Federated Machine Unlearning

Rem Yang[1], Supriyo Chakraborty[2], Parijat Dube[2]

[1] Department of Computer Science, Grainger College of Engineering, University of Illinois at Urbana-Champaign; [2] Thomas J. Watson Research Center, IBM Research

INTRODUCTION

- Powerful machine learning (ML) models require learning from huge datasets like ImageNet [1] (over 14 million images)
- Raises important questions in privacy: how do we securely *aggregate* and *remove* data in ML?

Machine Unlearning

- Aims to remove a model's knowledge of user data (which it has previously learned)
- Emerged in response to privacy-protecting legislation like GDPR's "right to be forgotten" [2]
- Main existing approach (SISA) [3] partitions a centralized dataset to train a model ensemble
  - Each model only sees a subset of training data
  - To unlearn, only the subset of models with leaving clients need to be retrained
  - Assumes access to all data and iid partitions, and thus does not work in a federated setting

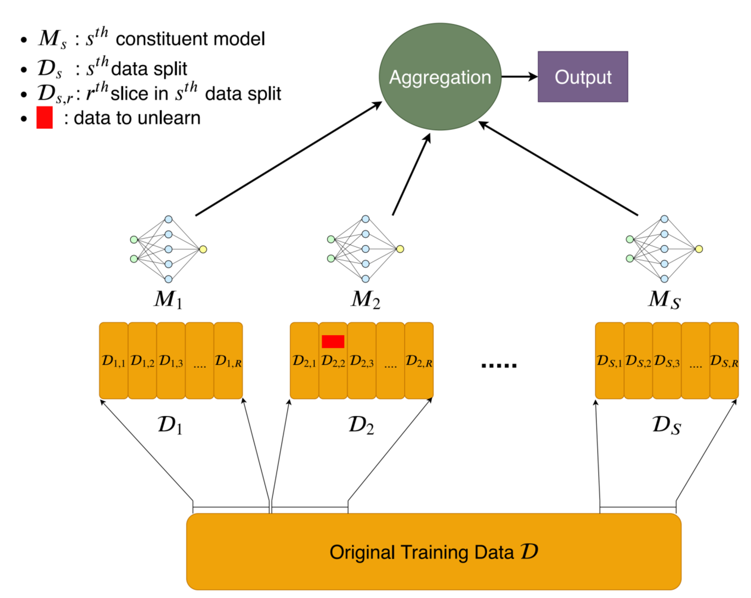

Figure 1. *SISA training strategy [3]. A centralized dataset is divided into disjoint parts $\{D_i\}$, and a collection of models $\{M_i\}$ are independently trained, each on only its own partition of data (i.e., a model M_i sees only data D_i). Each model's independent output is aggregated by majority voting to obtain a final output.*

Federated Learning

- Aims to decentralize model training
  - Clients compute local model updates on their own devices (using only their own data)
  - Central server in the cloud aggregates clients' models into a shared model
    - e.g., by averaging parameters across all client models [4]
- Significantly increases each user's data privacy, as no private data is shared
- Central challenge: each client's data can be significantly different (i.e., non-iid)
  - e.g., a client may only have images for a small subset of the classes of objects that we would like to classify

RESEARCH QUESTION

How do we efficiently train a machine learning model in a **federated setting** to be both **accurate** and a **fast unlearner**?

FEDERATED UNLEARNING

Model Structure

- Key idea: aggregate only a *subset* of client models at a time, and in *several stages*
- Utilize a tree to define aggregation structure
  - Divide N clients into L groups of size B
  - Initialize L leaf nodes; each leaf node only performs federation over the B clients within its group
  - Build a tree with height H that pairwise combines leaf nodes until we get to the root (level 0); idea is, at each decreasing level, federation occurs over a progressively increasing number of clients (see Training)

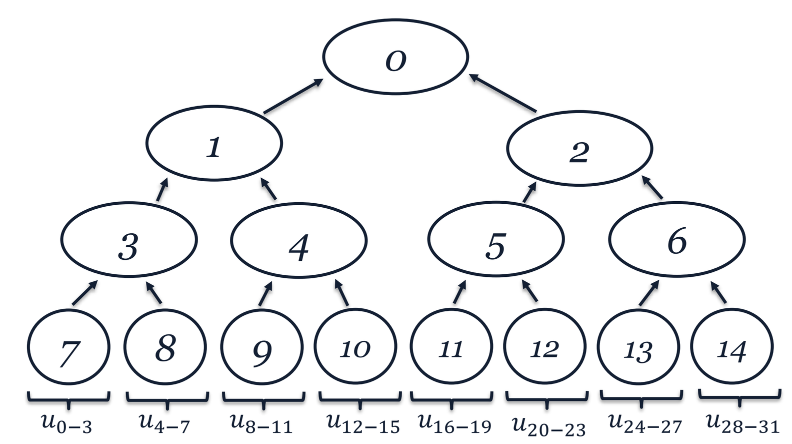

Figure 2. *Example of an aggregation tree where $N = 32$, $B = 4$, and $L = 8$. Each node represents a model that would be aggregated and stored at the server. u_{a-b} denotes that a leaf node contains clients a to b in its group.*

Training

1. Create initial model weights at server and distribute to all clients
2. At the leaf level (e.g., nodes 7-14 in Fig. 2), for each node, clients *within the node's group* perform E epochs of local training, which are then aggregated at the node (i.e., server); this is repeated for R communication rounds
3. At the subsequent level above (e.g., nodes 3-6 in Fig. 2), for each node, first aggregate the *already-trained* models of its children; afterwards, clients belonging to *groups within the node's subtree* perform federation (i.e., R rounds of local training and aggregation)
4. The process is repeated until we reach the root node; this node represents our final model, which incorporates knowledge from *all training data*

Unlearning

When a set of clients $\{u_i\}$ want their data unlearned:

1. For each u_i, identify the leaf node L_i to which it belongs; these represent the leaf nodes that need to be retrained (as they previously incorporated data from the leaving clients)
2. For each L_i, find the path from L_i to the root node; the union of nodes along *all* paths is the total set of models $\{T_i\}$ that need to be retrained
3. Sort $\{T_i\}$ in descending order according to the nodes' levels and initialize random model weights for the leaf-level nodes, then train each T_i in order (according to the procedure above)

METHODOLOGY

Dataset and Partitioning

- CIFAR-10 [5] image classification dataset
  - Contains 50,000 training + 10,000 test 32x32 color images in 10 object classes (e.g., cars, dogs)
  - To create non-iid conditions, we randomly split the train set into N disjoint parts (one for every client), where each contains data on only C classes

Neural Network Architecture

- Medium-sized convolutional neural network (CNN) with 3 convolutional + 2 fully-connected layers

Federation Setting and Parameters

- $N = 32$ clients and $B = 4$ (as in Fig. 2)
- $E = 10$ local training epochs and $R = 5$ communication rounds per node
- Each client is given 1560 images from C classes (smaller C means higher data non-uniformity); we investigate $C \in \{2, 3, 4\}$
- We use simple parameter averaging as our aggregation method, termed TreeAvg

Unlearning Setting and Parameters

- We consider unlearning 4 clients from two groups, u_{0-3} and u_{4-7} (2 from each group)
  - Representative of the scenario where clients with high probability of leaving are clustered together

Baseline

For *training*, we employ the baseline of traditional federated learning, where all client models are aggregated after each round (using FedAvg [4]); we use the same number of total training rounds (i.e., $(H + 1) \cdot R$) as TreeAvg for a fair comparison. Subsequently, for *unlearning*, the entire model must be retrained from scratch (with the rest of the staying clients). By construction, our unlearning cost is equal to the baseline *in the worst case* (when leaving clients are distributed across all leaf nodes).

Metrics

- Model classification accuracy
- Total rounds (across all clients) to unlearn

RESULTS

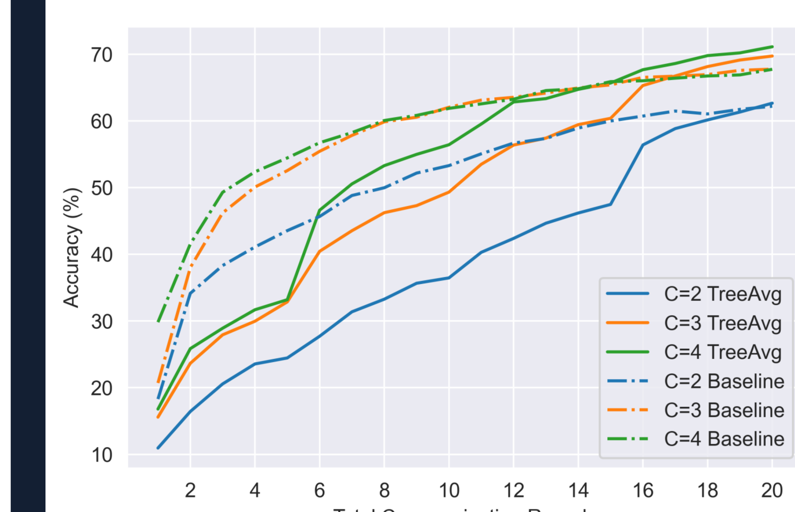

Figure 3. *Evolution of model test accuracy across communication rounds during original training with all 32 clients. For TreeAvg, we report the average accuracy of models 7-14 for rounds 1-5, of models 3-6 for rounds 6-10, of models 1-2 for rounds 11-15, and of model 0 (the final model) for rounds 16-20.*

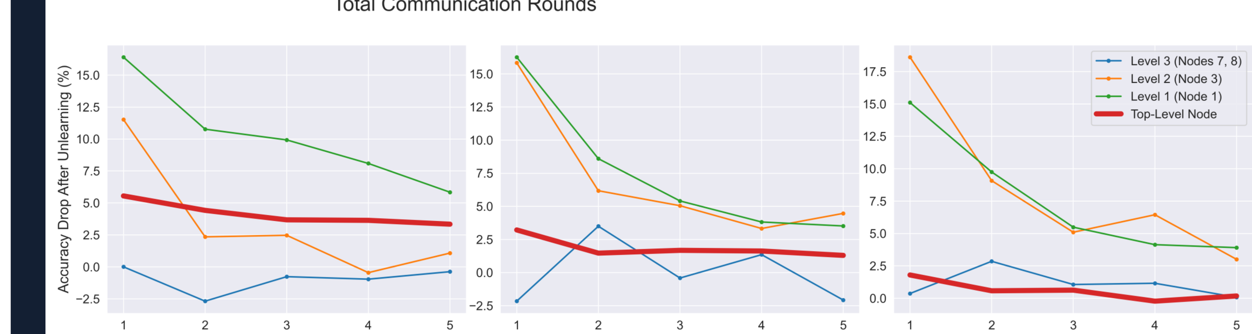

Figure 4. *Accuracy drop of retrained models compared to the original models after 4 clients left during unlearning, for each level of the aggregation tree across communication rounds, on $C = 2$ (left), $C = 3$ (center), and $C = 4$ (right). For Level 3 (the leaf level), we compute the average accuracy drop of models 7 and 8.*

	Original	After Unlearning
C = 2	**62.7** / 62.2	**59.3** / 56.1
C = 3	**69.7** / 67.8	**68.4** / 66.8
C = 4	**71.1** / 67.7	**70.9** / 70.6

Table 1. *Final model test accuracy for both original training and after unlearning, on each of the three non-iid scenarios of $C = 2, 3, 4$. Table entries are denoted (our accuracy) / (baseline accuracy).*

Ours	240
Baseline	560

Table 2. *Total number of communication rounds across all clients to unlearn, for ours and the baseline methods.*

FUTURE WORK

- Explore more federation parameters, e.g., numbers of clients and group sizes
- Consider varying amounts of unlearning requests and client locations + balancing aggregation tree once clients leave

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, pp. 248–255, IEEE, 2009.
[2] S. Shastri, M. Wasserman, and V. Chidambaram, "The seven sins of personal-data processing systems under GDPR," in *HotCloud*, 2019.
[3] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *S&P*, pp. 141–159, IEEE, 2021.
[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, pp. 1273–1282, PMLR, 2017.
[5] A. Krizhevsky, G. Hinton, "Learning multiple layers of features from tiny images," 2009.

ACKNOWLEDGMENTS

I would like to thank my mentors at IBM, Supriyo and Parijat, who provided valuable guidance. I would also like to thank ISUR and C3SR for giving me the opportunity to learn about and contribute to interesting problems in machine learning.

C3SR center for cognitive computing systems research

ILLINOIS