GATEDAFL: PRIVACY-PRESERVING AND RESOURCE-EFFICIENT ASYNCHRONOUS PERSONALIZED FEDERATED LEARNING WITH TRAINABLE THRESHOLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional federated learning (FL) methods typically rely on exchanging model parameters or loss gradients between clients and the server, which incurs substantial communication overhead and exposes sensitive information embedded in model updates. Threshold-gated training offers a more compact alternative, in which a learnable threshold per neuron determines whether its associated parameters are retained or pruned, and only these thresholds need to be communicated. This paper proposes GatedAFL, a threshold-gating method for asynchronous personalized FL. Clients keep their personalized parameters strictly local and upload only the threshold increment accumulated over their local updates. We first prove mathematically that threshold sharing never leaks more information than gradient sharing, and then quantify the privacy gap between the two approaches. Extensive experiments confirm that GatedAFL achieves enhanced resilience against privacy attacks while maintaining competitive accuracy. It reduces communication overhead by two to three orders of magnitude compared to FedAsync, and lowers computational cost by 55.6% relative to SpaFL.

1 INTRODUCTION

Federated learning (FL) is a distributed learning paradigm that enables clients to collaboratively train a shared model by exchanging model updates instead of raw data, thereby preserving data privacy while harnessing decentralized computation McMahan et al. (2017). Despite its promise, deploying FL in resource-constrained environments remains challenging. Three coupled obstacles limit its efficiency and trustworthiness: the substantial communication overhead incurred by repeatedly transmitting model updates, heterogeneity across client data distributions and hardware capabilities, and privacy leakage from the shared updates themselves.

Existing methods address these obstacles along complementary directions. Sparsification and low-rank decomposition reduce transmitted model updates Guastella et al. (2025); Li et al. (2025b). To accommodate non-independent and identically distributed (non-IID) data, personalized FL tailors models to individual clients Shen et al. (2025); Yu et al. (2025), while sparse and adaptive schemes such as FedEL Zhang et al. (2025) reduce the computational and communication burden on heterogeneous clients. Asynchronous methods such as AFLTuning Pang et al. (2026) enable clients to train and communicate without round-level synchronization. To protect privacy, recent approaches employ differential privacy Li et al. (2025a) or encrypted-gradient mechanisms Xue et al. (2025) to reduce information leakage from exchanged updates. Nevertheless, a common limitation persists across these lines of work: clients are required to transmit full or partial model parameters, their loss gradients, or compressed surrogates. For clients with limited bandwidth, large transmissions increase delays and aggravate staleness. Furthermore, privacy-preserving methods often incur additional computation or non-negligible accuracy degradation.

To overcome these limitations, in this work, we propose *GatedAFL*, a privacy-preserving and resource-efficient framework for asynchronous personalized FL. In contrast to conventional FL methods that exchange model parameters or gradients, GatedAFL communicates only trainable thresholds. Each threshold is a scalar associated with a neuron or filter. It determines whether the connected weights are retained or pruned by comparing their average magnitude against the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

threshold value. In this manner, every client keeps its personalized model parameters strictly local and exchanges only compact thresholds with the server. This substantially reduces communication overhead while limiting the model information exposed throughout training. The main contributions of this paper are summarized as follows:

- We propose GatedAFL, the first threshold gating framework for *asynchronous* personalized FL. *Clients* keep their personalized model parameters strictly local and transmit only the compact threshold increment accumulated over their local steps. *The server* adds buffered increments to the current global threshold, so a stale upload contributes only its increment without overwriting accumulated progress.
- We provide an information-theoretic analysis of GatedAFL and derive a quantitative bound on the privacy gap between sharing thresholds and sharing gradients. Modeling the attacker as a Bayes-optimal reconstruction adversary, we prove that threshold sharing never leaks more information than gradient sharing. We further quantify this privacy gap, showing that it is governed by the dimension difference between the threshold and the gradient, as well as the variance of the shared information. Under an isotropic Gaussian model, the retained information fraction depends only on the threshold map's rank relative to the gradient dimension.
- Through experiments on CIFAR-10, CIFAR-100, and Fashion-MNIST (FMNIST), we demonstrate that GatedAFL (i) substantially strengthens resistance to gradient-inversion attacks, reducing reconstruction peak signal-to-noise ratio (PSNR) by 10.7 dB on average relative to gradient sharing, (ii) reduces communication cost by 99.88% on average across the three datasets relative to FedAsync Xie et al. (2019) by transmitting only thresholds instead of full model parameters, and (iii) reduces local computation by 42.63% on average relative to FedAsync Xie et al. (2019) through threshold-based neuron pruning, while maintaining competitive predictive accuracy.

2 Related Work

Asynchronous FL eliminates round-level synchronization by allowing clients to train and communicate independently. FedAsync Xie et al. (2019) updates the global model upon each arrival and reweights it by staleness, whereas FedBuff Nguyen et al. (2022) buffers several updates before aggregation to improve scalability. Cached update calibration Wang et al. (2024) corrects asynchronous updates using cached client updates under heterogeneous data, while FedCompass Li et al. (2024) coordinates heterogeneous clients through computing power-aware scheduling and FedQS Li et al. (2025c) adaptively optimizes gradient and model aggregation in semi-asynchronous FL. However, these approaches still transmit model parameters or gradients. In contrast, GatedAFL exchanges only compact threshold increments with the server for aggregation.

Personalized FL addresses non-IID data by tailoring models to individual clients. pFedMe Dinh et al. (2020) and Ditto Li et al. (2021) regularize personalized models toward a global reference. pFedFDA McLaughlin & Su (2024) adapts global feature distributions to client-specific local distributions, while FedP3 Yi et al. (2024) combines personalized learning and pruning under model heterogeneity. *Communication-efficient FL* instead reduces the size of transmitted information. SpaFL Kim et al. (2024) learns structured sparse models through trainable thresholds. SparsyFed Guastella et al. (2025) adaptively updates sparse masks under heterogeneous data and DeComFL Li et al. (2025d) achieves dimension-free communication through zeroth-order optimization. FedSMU Lu et al. (2025) communicates symbolic model updates, whereas discrepancy-aware compression Wan et al. (2026) selects compression units by output discrepancy. GatedAFL differs from these approaches by learning client-specific structures through trainable thresholds and communicating only these threshold increments.

As for *privacy leakage and protection in FL*, keeping raw data local does not guarantee privacy. Gradient-inversion attacks, such as Deep Leakage from Gradients Zhu et al. (2019), Inverting Gradients Geiping et al. (2020), and GradInversion Yin et al. (2021), demonstrate that shared gradients can reconstruct training samples. SPEAR Dimitrov et al. (2024) extends exact gradient reconstruction to batches. FedInverse Wu et al. (2024) further studies information leakage from shared global models. On the protection side, DictPFL Xue et al. (2025) reduces homomorphic-encryption overhead while protecting all transmitted gradients, while the Sketched Gaussian Mechanism Li et al.

(2025a) combines communication compression with client-level differential privacy. GatedAFL is complementary to these approaches, as it reduces the information exposed by transmitting thresholds rather than gradients or parameters.

3 METHODOLOGY

GatedAFL extends trainable-threshold gating to asynchronous personalized FL. It retains the gating concept from synchronous FL Kim et al. (2024) but introduces two key changes: it aggregates incremental threshold updates instead of averaging absolute thresholds, and it eliminates the synchronization requirement. Moreover, prior threshold-based designs Kim et al. (2024) lack formal privacy guarantees, whereas we prove that threshold sharing leaks no more information than gradient sharing at fixed parameters and quantify the privacy gap.

3.1 STRUCTURED PRUNING WITH TRAINABLE THRESHOLDS

We first describe, for a single client, how a threshold induces a structured pruning mask on the local model, and then present the asynchronous protocol used to communicate these thresholds.

There are K clients, indexed by $k = 1, \dots, K$. Client k holds a local dataset $\mathcal{D}_k$, whose cardinality $|\mathcal{D}_k|$ is the number of samples. Each client k maintains an L-layer neural network with model parameters $\{\boldsymbol{W}_k^1, \boldsymbol{W}_k^2, \dots, \boldsymbol{W}_k^L\}$, where $\boldsymbol{W}_k^l \in \mathbb{R}^{n_{\text{out}}^l \times n_{\text{in}}^l}$ denotes the weight matrix of the l-th layer, and n_{in}^l [1], n_{out}^l are the numbers of input and output neurons. Let $w_{k,ij}^l$ denote the (i,j)-th entry of $\boldsymbol{W}_k^l$. Stacking the parameters of all L layers gives the parameter vector $\boldsymbol{w}_k \in \mathbb{R}^d$, where $d = \sum_{l=1}^L n_{\text{out}}^l n_{\text{in}}^l$ is the total number of model parameters.

To perform structured pruning, we introduce a threshold for each output neuron. Specifically, for layer l, let $\boldsymbol{\tau}^l = (\tau_1^l, \dots, \tau_{n_{\text{out}}^l}^l) \in \mathbb{R}^{n_{\text{out}}^l}$ be the threshold vector, where τ_i^l is the threshold for the i-th output neuron. For neuron i in layer l, we compute the average magnitude of its input weights as

$$\nu_{k,i}^l = \frac{1}{n_{\text{in}}^l} \sum_{j=1}^{n_{\text{in}}^l} |w_{k,ij}^l|, \tag{1}$$

where the sum runs over the n_{in}^l input weights of neuron i. For client k, this neuron is retained if $\nu_{k,i}^l \geq \tau_i^l$ and is pruned otherwise. Therefore, the binary mask $p_{k,i}^l$ of neuron i in layer l is $p_{k,i}^l = \Theta(\nu_{k,i}^l - \tau_i^l)$, where $\Theta(\cdot)$ is the unit step function, i.e., $\Theta(x) = 1$ if $x \geq 0$ and $\Theta(x) = 0$ otherwise. Thus, the threshold vectors induce a structured pruning mask $\boldsymbol{p}_k \in \{0,1\}^d$ on the local model. Each per-neuron entry $p_{k,i}^l$ is applied to all n_{in}^l parameters connected to that neuron, broadcasting the mask to dimension d. The threshold vector $\boldsymbol{\tau} = \{\boldsymbol{\tau}^1, \dots, \boldsymbol{\tau}^L\}$ is shared globally across clients, so every client prunes against the same threshold values while retaining its own weights $\boldsymbol{w}_k$. The pruned local model at client k is $\widetilde{\boldsymbol{w}}_k = \boldsymbol{w}_k \odot \boldsymbol{p}_k$, where $\odot$ denotes the Hadamard product.

For a shared threshold vector $\boldsymbol{\tau}$, the clients jointly solve the following problem:

$$\min_{\{\boldsymbol{w}_k\}_{k=1}^K,\, \boldsymbol{\tau}} \frac{1}{K} \sum_{k=1}^K F_k(\boldsymbol{w}_k, \boldsymbol{\tau}), \tag{2a}$$

$$\text{s.t.} \quad F_k(\boldsymbol{w}_k, \boldsymbol{\tau}) = \frac{1}{|\mathcal{D}_k|} \sum_{i=1}^{|\mathcal{D}_k|} f\left(\boldsymbol{w}_k \odot \boldsymbol{p}_k; \boldsymbol{\xi}_i\right), \quad \forall k \in \{1, \dots, K\}, \tag{2b}$$

where $f(\cdot)$ is the loss on a single sample and $\boldsymbol{\xi}_i$ is the i-th sample of $\mathcal{D}_k$. Here the global threshold vector $\boldsymbol{\tau}$ is the only variable set shared across clients, and it is also the only quantity exchanged with the server.

[1] For a convolutional layer, $n_{\text{in}}^l = c_{\text{in}}^l \times k^l \times h^l$, with c_{in}^l input channels and kernel size $k^l \times h^l$.

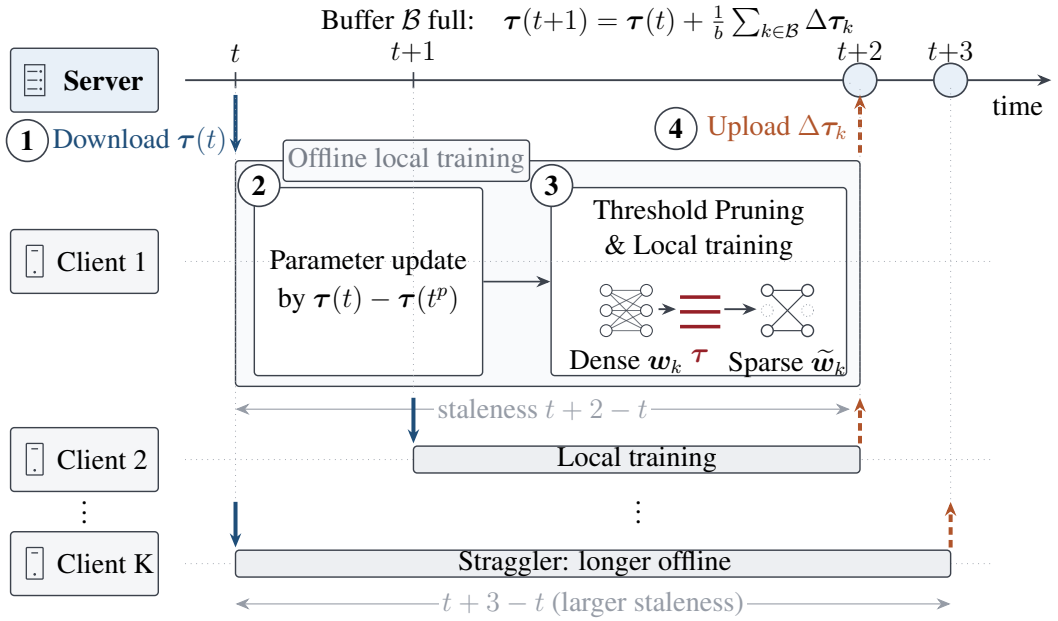


Figure 1: Overview of GatedAFL. On connecting, client k (①) downloads the global threshold $\boldsymbol{\tau}(t)$ and trains offline: it (②) updates its local weights by the threshold change $\boldsymbol{\tau}(t) - \boldsymbol{\tau}(t^p)$ and (③) runs E local SGD steps, during which the thresholds $\boldsymbol{\tau}$ prune the dense model $\boldsymbol{w}_k$ into the sparse model $\widetilde{\boldsymbol{w}}_k$, and finally (④) uploads its increment $\Delta\boldsymbol{\tau}_k$ at the next connection. The server aggregates buffered increments once $\mathcal{B}$ is full. Straggler with longer offline period yields greater staleness $t - t_k^p$.

3.2 The GatedAFL Framework

Figure 1 illustrates the overall GatedAFL framework. Specifically, GatedAFL runs asynchronously, with training organized into rounds indexed by a server version counter $t = 0, 1, 2, \ldots$, and $\boldsymbol{\tau}(t)$ denotes the global threshold of round t. We describe a single client–server interaction in chronological order, which proceeds in four steps:

Step 1 (Threshold download). When client k connects, it downloads from the server the current global threshold $\boldsymbol{\tau}(t)$ and records the associated server version t.

Step 2 (Parameter update). The client then updates its local weights according to the change in the global threshold $\boldsymbol{\tau}(t) - \boldsymbol{\tau}(t^p)$, where t^p denotes the round in which it last connected. The difference $t - t^p$ measures the staleness of the client's local state. For layer l, let $w_{k,ij}^l(t^p)$ denote the locally retained weight from the previous connection round that connects input j to neuron i. Its entrywise update is given by

$$w_{k,ij}^l(t) = w_{k,ij}^l(t^p) + \frac{1}{n_{\text{in}}^l}\left(\tau_i^l(t) - \tau_i^l(t^p)\right) q_{k,i}^l, \tag{3}$$

where the sign-alignment indicator follows SpaFL Kim et al. (2024):

$$q_{k,i}^l = \text{XOR}\left(\text{sign}\left(\textstyle\sum_{j=1}^{n_{\text{in}}^l} w_{k,ij}^l(t^p)\right), \text{sign}\left(\tau_i^l(t) - \tau_i^l(t^p)\right)\right). \tag{4}$$

It determines the update direction from the signs of the summed input weights and the threshold change, so all n_{in}^l weights connected to neuron i share the same update direction.

Step 3 (Local training). The client then trains locally and *offline* for E mini-batch stochastic gradient descent (SGD) steps on its dataset $\mathcal{D}_k$, starting from the updated weights $\boldsymbol{w}_k(t)$ of equation (3) and the downloaded global threshold $\boldsymbol{\tau}(t)$. Let $\boldsymbol{w}_k^*(t)$ and $\boldsymbol{\tau}_k^*(t)$ denote the local weights and thresholds after these E steps. Telescoping the per-step updates gives

$$\boldsymbol{w}_k^*(t) = \boldsymbol{w}_k(t) - \eta\textstyle\sum_{e=0}^{E-1} \boldsymbol{g}_{k,e}, \quad \boldsymbol{\tau}_k^*(t) = \boldsymbol{\tau}(t) - \eta\textstyle\sum_{e=0}^{E-1} \boldsymbol{h}_{k,e} + \rho\eta\textstyle\sum_{e=0}^{E-1}\exp(-\boldsymbol{\tau}_{k,e}), \tag{5}$$

where the stochastic gradients of the local objective F_k in equation (2b) with respect to the parameters and thresholds at step e are defined as

$$\boldsymbol{g}_{k,e} = \nabla_{\widetilde{\boldsymbol{w}}} F_k(\widetilde{\boldsymbol{w}}_{k,e}, \boldsymbol{\tau}_{k,e}; \boldsymbol{\xi}_{k,e}), \quad \boldsymbol{h}_{k,e} = \nabla_{\boldsymbol{\tau}} F_k(\widetilde{\boldsymbol{w}}_{k,e}, \boldsymbol{\tau}_{k,e}; \boldsymbol{\xi}_{k,e}). \tag{6}$$

These gradients use an identity straight-through estimator (STE), replacing the derivative of the unit step by one during backpropagation, and are evaluated on $\widetilde{\boldsymbol{w}}_{k,e}$ with a mini-batch $\boldsymbol{\xi}_{k,e}$ sampled from $\mathcal{D}_k$, and $\boldsymbol{\tau}_{k,e}$ is the local threshold iterate at step $e = 0, \ldots, E-1$, with $\boldsymbol{\tau}_{k,0} = \boldsymbol{\tau}(t)$. The learning rate is $\eta > 0$. The exponential term in equation (5) acts as a sparsity-promoting regularizer with regularization weight $\rho > 0$. Since $\exp(-\tau^l_{k,i})$ decays with $\tau^l_{k,i}$, a smaller threshold receives a larger upward push at each step, so the regularizer encourages thresholds to grow over training.

Step 4 (Threshold upload). After training, the client computes the threshold increment $\Delta\boldsymbol{\tau}_k \triangleq \boldsymbol{\tau}^*_k(t) - \boldsymbol{\tau}(t)$, i.e., the cumulative threshold change between the local threshold after training and the global one downloaded in Step 1, and uploads it at its **next connection**. Since training is offline, the server may have advanced in the meantime. The uploaded increment therefore captures only the client's local progress over its E steps.

At the server, upon receiving a threshold increment $\Delta\boldsymbol{\tau}_k$ from a client k, the server appends it to a buffer $\mathcal{B}$ of fixed capacity b. Here $|\mathcal{B}|$ denotes the current number of buffered increments. If $|\mathcal{B}| < b$, the server retains the increments and waits for further uploads without updating the global threshold. When $|\mathcal{B}| = b$, the server advances the global threshold by adding the averaged increment on top of the current threshold:

$$\boldsymbol{\tau}(t+1) = \boldsymbol{\tau}(t) + \frac{1}{b} \sum_{k \in \mathcal{B}} \Delta\boldsymbol{\tau}_k, \tag{7}$$

and then clears the buffer. Because the server stores and aggregates only thresholds, the transmitted payload is far smaller than in parameter-averaging FL. Recall from Section 3.1 that for each layer l, the threshold vector is $\boldsymbol{\tau}^l \in \mathbb{R}^{n^l_{\text{out}}}$, while the weight matrix is $\boldsymbol{W}^l_k \in \mathbb{R}^{n^l_{\text{out}} \times n^l_{\text{in}}}$. Hence, the communicated dimension is $m = \sum_l n^l_{\text{out}}$ rather than $d = \sum_l n^l_{\text{out}} n^l_{\text{in}}$. The aggregation is also robust to staleness, in contrast to SpaFL Kim et al. (2024), which averages thresholds as $\boldsymbol{\tau}(t+1) = \frac{1}{b} \sum_{k \in \mathcal{B}} \boldsymbol{\tau}_{k,E}$. A stale upload in SpaFL therefore pulls the global threshold back toward an old average. GatedAFL in equation (7) instead adds each increment on top of the *current* global threshold, so accumulated progress is preserved and each upload contributes only through its increment.

In summary, GatedAFL provides three key advantages: communication efficiency, robustness to staleness, and support for heterogeneous or asynchronous clients. We discuss its privacy benefits in the next section.

4 Privacy Analysis of GatedAFL

We first prove that sharing threshold gradients leaks no more private information than sharing full model gradients. Following the Bayesian reconstruction-risk framework (Balunović et al., 2022), we model the strongest possible reconstruction attacker. Given all available observations, it forms a posterior over the private data and outputs the Bayes-optimal guessing distribution. This formulation covers arbitrary reconstruction procedures, including gradient-inversion attacks (Zhu et al., 2019; Geiping et al., 2020; Yin et al., 2021). To quantify leakage, we use the log-loss reconstruction risk (du Pin Calmon & Fawaz, 2012). Its minimum captures the uncertainty that remains for the strongest attacker after observing the shared information. Because the prior uncertainty is the same in the settings we compare, less remaining uncertainty means more information leakage.

Hereafter, we use bold uppercase letters to denote random vectors or matrices, and bold lowercase letters denote vector realizations or fixed vectors. Scalars are nonbold. Distributions use uppercase nonbold letters, and their probability mass or density functions use lowercase letters. Let $\boldsymbol{D}$ be the private data, and let $\boldsymbol{G}$ and $\boldsymbol{Z}$ be the model and threshold gradients produced at the same fixed local parameters $\boldsymbol{w}$, which are not transmitted. Write I for mutual information.

Let $\boldsymbol{Y}$ be the information observed by the attacker, with realization $\boldsymbol{y}$, and let $\boldsymbol{d}$ be a realization of the private data $\boldsymbol{D}$. For this conditional comparison, we give the attacker the same fixed parameters $\boldsymbol{w}$ as side information, without assuming that the protocol transmits them. Given an observation

$\boldsymbol{Y} = \boldsymbol{y}$, the attacker outputs a guessing distribution $S_{\boldsymbol{D}|\boldsymbol{w},\boldsymbol{y}}$ over $\boldsymbol{D}$. Its probability mass or density function $s(\boldsymbol{d} \mid \boldsymbol{w}, \boldsymbol{y})$ is the probability mass or density assigned to the value $\boldsymbol{D} = \boldsymbol{d}$. Following du Pin Calmon & Fawaz (2012), the log-loss reconstruction risk is defined as

$$\mathcal{R}(S) \triangleq \mathbb{E}_{\boldsymbol{D},\boldsymbol{Y}|\boldsymbol{w}}[-\log s(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})]\,. \tag{8}$$

The expectation is taken over $\boldsymbol{D}$ and $\boldsymbol{Y}$ given $\boldsymbol{w}$. Define the optimal risk as $\mathcal{R}^*(\boldsymbol{Y}) \triangleq \min_S \mathcal{R}(S)$. A low risk means that the attacker assigns high probability mass or density to the correct value on average, whereas a high optimal risk means that uncertainty remains.

We aim to show that replacing the model gradient by the threshold gradient cannot reduce the strongest attacker's reconstruction risk, and to identify the information withheld by this replacement. Theorem 1 formalizes this comparison.

Theorem 1 (Privacy gain of threshold sharing)**.** *For gradients produced at the same fixed parameters $\boldsymbol{w}$, suppose $\boldsymbol{Z}$ is a deterministic function of $\boldsymbol{G}$ and the relevant entropy and information quantities are finite. Then the optimal log-loss reconstruction risks satisfy*

$$\begin{aligned} \mathcal{R}^*(\boldsymbol{Z}) - \mathcal{R}^*(\boldsymbol{G}) &= I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) - I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) \\ &= I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{Z}, \boldsymbol{w}) \geq 0. \end{aligned} \tag{9}$$

Hence threshold-gradient sharing leaks no more information about $\boldsymbol{D}$ than model-gradient sharing under this conditional comparison.

Proof. To interpret this risk, we compare the attacker's guess S with the true posterior $P_{\boldsymbol{D}|\boldsymbol{w},\boldsymbol{Y}}$, whose probability mass or density function is $p(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})$. This is the conditional distribution of the private data given the observation. Write H for conditional entropy (differential entropy for continuous data) and D_{KL} for Kullback–Leibler (KL) divergence. Multiplying and dividing the argument of the logarithm in equation (8) by this posterior yields the following equivalent form:

$$\begin{aligned} \mathcal{R}(S) &= \mathbb{E}_{\boldsymbol{D},\boldsymbol{Y}|\boldsymbol{w}}\left[-\log p(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y}) + \log \frac{p(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})}{s(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})}\right] \\ &= \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y}) + \mathbb{E}_{\boldsymbol{Y}|\boldsymbol{w}}\big[D_{\mathrm{KL}}(P_{\boldsymbol{D}|\boldsymbol{w},\boldsymbol{Y}} \| S_{\boldsymbol{D}|\boldsymbol{w},\boldsymbol{Y}})\big]. \end{aligned} \tag{10}$$

The conditional entropy $\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})$ is the expectation of $-\log p(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})$ under the true distribution. Conditioning the log-ratio on $\boldsymbol{Y}$ gives the KL divergence between P and S.

Equation (10) decomposes the risk into conditional entropy and expected KL divergence. Since $D_{\mathrm{KL}} \geq 0$ with equality if and only if $S = P$, the best possible attacker outputs the true posterior, and the minimum risk over all guesses is $\mathcal{R}^*(\boldsymbol{Y}) = \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})$.

We now compare the privacy risk induced by the threshold-gradient observation $\boldsymbol{Z}$ and that induced by the model-gradient observation $\boldsymbol{G}$. Since $\mathcal{R}^*(\boldsymbol{Y}) = \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y})$ holds for any observation $\boldsymbol{Y}$, we evaluate it at $\boldsymbol{Y} = \boldsymbol{Z}$ and $\boldsymbol{Y} = \boldsymbol{G}$. For each case, we rewrite the conditional entropy using $\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}, \boldsymbol{Y}) = \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}) - I(\boldsymbol{D}; \boldsymbol{Y} \mid \boldsymbol{w})$. Subtracting the two expressions cancels the common term $\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w})$, leaving

$$\mathcal{R}^*(\boldsymbol{Z}) - \mathcal{R}^*(\boldsymbol{G}) = I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) - I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}). \tag{11}$$

A smaller $I(\boldsymbol{D}; \boldsymbol{Y} \mid \boldsymbol{w})$ implies a larger optimal reconstruction risk and therefore stronger privacy.

To prove that equation (11) is always nonnegative, we next compare the mutual information that the two observations $\boldsymbol{Z}$ and $\boldsymbol{G}$ carry about the private data $\boldsymbol{D}$. For client k and layer l, under the identity STE of Step 3, SpaFL (Kim et al., 2024) shows that the i-th threshold-gradient entry equals $z^l_{k,i} = -\sum_{j=1}^{n^l_{\mathrm{in}}} \{\boldsymbol{g}_k\}^l_{ij}\, w^l_{k,ij}$, where $\{\boldsymbol{g}_k\}^l_{ij}$ and $z^l_{k,i}$ are the realized model- and threshold-gradient entries, respectively. The coefficients $w^l_{k,ij}$ are fixed local weights and are not transmitted. Stacking these linear combinations across neurons and layers gives

$$\boldsymbol{Z} = \boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{G}, \qquad \boldsymbol{A}_{\boldsymbol{w}} = -\operatorname{blkdiag}\Big([w^1_{k,11}, \ldots, w^1_{k,1n^1_{\mathrm{in}}}], \ldots, [w^L_{k,n^L_{\mathrm{out}}1}, \ldots, w^L_{k,n^L_{\mathrm{out}}n^L_{\mathrm{in}}}]\Big)\,. \tag{12}$$

Each block of $\boldsymbol{A}_{\boldsymbol{w}} \in \mathbb{R}^{m \times d}$ is a neuron's negated input-weight row, so this fixed linear map combines model-gradient entries into one threshold-gradient entry per neuron.[2]

[2]We analyze the single-step threshold gradient $\boldsymbol{Z}$, whereas the protocol transmits the cumulative E-step increment $\Delta\boldsymbol{\tau}_k$. Theorem 3 in Appendix A.3 bounds its leakage by that of the threshold-gradient sequence, given the threshold the server issued.

For any random vectors $\boldsymbol{X}$, $\boldsymbol{Y}_1$, and $\boldsymbol{Y}_2$, the chain rule of mutual information states $I(\boldsymbol{X}; \boldsymbol{Y}_1, \boldsymbol{Y}_2) = I(\boldsymbol{X}; \boldsymbol{Y}_1) + I(\boldsymbol{X}; \boldsymbol{Y}_2 \mid \boldsymbol{Y}_1)$. Applying it to $(\boldsymbol{G}, \boldsymbol{Z})$ in both orders, conditioned on $\boldsymbol{w}$, gives

$$I(\boldsymbol{D}; \boldsymbol{G}, \boldsymbol{Z} \mid \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) + I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{G}, \boldsymbol{w}), \tag{13a}$$

$$I(\boldsymbol{D}; \boldsymbol{G}, \boldsymbol{Z} \mid \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) + I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{Z}, \boldsymbol{w}). \tag{13b}$$

Since $\boldsymbol{A}_{\boldsymbol{w}}$ is fixed by $\boldsymbol{w}$, equation (12) determines $\boldsymbol{Z}$ given $\boldsymbol{G}$ and $\boldsymbol{w}$. Consequently, the conditional mutual information $I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{G}, \boldsymbol{w})$ is zero, and equation (13a) reduces to

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{G}, \boldsymbol{w}) = 0 \implies I(\boldsymbol{D}; \boldsymbol{G}, \boldsymbol{Z} \mid \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}).$$

Equating the resulting equality with the right-hand side of equation (13b) and rearranging yields

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) - I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{Z}, \boldsymbol{w}), \tag{14}$$

Substituting equation (14) into equation (11) gives $\mathcal{R}^*(\boldsymbol{Z}) - \mathcal{R}^*(\boldsymbol{G}) = I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{Z}, \boldsymbol{w}) \geq 0$, proving Theorem 1. Appendix A.1 interprets this gap as residual entropy when the full gradient determines the private data. □

Theorem 1 establishes, without a distributional assumption, that threshold sharing cannot make private-data reconstruction easier under the same fixed model parameters. Thus, threshold sharing provides a strict privacy gain whenever the full gradient carries additional data information beyond the threshold gradient. To quantify its magnitude, we next consider jointly Gaussian data and gradients with isotropic gradient uncertainty. Under these assumptions, the retained fraction of gradient information depends only on the rank of the threshold map relative to the full gradient dimension.

Theorem 2 (Rank-based leakage of threshold sharing)**.** *Conditioned on $\boldsymbol{w}$, suppose $(\boldsymbol{D}, \boldsymbol{G})$ is jointly Gaussian. Let $\boldsymbol{\Sigma}_{\boldsymbol{G}} = \mathrm{Cov}(\boldsymbol{G} \mid \boldsymbol{w})$ and $\boldsymbol{\Sigma}_{\boldsymbol{G}|\boldsymbol{D}} = \mathrm{Cov}(\boldsymbol{G} \mid \boldsymbol{D}, \boldsymbol{w})$ be the marginal and data-conditioned gradient covariance matrices, where Cov denotes conditional covariance. Suppose these matrices are isotropic, $\boldsymbol{\Sigma}_{\boldsymbol{G}} = \sigma_{\boldsymbol{G}}^2 \boldsymbol{I}_d$ and $\boldsymbol{\Sigma}_{\boldsymbol{G}|\boldsymbol{D}} = \sigma_0^2 \boldsymbol{I}_d$, where $\boldsymbol{I}_d$ is the $d \times d$ identity matrix and $\sigma_{\boldsymbol{G}}^2$ and σ_0^2 are the per-coordinate variances satisfying $0 < \sigma_0^2 \leq \sigma_{\boldsymbol{G}}^2$. For $\boldsymbol{Z} = \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{G}$, let $r = \mathrm{rank}(\boldsymbol{A}_{\boldsymbol{w}})$ denote the rank of the threshold map. Then the leaked information satisfies*

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = \frac{r}{d} I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}). \tag{15}$$

When the model gradient has positive information leakage, the threshold gradient therefore retains the fraction r/d of that information.

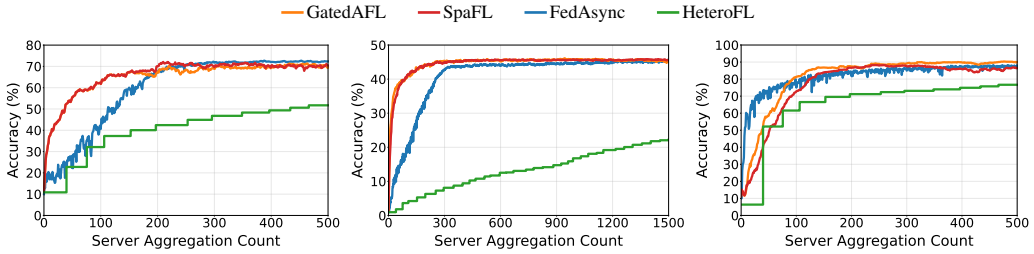
The proof is given in Appendix A.2, together with the general covariance expression. Under the assumptions of Theorem 2, if $\boldsymbol{A}_{\boldsymbol{w}}$ has full row rank m, the retained fraction is m/d. For example, $d = 10^6$ gradient scalars versus $m = 10^3$ independent threshold scalars leave only 10^{-3} of the information revealed by gradient sharing. Equivalently, normalizing gradient leakage to one gives threshold leakage m/d. This quantitative fraction relies on the Gaussian and isotropic assumptions, whereas the non-increase result of Theorem 1 does not.

5 Experiment

5.1 Experiment Setup

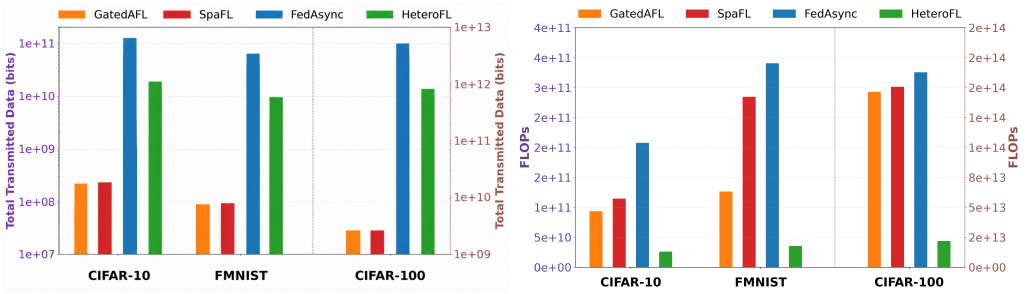
We conduct experiments on three image classification datasets: CIFAR-10, CIFAR-100 Krizhevsky & Hinton (2009), and FMNIST Xiao et al. (2017). For all datasets, $K = 100$ clients train and communicate asynchronously, and the server uses an aggregation buffer of size $b = 10$. The completion delay of each local training task on client k is independently sampled as $d_k \sim 1 + \mathrm{Exp}(1/(\lambda|\mathcal{D}_k|))$, where Exp is parameterized by its rate and the delay-scale parameter is $\lambda = 1$. We generate non-IID client data using a Dirichlet distribution with concentration parameter 0.2 for FMNIST and 0.1 for CIFAR-10 and CIFAR-100. We train LeNet on FMNIST, Vision Transformer (ViT) on CIFAR-10, and ResNet-18 on CIFAR-100, with respective learning rates $\eta = (0.001, 0.01, 0.01)$, local steps $E = (5, 3, 5)$, regularization weights $\rho = (0.0015, 0.0005, 0.0003)$, and batch sizes $(64, 16, 64)$. Sensitivity analyses are provided in Appendix B.

We compare GatedAFL with three baselines: FedAsync Xie et al. (2019) transmits full model parameters and updates the global model upon each upload. HeteroFL Diao et al. (2021) trains client-specific submodels of different widths and aggregates shared parameters. SpaFL Kim et al. (2024) prunes layers with trainable thresholds and averages thresholds rather than increments.



(a) CIFAR-10 (b) CIFAR-100 (c) FMNIST

Figure 2: Test accuracy of GatedAFL against the baseline algorithms on three datasets.



(a) Transmitted data (b) FLOPs

Figure 3: Communication cost (cumulative transmitted bits) and FLOPs of GatedAFL and baselines. Both panels use the left axis for CIFAR-10/FMNIST and the right axis for CIFAR-100.

5.2 Accuracy Evaluation

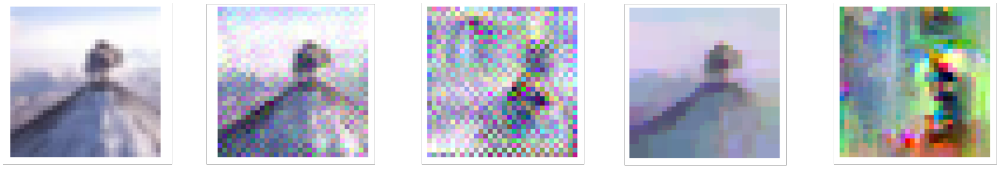
Figure 2 illustrates the test accuracy against the server aggregation count on CIFAR-10, CIFAR-100, and FMNIST. We run SpaFL in the same asynchronous, buffered setting as GatedAFL, with thresholding fixed, to isolate the effect of the aggregation rule.

As shown in Figure 2(a), the two threshold-based methods achieve the fastest convergence on CIFAR-10. GatedAFL and SpaFL reach 69.41% within roughly 100 aggregations, whereas FedAsync catches up only after 231 aggregations. The accuracies of all three methods eventually stabilize at around 70%, while HeteroFL increases stepwise to 51.72% by 500 aggregations. On CIFAR-100 (Figure 2(b)), GatedAFL again exhibits convergence behavior nearly identical to SpaFL. Both reach 45.24% accuracy much earlier than FedAsync, whereas HeteroFL attains only 22.29% even after 1500 aggregations. On FMNIST (Figure 2(c)), FedAsync initially improves fastest. However, GatedAFL surpasses it to attain the highest final accuracy of 89.96%, followed closely by SpaFL, while HeteroFL reaches 76.67%. These results are consistent with the benefits of combining asynchronous training and incremental threshold aggregation. Unlike SpaFL's averaging of thresholds, incremental aggregation preserves global progress under stale updates. Compared with FedAsync, which exchanges model parameters, GatedAFL communicates only compact thresholds, providing more staleness-tolerant structural guidance while keeping personalized weights local. HeteroFL attains lower accuracy, although these aggregation-count curves do not isolate straggler delays.

Figure 3 compares communication and computational costs, with cumulative transmission in Figure 3(a) measured in bits using 32-bit floating-point parameters and thresholds. Since GatedAFL exchanges only compact threshold vectors rather than full model parameters, it uploads 2.27×10^8, 9.28×10^7, and 2.30×10^9 bits on CIFAR-10, FMNIST, and CIFAR-100, two to three orders of magnitude below FedAsync (1.29×10^{11}, 6.89×10^{10}, and 5.36×10^{12} bits), while HeteroFL transmits somewhat less than FedAsync (1.92×10^{10}, 1.03×10^{10}, and 7.59×10^{11} bits) but still far more than GatedAFL, albeit with a significant accuracy penalty. See Appendix C for cumulative communication costs. Figure 3(b) shows the computational advantage of incremental aggregation:

Table 1: Defense performance of threshold sharing versus gradient sharing under two gradient-inversion attacks: inverting gradients (IG) and GradInversion.

Attack	Metric	CIFAR-10		CIFAR-100		FMNIST	
		Threshold	Gradient	Threshold	Gradient	Threshold	Gradient
IG	MSE	0.056	0.007	0.075	0.009	0.159	0.028
	PSNR	13.087	26.083	11.691	20.243	8.209	17.613
	SSIM	0.363	0.853	0.253	0.771	0.088	0.561
GradInversion	MSE	0.058	0.003	0.057	0.008	0.140	0.016
	PSNR	12.883	25.812	12.695	20.641	8.886	21.301
	SSIM	0.364	0.873	0.263	0.750	0.120	0.677



(a) CIFAR-100

Figure 4: Reconstructed images by IG and GradInversion from gradients versus thresholds on CIFAR-100, compared with the original images.

GatedAFL attains lower floating-point operations (FLOPs) than SpaFL, 9.35×10^{10} on CIFAR-10 versus 1.15×10^{11} for SpaFL (an 18.4% reduction) and 1.26×10^{11} on FMNIST versus 2.85×10^{11} (a 55.6% reduction). This is consistent with incremental aggregation avoiding stale thresholds and supporting more aggressive pruning at comparable accuracy. The gain is smaller for ResNet-18 on CIFAR-100 (1.4647×10^{14} versus 1.5060×10^{14}, a 2.7% reduction), indicating that the computational benefit varies across model–dataset pairs.

5.3 Privacy Evaluation

We use Inverting Gradients (IG) Geiping et al. (2020) and GradInversion Yin et al. (2021) to attack models on CIFAR-10, CIFAR-100, and FMNIST. We perform single-sample single-step local training and compare threshold sharing with gradient sharing in terms of reconstruction quality.

Table 1 reports mean squared error (MSE), structural similarity (SSIM), and PSNR. Lower reconstruction quality indicates stronger protection against the evaluated attacks. On CIFAR-100 (Figure 4), both attacks recover recognizable images from gradients but poorly from thresholds. These findings are consistent with Theorem 1, which establishes non-increasing leakage for threshold gradients at fixed parameters, and Theorem 2, which quantifies the gap under isotropic Gaussian assumptions. Overall, GatedAFL reduces communication and computation, improves privacy, and maintains comparable or superior accuracy. See Appendix D for additional reconstructions.

6 Conclusion

We proposed GatedAFL, an asynchronous personalized FL framework sharing only threshold increments, not model parameters or gradients, to reduce overhead and information exposure. Incremental aggregation preserves global threshold progress under stale uploads. We proved that threshold sharing leaks no more than gradient sharing at fixed parameters and quantified the gap under isotropic Gaussian assumptions. Experiments on CIFAR-10, CIFAR-100 and FMNIST showed competitive accuracy with markedly stronger resistance to gradient-inversion attacks. Future work will extend threshold gating to larger models, integrate secure aggregation and differential privacy, and validate deployment on real edge devices.

AI USE STATEMENT

In this work, we used large language models (LLMs) to improve the manuscript's language and readability, assist with proof writing and clarification of assumptions, and refine the interpretation of results. The tools were not used to generate research ideas, develop the proposed algorithm, or conduct experiments. All LLM-assisted revisions were reviewed and verified by the authors, who take responsibility for the final content.

REFERENCES

Mislav Balunović, Dimitar Iliev Dimitrov, Robin Staab, and Martin Vechev. Bayesian framework for gradient leakage. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=f2lrIbGx3x7.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=TNkPBBYFkXg.

Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. SPEAR: Exact gradient inversion of batches in federated learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 106768–106799, 2024. doi: 10.52202/079017-3390. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c13cd7feab4beb1a27981e19e2455916-Abstract-Conference.html.

Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21394–21405, 2020. URL https://papers.nips.cc/paper/2020/hash/f4f1f13c8289ac1b1ee0ff176b56fc60-Abstract.html.

Flávio du Pin Calmon and Nadia Fawaz. Privacy against statistical inference. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1401–1408, 2012. doi: 10.1109/Allerton.2012.6483382.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, pp. 16937–16947, 2020. URL https://papers.nips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.

Adriano Guastella, Lorenzo Sani, Alex Iacob, Alessio Mora, Paolo Bellavista, and Nicholas Donald Lane. SparsyFed: Sparse adaptive federated learning. In *The Thirteenth International Conference on Learning Representations*, volume 2025, pp. 14781–14813, 2025. URL https://openreview.net/forum?id=OBUQNASaWw.

Minsu Kim, Walid Saad, Mérouane Debbah, and Choong Seon Hong. SpaFL: Communication-efficient federated learning with sparse models and low computational overhead. In *Advances in Neural Information Processing Systems*, volume 37, pp. 86500–86527, 2024. doi: 10.52202/079017-2747. URL https://papers.nips.cc/paper_files/paper/2024/hash/9d6d351ba8028a50382f42a065d31bf0-Abstract-Conference.html.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Qiaobo Li, Zhijie Chen, and Arindam Banerjee. Sketched gaussian mechanism for private federated learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 131952–132017, 2025a. doi: 10.52202/085713-4394. URL https://papers.neurips.cc/paper_files/paper/2025/hash/bef4c645dd0754c36c55b75af8c57216-Abstract-Conference.html.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Shijie Xu, Weihong Luo, Yuhua Li, Xiuqiang He, and Ruixuan Li. The panaceas for improving low-rank decomposition in communication-efficient federated learning. In *Proceedings of the 42nd International Conference on Machine*

Learning, volume 267, pp. 35536–35561, 2025b. URL https://proceedings.mlr.press/v267/li25bn.html.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 6357–6368, 2021. URL https://proceedings.mlr.press/v139/li21h.html.

Yunbo Li, Jiaping Gui, Zhihang Deng, Fanchao Meng, and Yue Wu. FedQS: Optimizing gradient and model aggregation for semi-asynchronous federated learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 86502–86550, 2025c. doi: 10.52202/085713-2897. URL https://papers.nips.cc/paper_files/paper/2025/hash/7d3f1540bb7af35813578db81d03b0dd-Abstract-Conference.html.

Zhe Li, Bicheng Ying, Zidong Liu, Chaosheng Dong, and Haibo Yang. Achieving dimension-free communication in federated learning via zeroth-order optimization. In *International Conference on Learning Representations*, volume 2025, pp. 23769–23803, 2025d. URL https://openreview.net/forum?id=omrLHFzC37.

Zilinghan Li, Pranshu Chaturvedi, Shilan He, Han Chen, Gagandeep Singh, Volodymyr Kindratenko, Eliu Huerta, Kibaek Kim, and Ravi Madduri. FedCompass: Efficient cross-silo federated learning on heterogeneous client devices using a computing power-aware scheduler. In *International Conference on Learning Representations*, volume 2024, pp. 39109–39145, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/a9f3457fa97f106f1756885237787789-Abstract-Conference.html.

Xinyi Lu, Hao Zhang, Chenglin Li, Weijia Lu, Zhifei Yang, Wenrui Dai, Xiaodong Zhang, Xiaofeng Ma, Can Zhang, Junni Zou, and Hongkai Xiong. FedSMU: Communication-efficient and generalization-enhanced federated learning through symbolic model updates. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 40903–40929, 2025. URL https://proceedings.mlr.press/v267/lu25u.html.

Connor J. McLaughlin and Lili Su. Personalized federated learning via feature distribution adaptation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 77038–77059, 2024. doi: 10.52202/079017-2451. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8ce6c5450ccddbe6adee4b3749893587-Abstract-Conference.html.

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*, volume 54, pp. 1273–1282, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

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*, volume 151, pp. 3581–3607, 2022. URL https://proceedings.mlr.press/v151/nguyen22b.html.

Xiaoyi Pang, Yajie Zhou, Zhibo Wang, Xuanyi Hao, Tao Han, and Song Guo. Unlocking the potential of asynchronous federated learning in non-IID edge environments. *IEEE Transactions on Mobile Computing*, 25(10):15897–15908, 2026. doi: 10.1109/TMC.2026.3683093. URL https://doi.org/10.1109/TMC.2026.3683093.

Lei Shen, Zhenheng Tang, Lijun Wu, Yonggang Zhang, Xiaowen Chu, Tao Qin, and Bo Han. Hotpluggable federated learning: Bridging general and personalized FL via dynamic selection. In *International Conference on Learning Representations*, volume 2025, pp. 102648–102684, 2025. URL https://openreview.net/forum?id=B8akWa62Da.

Zhiyi Wan, Yijia Chi, Liang Li, Wanrou Du, Miao Pan, and Xiaoqi Qin. Enhancing communication compression via discrepancy-aware calibration for federated learning. In *International Conference on Learning Representations*, volume 2026, pp. 69989–70009, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/72235260ae8d57ac42638a26d3b7d089-Abstract-Conference.html.

Yujia Wang, Yuanpu Cao, Jingcheng Wu, Ruoyu Chen, and Jinghui Chen. Tackling the data heterogeneity in asynchronous federated learning with cached update calibration. In *International Conference on Learning Representations*, volume 2024, pp. 16160–16192, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/464c70add1e8119c4301de4e1dde17b4-Abstract-Conference.html.

Di Wu, Jun Bai, Yiliao Song, Junjun Chen, Wei Zhou, Yong Xiang, and Atul Sajjanhar. FedInverse: Evaluating privacy leakage in federated learning. In *International Conference on Learning Representations*, volume 2024, pp. 44526–44556, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c225136cfe52a8fd66658bbcf9d894ab-Abstract-Conference.html.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. URL https://arxiv.org/abs/1708.07747.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019. URL https://arxiv.org/abs/1903.03934.

Jiaqi Xue, Mayank Kumar, Yuzhang Shang, Shangqian Gao, Rui Ning, Mengxin Zheng, Xiaoqian Jiang, and Qian Lou. DictPFL: Efficient and private federated learning on encrypted gradients. In *Advances in Neural Information Processing Systems*, volume 38, pp. 119806–119829, 2025. doi: 10.52202/085713-4001. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/adb02256990eb4369268d554fbfc287f-Abstract-Conference.html.

Kai Yi, Nidham Gazagnadou, Peter Richtarik, and Lingjuan Lyu. FedP3: Federated personalized and privacy-friendly network pruning under model heterogeneity. In *International Conference on Learning Representations*, volume 2024, pp. 46587–46619, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cbc4ab80cd77aa0eb87da062fbcddb46-Abstract-Conference.html.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yin_See_Through_Gradients_Image_Batch_Recovery_via_GradInversion_CVPR_2021_paper.html.

Xin Yu, Zelin He, Ying Sun, Lingzhou Xue, and Runze Li. Understanding the statistical accuracy-communication trade-off in personalized federated learning with minimax guarantees. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 73009–73050, 2025. URL https://proceedings.mlr.press/v267/yu25h.html.

Letian Zhang, Bo Chen, Jieming Bian, Lei Wang, and Jie Xu. FedEL: Federated elastic learning for heterogeneous devices. In *Advances in Neural Information Processing Systems*, volume 38, pp. 35665–35696, 2025. doi: 10.52202/085713-1199. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3327377ee37b8c87c43a7d6a4ea3d279-Abstract-Conference.html.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, pp. 14774–14784, 2019. URL https://papers.nips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html.

A Supplementary Privacy Analysis and Proofs

A.1 Quantifying the privacy gap via residual entropy

For discrete private data with $0 < \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w}) < \infty$, we assume that the attacker, given the full model gradient $\boldsymbol{G}$, can uniquely determine the private data[3] $\boldsymbol{D}$, i.e., $\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{G}, \boldsymbol{w}) = 0$. Under this assumption, the lost-information term in equation 14 reduces to

$$\begin{aligned} I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{Z}, \boldsymbol{w}) &= \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{Z}, \boldsymbol{w}) - \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{G}, \boldsymbol{Z}, \boldsymbol{w}) \\ &= \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{Z}, \boldsymbol{w}). \end{aligned} \tag{16}$$

Substituting equation 16 into equation 14 yields

$$I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) - I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{Z}, \boldsymbol{w}), \tag{17}$$

which states that the privacy advantage of uploading $\boldsymbol{Z}$ instead of $\boldsymbol{G}$ equals the residual uncertainty about $\boldsymbol{D}$ after observing $\boldsymbol{Z}$. The assumption also gives $I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) = \mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w})$. We normalize by $\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w})$, so that every leakage value is expressed as a fraction of what gradient sharing reveals. Under this normalization, equation (17) becomes

$$\frac{I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w})}{\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w})} = 1 - \frac{\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{Z}, \boldsymbol{w})}{\mathsf{H}(\boldsymbol{D} \mid \boldsymbol{w})}. \tag{18}$$

A.2 Proof of Theorem 2

Proof. For jointly Gaussian $(\boldsymbol{D}, \boldsymbol{G})$, let $\boldsymbol{\Sigma}_{\boldsymbol{Z}} = \mathrm{Cov}(\boldsymbol{Z} \mid \boldsymbol{w})$ and $\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{D}} = \mathrm{Cov}(\boldsymbol{Z} \mid \boldsymbol{D}, \boldsymbol{w})$. When the marginal and data-conditioned gradient covariance matrices are positive definite, the two threshold covariance matrices have the same rank and support. Their conditional mutual information is therefore

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = \frac{1}{2} \log \frac{\mathrm{pdet}(\boldsymbol{\Sigma}_{\boldsymbol{Z}})}{\mathrm{pdet}(\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{D}})}, \tag{19}$$

where pdet is the pseudo-determinant, the product of the nonzero eigenvalues. Since $\boldsymbol{Z} = \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{G}$, the covariances become

$$\boldsymbol{\Sigma}_{\boldsymbol{Z}} = \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{\Sigma}_{\boldsymbol{G}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{D}} = \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{\Sigma}_{\boldsymbol{G}|\boldsymbol{D}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}. \tag{20}$$

Substituting equation 20 into equation 19 yields the general covariance expression

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = \frac{1}{2} \log \frac{\mathrm{pdet}\big(\boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{\Sigma}_{\boldsymbol{G}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}\big)}{\mathrm{pdet}\big(\boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{\Sigma}_{\boldsymbol{G}|\boldsymbol{D}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}\big)}. \tag{21}$$

Under the isotropy assumptions of Theorem 2, both covariance matrices are multiples of $\boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}$. If $r = 0$, $\boldsymbol{Z}$ is constant and equation (15) holds with zero threshold leakage. Otherwise, let $\lambda_1, \ldots, \lambda_r$ be the nonzero eigenvalues of this matrix. For any scalar $c > 0$, $\mathrm{pdet}(c\, \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{A}_{\boldsymbol{w}}^{\mathsf{T}}) = c^r \prod_{i=1}^{r} \lambda_i$. Hence equation (21) becomes

$$I(\boldsymbol{D}; \boldsymbol{Z} \mid \boldsymbol{w}) = \frac{1}{2} \log \frac{(\sigma_{\boldsymbol{G}}^2)^r \prod_{i=1}^{r} \lambda_i}{(\sigma_0^2)^r \prod_{i=1}^{r} \lambda_i} = \frac{r}{2} \log \frac{\sigma_{\boldsymbol{G}}^2}{\sigma_0^2}. \tag{22}$$

Applying the same Gaussian formula to the full gradient gives

$$I(\boldsymbol{D}; \boldsymbol{G} \mid \boldsymbol{w}) = \frac{d}{2} \log \frac{\sigma_{\boldsymbol{G}}^2}{\sigma_0^2}. \tag{23}$$

Multiplying both sides of equation 23 by r/d and using equation 22 yields equation (15), including the zero-leakage case $\sigma_{\boldsymbol{G}}^2 = \sigma_0^2$, which completes the proof. □

[3] To intuitively analyze the privacy of threshold sharing relative to gradient sharing, we adopt the conservative assumption that gradients already leak all data information. This special case is not required by Theorem 1.

A.3 Information-Theoretic Analysis of the Threshold Increment

In this appendix, we show that the theoretical analysis carries over to the quantity actually communicated by the GatedAFL protocol, namely the cumulative threshold increment $\Delta\boldsymbol{\tau}_k \triangleq \boldsymbol{\tau}_k^*(t) - \boldsymbol{\tau}(t)$ of Step 4.

Theorem 3 (Increment leakage upper bound)**.** *Let $\boldsymbol{h}_{k,e} = \nabla_{\boldsymbol{\tau}} F_k(\widetilde{\boldsymbol{w}}_{k,e}, \boldsymbol{\tau}_{k,e}; \boldsymbol{\xi}_{k,e})$ be the threshold stochastic gradient at local step e, and let $\Delta\boldsymbol{\tau}_k = \boldsymbol{\tau}_{k,E} - \boldsymbol{\tau}_{k,0}$ be the threshold increment defined in Step 4, where $\boldsymbol{\tau}_{k,0} = \boldsymbol{\tau}(t)$ is the downloaded global threshold and $\boldsymbol{\tau}_{k,E} = \boldsymbol{\tau}_k^*(t)$ is the local threshold after the E steps of Step 3. Conditioned on the downloaded threshold $\boldsymbol{\tau}_{k,0}$ and the model parameters $\boldsymbol{w}$, the transmitted threshold increment $\Delta\boldsymbol{\tau}_k$ leaks no more information about the private data $\boldsymbol{D}$ than the sequence of per-step threshold gradients $\{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1}$. Moreover, its leakage is exactly the same as that of the final local threshold $\boldsymbol{\tau}_{k,E}$. Formally,*

$$I(\boldsymbol{D}; \Delta\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{\tau}_{k,E} \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}) \le I(\boldsymbol{D}; \{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1} \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}). \tag{24}$$

Here $I(\boldsymbol{D}; \Delta\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w})$ is the amount of information that the transmitted increment leaks about the private data $\boldsymbol{D}$ given $\boldsymbol{\tau}_{k,0}$ and $\boldsymbol{w}$.

Proof. All quantities are conditioned on the downloaded threshold $\boldsymbol{\tau}_{k,0}$, which the server issues in Step 1 and is therefore known to it. We use the data-processing inequality (DPI): if $X \to Y \to V$ forms a Markov chain, then $I(X;V) \le I(X;Y)$, so processing Y by any (possibly randomized) function cannot increase the information it carries about X.

The proof proceeds in two steps. First, we show that, given $\boldsymbol{\tau}_{k,0}$, the transmitted increment $\Delta\boldsymbol{\tau}_k$ and the final local threshold $\boldsymbol{\tau}_{k,E}$ are informationally equivalent. Second, we show that $\Delta\boldsymbol{\tau}_k$ leaks no more than the sequence of per-step threshold gradients $\{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1}$ that produces it. Both steps rely on the DPI.

Equality. Given $\boldsymbol{\tau}_{k,0}$, the map $\boldsymbol{\tau}_{k,E} \mapsto \boldsymbol{\tau}_{k,E} - \boldsymbol{\tau}_{k,0} = \Delta\boldsymbol{\tau}_k$ is a bijection, with inverse $\Delta\boldsymbol{\tau}_k \mapsto \boldsymbol{\tau}_{k,0} + \Delta\boldsymbol{\tau}_k$. Applying the DPI in both directions gives the two opposite inequalities, hence $I(\boldsymbol{D}; \Delta\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}) = I(\boldsymbol{D}; \boldsymbol{\tau}_{k,E} \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w})$: once the starting threshold is fixed, the increment and the final threshold are two encodings of the same information.

Inequality. Telescoping the local threshold recursion $\boldsymbol{\tau}_{k,e+1} = \boldsymbol{\tau}_{k,e} - \eta\boldsymbol{h}_{k,e} + \rho\eta\exp(-\boldsymbol{\tau}_{k,e})$ of Step 3 over $e = 0, \dots, E-1$ gives

$$\Delta\boldsymbol{\tau}_k = \boldsymbol{\tau}_{k,E} - \boldsymbol{\tau}_{k,0} = -\eta \sum_{e=0}^{E-1} \boldsymbol{h}_{k,e} + \rho\eta \sum_{e=0}^{E-1} \exp(-\boldsymbol{\tau}_{k,e}),$$

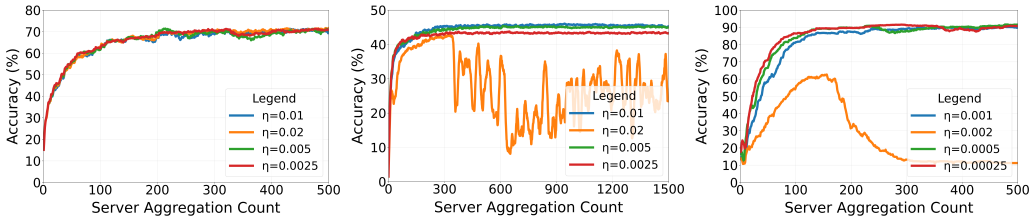
where every iterate $\boldsymbol{\tau}_{k,e}$ appearing in the regularizer is itself generated by the same recursion from $\boldsymbol{\tau}_{k,0}$ and $\{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1}$. Hence, conditioned on $\boldsymbol{\tau}_{k,0}$, the increment $\Delta\boldsymbol{\tau}_k$ is a deterministic function of $\{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1}$ alone, so $\boldsymbol{D} \to \{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1} \to \Delta\boldsymbol{\tau}_k$ is a Markov chain given $(\boldsymbol{\tau}_{k,0}, \boldsymbol{w})$, and the DPI yields

$$I(\boldsymbol{D}; \Delta\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}) \le I(\boldsymbol{D}; \{\boldsymbol{h}_{k,e}\}_{e=0}^{E-1} \mid \boldsymbol{\tau}_{k,0}, \boldsymbol{w}),$$

which together with the equality proves equation 24. □

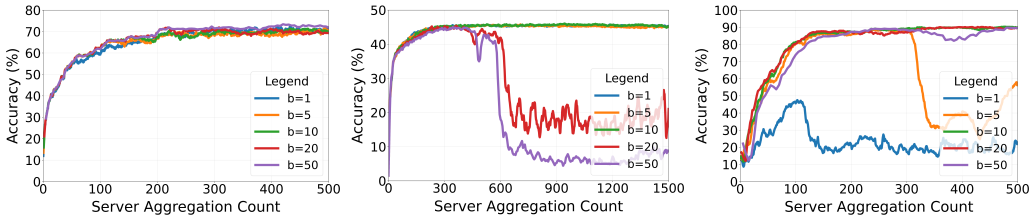
Theorem 3 connects the theory of Section 4 to the GatedAFL protocol. Conditioned on the threshold $\boldsymbol{\tau}_{k,0}$ that the server itself issued and $\boldsymbol{w}$, the communicated increment $\Delta\boldsymbol{\tau}_k$ leaks no more information about the private data $\boldsymbol{D}$ than the sequence of per-step threshold gradients that produced it. Thus, a leakage bound for that sequence also bounds the increment actually transmitted by GatedAFL. Theorem 1 gives the corresponding single-step comparison at fixed model parameters.

Equation (24) clarifies the role of transmitting increments. Against the honest-but-curious server that issued $\boldsymbol{\tau}_{k,0}$, the increment and the final threshold are informationally equivalent, so the benefit of communicating increments lies in the staleness-robust aggregation of equation 7. Against an eavesdropper that observes only the uplink and does not know $\boldsymbol{\tau}_{k,0}$, the increment additionally hides the absolute position of the threshold, since any shifted pair $(\boldsymbol{\tau}_{k,0} + \boldsymbol{c}, \boldsymbol{\tau}_{k,E} + \boldsymbol{c})$ is consistent with the observed $\Delta\boldsymbol{\tau}_k$.



(a) CIFAR-10 (b) CIFAR-100 (c) FMNIST

Figure 5: Sensitivity of GatedAFL to the learning rate η.



(a) CIFAR-10 (b) CIFAR-100 (c) FMNIST

Figure 6: Sensitivity of GatedAFL to the aggregation buffer size b.

B Additional Sensitivity Experiments

We further evaluate the sensitivity of GatedAFL to four factors that govern local optimization, asynchronous aggregation, federation scale, and statistical heterogeneity: the learning rate η, the server buffer size b, the number of clients K, and the Dirichlet concentration parameter α. We vary one factor at a time while keeping the remaining configurations identical to those in Section 5.1. Figures 5–8 report test accuracy against the number of server aggregations.

B.1 Sensitivity to the learning rate

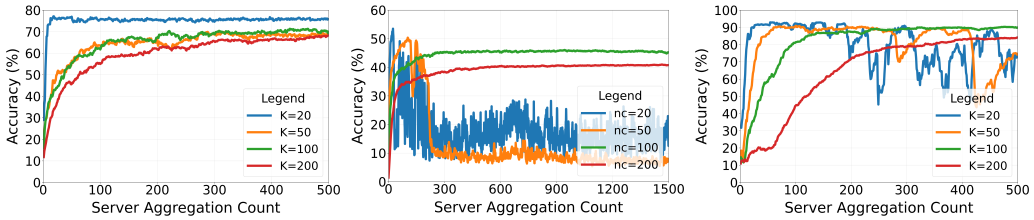
Figure 5 evaluates the sensitivity of GatedAFL to the local learning rate. Specifically, we consider $\eta \in \{0.0025, 0.005, 0.01, 0.02\}$ for CIFAR-10 and CIFAR-100, and $\eta \in \{0.00025, 0.0005, 0.001, 0.002\}$ for FMNIST.

On CIFAR-10, all four learning rates show similar convergence behavior and stabilize at comparable accuracy, with only minor differences throughout training. On CIFAR-100, the learning rate has a more pronounced effect. The settings $\eta = 0.0025$, 0.005, and 0.01 all converge stably, with $\eta = 0.01$ attaining the highest accuracy of 45.24%, followed closely by $\eta = 0.005$. In contrast, $\eta = 0.02$ becomes highly unstable after 342 server aggregation rounds, exhibiting large accuracy oscillations and a substantial degradation in later training. On FMNIST, $\eta = 0.00025, 0.0005$, and 0.001 all converge to approximately 90% accuracy, although their early-stage convergence speeds differ. In contrast, $\eta = 0.002$ improves rapidly at the beginning but becomes unstable after 156 server aggregations and subsequently declines.

Overall, the learning rate has a limited effect over the tested range on CIFAR-10, whereas CIFAR-100 and FMNIST become sensitive to the largest learning rate tested. In particular, the instability of $\eta = 0.02$ on CIFAR-100 and $\eta = 0.002$ on FMNIST suggests that excessively large learning rates can destabilize the joint optimization of the personalized model parameters and trainable thresholds.

B.2 Sensitivity to the aggregation buffer size

Figure 6 studies the effect of the server buffer size with $b \in \{1, 5, 10, 20, 50\}$. GatedAFL aggregates threshold increments once the buffer is full, so b determines the number of arriving client updates averaged before each global threshold update.



(a) CIFAR-10 (b) CIFAR-100 (c) FMNIST

Figure 7: Sensitivity of GatedAFL to the number of clients K.

On CIFAR-10, the different buffer sizes produce similar convergence behavior and final accuracy. On CIFAR-100, small and moderate buffer sizes $b = 1$, 5, and 10 exhibit similar stable convergence and remain around 45% accuracy range. In contrast, the larger buffers $b = 20$ and $b = 50$ become unstable after 542 and 535 server aggregations, respectively, with accuracy dropping sharply and remaining substantially below that of the smaller buffer configurations. FMNIST shows a stronger dependence on b. With $b = 1$, applying each arriving threshold increment directly results in persistent oscillations and low accuracy. Increasing the buffer to $b = 5$ improves early convergence, but the accuracy drops substantially at later stages. In comparison, $b = 10$ and $b = 20$ remain close to 90% accuracy after convergence. The larger buffer $b = 50$ also reaches a similar accuracy level, with a temporary decrease later in training.

These results show that averaging several asynchronous threshold increments can reduce the variability of individual updates before they are applied to the global threshold, but an excessively large buffer can also increase the effective staleness of the aggregated information. The effect is relatively small on CIFAR-10, while CIFAR-100 is particularly sensitive to large buffer sizes and FMNIST benefits most from moderate buffers.

B.3 Sensitivity to the number of clients

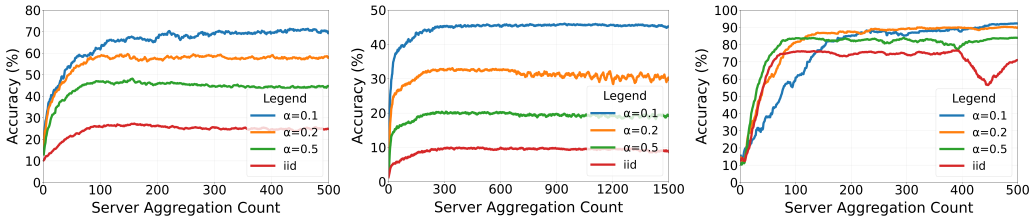
Figure 7 evaluates GatedAFL under different federation scales, with $K \in \{20, 50, 100, 200\}$. Varying K changes both the amount of local data available to each client and the composition of asynchronous client updates.

On CIFAR-10, $K = 20$ converges fastest and stabilizes at 75.87% accuracy. As K increases to 50, 100, and 200, early-stage convergence becomes progressively slower, while all configurations continue to improve over training. CIFAR-100 exhibits a different pattern. With $K = 20$ and $K = 50$, the accuracy initially increases rapidly but subsequently becomes highly unstable and drops substantially. In comparison, $K = 100$ provides the most stable trajectory and converges to 45.24% accuracy. Increasing the population further to $K = 200$ results in slower convergence and a lower final accuracy of 40.77%, although training remains considerably more stable than with $K = 20$ or $K = 50$. On FMNIST, $K = 20$ and $K = 50$ reach high accuracy rapidly but exhibit pronounced fluctuations later in training. In comparison, $K = 100$ converges smoothly to 89.96% accuracy and remains relatively stable. Increasing the number of clients to $K = 200$ slows convergence, but produces a smoother trajectory than the smaller client configurations.

The results reveal a federation-scale trade-off between convergence speed and training stability. With fewer clients, each client holds more local data, which generally facilitates faster early optimization, whereas increasing the number of clients reduces the amount of local data per client and tends to slow convergence. Meanwhile, the effect of the client population on training stability varies across datasets. Overall, the results show that the choice of K affects both convergence dynamics and stability, with the balance differing across the tested datasets.

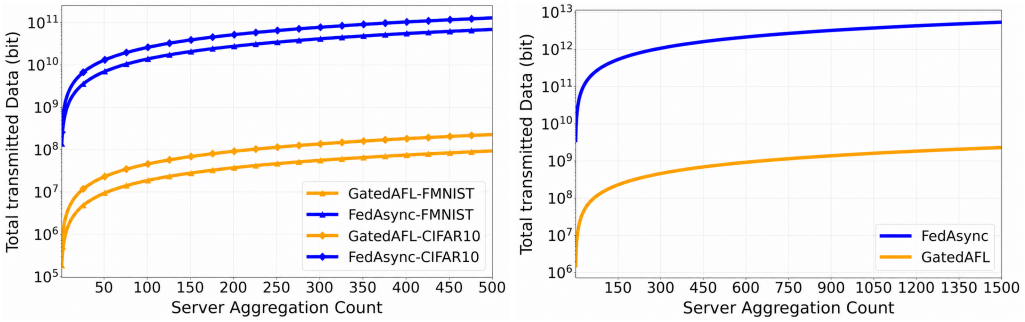
B.4 Sensitivity to statistical heterogeneity

Figure 8 evaluates the effect of statistical heterogeneity by varying the Dirichlet concentration parameter $\alpha \in \{0.1, 0.2, 0.5\}$, with independent and identically distributed (IID) partitioning included



(a) CIFAR-10 (b) CIFAR-100 (c) FMNIST

Figure 8: Sensitivity of GatedAFL to the Dirichlet parameter α, with IID partitioning for comparison.



(a) CIFAR10 and FMNIST (b) CIFAR100

Figure 9: Cumulative transmitted data of GatedAFL and FedAsync against the number of server aggregations on CIFAR-10, FMNIST, and CIFAR-100.

for comparison. Smaller values of α correspond to more heterogeneous label distributions across clients.

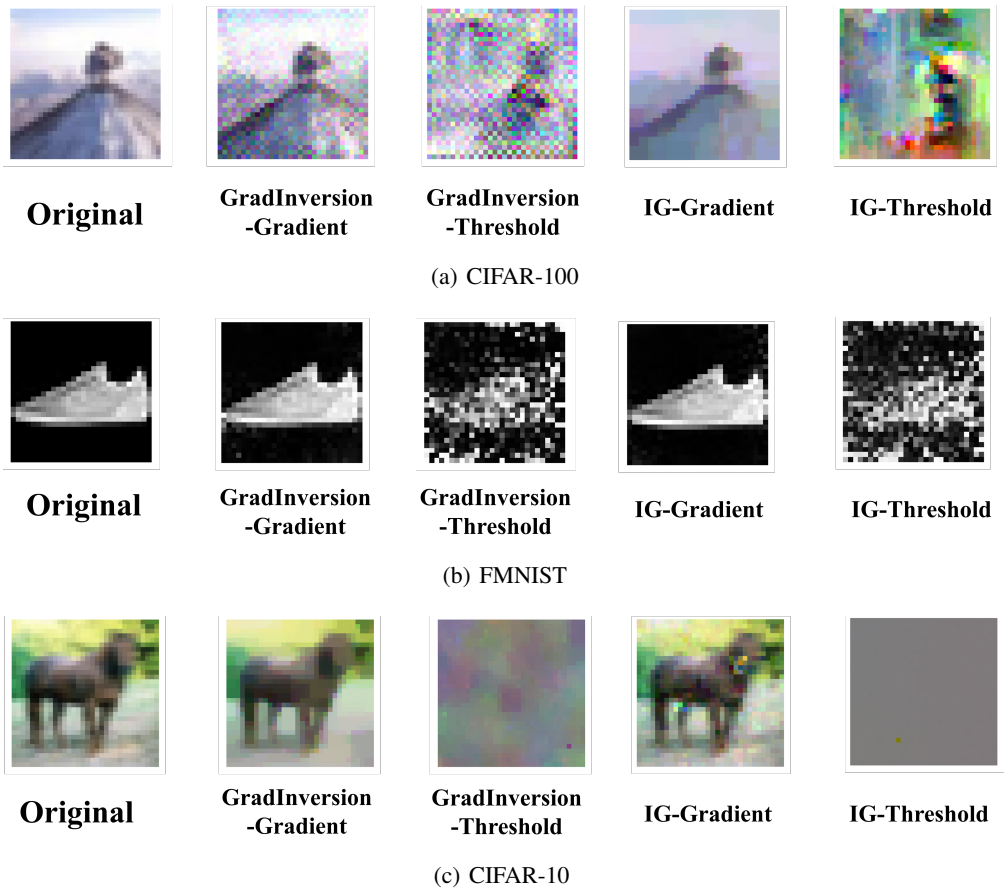
On CIFAR-10, $\alpha = 0.1$ achieves the highest accuracy at approximately 69.42%, followed by $\alpha = 0.2$ and $\alpha = 0.5$, while the IID setting reaches a lower plateau. The same trend is even more pronounced on CIFAR-100. The $\alpha = 0.1$ configuration converges to 45.24% accuracy, whereas $\alpha = 0.2$ and $\alpha = 0.5$ stabilize at progressively lower levels, and the IID setting achieves the lowest accuracy. On FMNIST, the $\alpha = 0.1$ and $\alpha = 0.2$ settings also achieve higher final accuracy than $\alpha = 0.5$ and the IID setting.

Overall, the results show that statistical heterogeneity leads to different accuracy levels across the tested datasets, with smaller α generally yielding better performance. The magnitude of this effect varies across datasets, indicating that GatedAFL responds differently to the degree of data heterogeneity under different model and dataset settings. These observations are consistent with the personalized design of GatedAFL, in which model parameters remain local to each client while only the trainable thresholds are shared globally.

C Cumulative Communication Cost Over Training

Figure 9 reports the cumulative transmitted data against the number of server aggregations on CIFAR-10, CIFAR-100, and FMNIST. As training proceeds, the cumulative communication cost of both methods increases continuously, while GatedAFL remains substantially below FedAsync throughout the entire training process. At the end of training, GatedAFL transmits 2.27×10^8, 9.28×10^7, and 2.30×10^9 bits on CIFAR-10, FMNIST, and CIFAR-100, respectively, compared with 1.29×10^{11}, 6.89×10^{10}, and 5.36×10^{12} bits for FedAsync.

This persistent gap follows directly from the quantities exchanged at each interaction. FedAsync communicates full model parameters, whereas GatedAFL transmits only compact threshold incre-



(a) CIFAR-100

(b) FMNIST

(c) CIFAR-10

Figure 10: Reconstruction results under IG and GradInversion on CIFAR-10, CIFAR-100, and FMNIST.

ments, with one threshold associated with each neuron or filter. Therefore, the reduction is incurred at every communication event and accumulates throughout training.

The same behavior is observed across all three datasets despite their different model architectures and training horizons. In particular, CIFAR-100 is evaluated for 1500 server aggregations and consequently accumulates substantially more transmitted data than the other two datasets, yet the separation between GatedAFL and FedAsync remains pronounced throughout training. These results demonstrate that the communication efficiency of GatedAFL is sustained as training progresses and across different dataset and model settings, rather than being limited to a particular training stage or experimental configuration.

D Additional Reconstruction Results Under Gradient-Inversion Attacks

Figure 10 provides additional reconstruction examples for CIFAR-10, CIFAR-100, and FMNIST under IG and GradInversion. For each attack, we compare reconstructions obtained from the full model gradient with those obtained from the threshold representation.

The qualitative results show a clear difference between the two shared representations. With gradient sharing, both attacks recover substantial visual information from the original samples. On CIFAR-10, the reconstructed images preserve the object shape and much of the surrounding appearance. A similar pattern is observed on CIFAR-100, where the main object and its spatial structure remain recognizable despite reconstruction artifacts. On FMNIST, the shape of the clothing item is recovered particularly clearly from the gradients.

In comparison, reconstruction from the threshold representation is substantially degraded across all three datasets. On CIFAR-10, the reconstructed images contain little recognizable object structure, while the CIFAR-100 reconstructions are dominated by distorted colors and spatial artifacts. On FMNIST, the threshold-based reconstructions retain at most coarse intensity patterns, but the object boundary and detailed shape are considerably less distinguishable than in the gradient-based reconstructions. These qualitative differences are consistent across both IG and GradInversion and agree with the quantitative results in Table 1, where threshold sharing yields higher MSE and lower PSNR and SSIM than gradient sharing across the three datasets and both attacks.