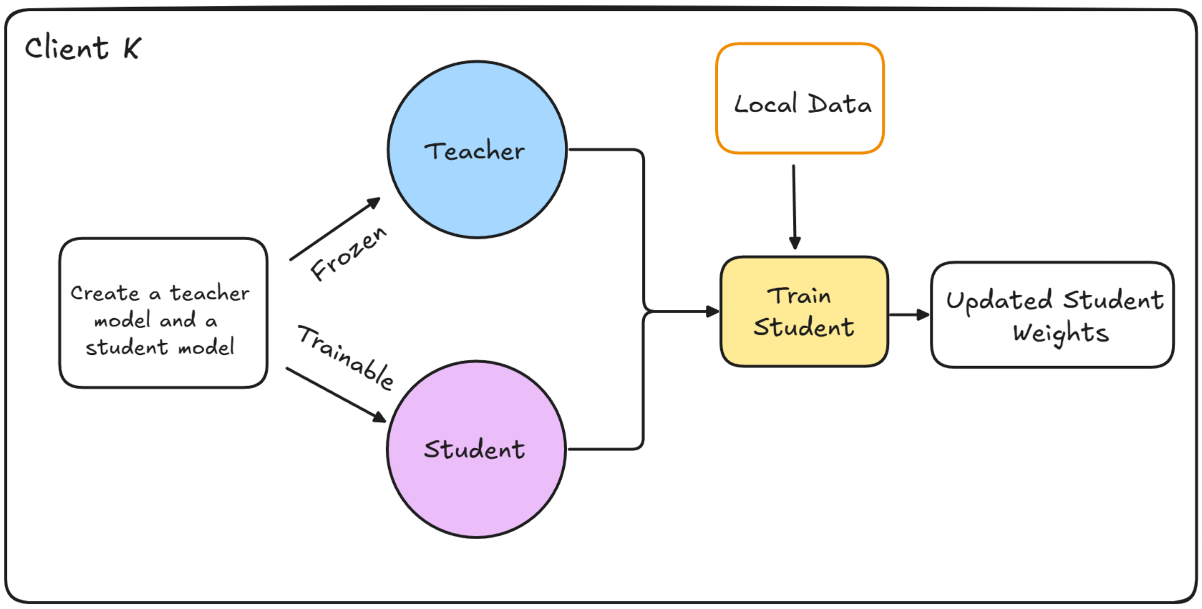

Figure 3 : Client Workflow of proposed solution

When a client receives the global model, it duplicates it into two copies:

- A **teacher model**: kept frozen, used only for inference.
- A **student model**: identical at the start, but trainable.

The client then trains the student using its local data. For each mini-batch, the following steps occur

- The teacher produces probabilities (soft targets) on the input batch.
- The student also produces probabilities.

Two loss components are computed

1. Cross-Entropy with the ground truth labels.
2. Knowledge Distillation loss: KL-divergence between teacher and student predictions, scaled by λ and temperature T^2.

The final loss is a weighted combination of Cross-Entropy and Knowledge Distillation. After training, the updated student weights are returned to the server.

Other refinements

- Confidence threshold (τ): If the teacher's maximum predicted probability is below τ, the KD loss is skipped for that sample, ensuring unreliable teacher outputs do not mislead the student.

- Warm-up for λ: Instead of applying full regularization from the beginning, λ can start small and gradually increase during early rounds, preventing over-constraining when models are still untrained.

This design ensures that client updates are better aligned with the global model while maintaining the lightweight and communication-efficient nature of FedAvg.

6) Experimental plan

The experimental plan is structured to systematically compare the enhanced method against baseline FedAvg in controlled conditions. The evaluation emphasizes whether the knowledge-distillation regularizer reduces client drift and improves convergence on non-IID data.

6.1 Datasets

The following benchmark datasets are selected, each reflecting different data characteristics that stress test federated learning:

- **CIFAR-10:**A standard image classification dataset with 10 balanced classes. For evaluation, it can be partitioned into IID splits (uniform random distribution across clients) and non-IID splits.
- **FEMNIST:**A character recognition dataset with naturally heterogeneous data distribution. Each client corresponds to a different writer, meaning the label and style distributions vary significantly across clients.
- **Shakespeare:**A text dataset derived from the works of Shakespeare, formatted as next-character prediction tasks. Clients are defined by speaking roles, producing highly skewed distributions.

These three datasets together give coverage across image, handwriting, and text domains, allowing observation of whether the improvement generalizes across modalities.

6.2 Setup

- The same server-side and communication logic is preserved between baseline and enhanced versions.
- Clients differ only in whether they use the knowledge regularization during training.

6.3 Evaluation criteria

Accuracy over rounds	Track how quickly models improve and the stability of their progression.
Rounds-to-target accuracy	Compare the number of communication rounds required to reach a set accuracy level.
Final accuracy	Measure overall model quality at the end of training.
Stability	Observe whether client drift is reduced, especially on non-IID splits.

6.4 Hyperparameter considerations

The role of regularization is controlled by λ (weight), T (temperature), and τ (confidence threshold). To test their effects, a grid search will be conducted:

- $\lambda \in \{0.1, 0.5, 1.0, 2.0\}$
- $T \in \{1, 2, 4\}$
- $\tau \in \{0, 0.7, 0.9\}$

This study ensures that reported improvements are not tied to a single narrow choice of parameters but reflect a range of practical settings.

7) Feasibility

The proposed method is feasible within the course project scope from multiple perspectives:

Implementation Feasibility
The design only requires modifications on the client side of FedAvg. The server, aggregation function, and communication protocol remain identical. This ensures implementation complexity is low since only the client training loop is extended with a teacher-student setup and an extra KD loss term.

Computational Feasibility
The extra cost of computing teacher predictions per batch is negligible compared to training the student model. Both teacher and student share the same architecture, and the teacher is frozen, so backpropagation does not apply. Memory requirements are minimal, as the client only stores two model copies during local training.

Experimental Feasibility
The datasets selected (CIFAR-10, FEMNIST, Shakespeare) are standard benchmarks with well-established preprocessing pipelines. These datasets are small enough to run experiments efficiently on limited hardware resources (single GPU or CPU cluster).

Research Feasibility
The method builds directly on FedAvg, so evaluation results are directly comparable to the widely accepted baseline. Hyperparameter tuning is straightforward and limited to three interpretable parameters (λ, T, τ).

Risk Management:

- If λ is too small, the method reduces to standard FedAvg (safe fallback).
- If λ is too large, accuracy may drop due to over-regularization. This is mitigated through grid search and λ warm-up.
- If teacher predictions are noisy in early rounds, confidence thresholding reduces the risk of propagating poor guidance.

Overall, the approach is lightweight, robust, and realistic to complete within the project timeline**.**

8) Timeline

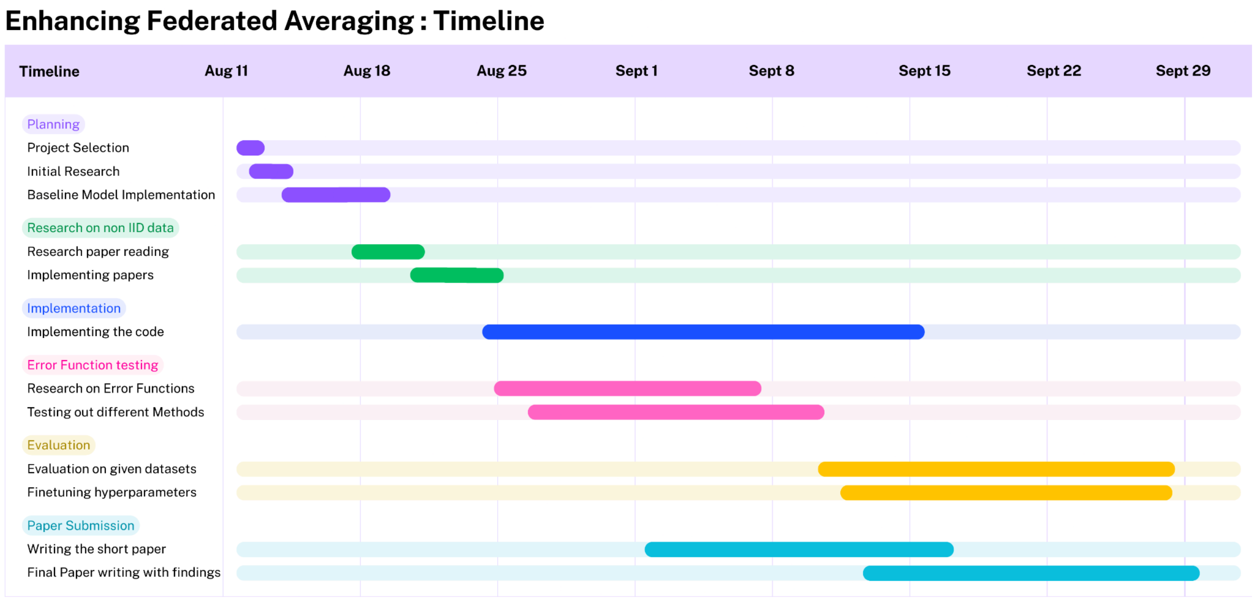

Figure 4 : Timeline

9) References

1. H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data." *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

2. Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated Learning: Strategies for Improving Communication Efficiency." *arXiv preprint arXiv:1610.05492*, 2016.

3. Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. "Federated Multi-Task Learning." *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

4. Qinbin Li, Bingsheng He, and Dawn Song. "Model-Contrastive Federated Learning." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

5. Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning." *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

6. Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions." *IEEE Signal Processing Magazine*, 37(3):50-60, 2020.

7. Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network." *arXiv preprint arXiv:1503.02531*, 2015.

8. Yoon Kim and Alexander M. Rush. "Sequence-Level Knowledge Distillation." *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

9. Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. "Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding." *arXiv preprint arXiv:1904.09482*, 2019.

10. Qinbin Li, Zeyi Wen, and Bingsheng He. "Practical Federated Gradient Boosting Decision Trees." *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

10) Appendices

10.1 Pseudo-code for Client Update

```python
def client_update(global_model, local_data, lambda_, T, tau):
    # Step 1: Duplicate the model
    teacher = clone(global_model).freeze()
    student = clone(global_model).trainable()
    optimizer = SGD(student.parameters(), lr=lr)

    # Step 2: Train with hybrid loss
    for x, y in local_data:
        teacher_out = softmax(teacher(x) / T)
        student_out = softmax(student(x) / T)

        ce_loss = cross_entropy(y, student_out)
        kd_loss = kl_divergence(teacher_out, student_out) * (T**2)

        if max(teacher_out) >= tau:
            loss = ce_loss + lambda_ * kd_loss
        else:
            loss = ce_loss

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    # Step 3: Return updated weights
    return student.weights()
```

10.2 Risk Assessment

- Hyperparameter sensitivity**:** Results depend on λ, T, and τ. Addressed by systematic grid search.
- Over-regularization**:** If λ is too large, students may ignore local labels. Mitigated by warm-up strategy.
- Under-regularization**:** If λ is too small, effect reduces to FedAvg. Still a valid baseline comparison.

10.3 Resource Requirements

- Hardware**:** Single GPU or CPU cluster sufficient for datasets used.
- Software**:** Standard ML frameworks (e.g., PyTorch or TensorFlow).
- Data**:** Publicly available datasets (CIFAR-10, FEMNIST, Shakespeare).