FEDERATED DATA PRUNING VIA ACTIVATION STABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) keeps data on-device, but pays for it in compute: clients spend most of every round re-processing data that have already stopped contributing to learning. Existing FL data-pruning methods rely on static importance scores, server-side validation, or aggregated centroids: assumptions that break under privacy constraints and non-IID heterogeneity. In this work, we ask whether a purely local, training-aware data redundancy signal exists, and answer in the affirmative. We introduce activation stability-based data pruning at the batch level for FL, a lightweight, architecture-agnostic plug-in that permanently removes batches whose mean hidden-activation variance saturates across iterations, a signal already produced by the local forward pass, requiring no extra communication, no server data, and no architectural change. We show that activation stability and client drift evolve on decoupled time scales, explaining why local pruning remains reliable under heterogeneity. Across CIFAR-10/100, SVHN, and ImageNet-1K with ResNet-18/50 and CvT, our framework discards up to 45% of training batches at almost no accuracy cost, saving up to 45% of the GPU node-hours.

1 INTRODUCTION

Federated Learning (FL) enables collaborative training across decentralized clients without sharing raw data, and is central to privacy-preserving machine learning. It is widely used in applications such as computer vision Chen et al. (2023), medical imaging Koutsoubis et al. (2024), transportation analytics Alam et al. (2023), and autonomous driving Fanì et al. (2023). As FL scales, system costs, client computation, energy consumption, time, and battery usage become major bottlenecks. A key but under-explored contributor is *data-side redundancy*: once local representations converge, clients repeatedly process data that no longer provide useful learning signals. While centralized training has addressed this via data pruning, identifying informative samples in FL is more challenging. Data is fragmented across clients, decisions must be made locally without communication and client participation is intermittent. These constraints make training-aware, purely on-device data pruning in FL both challenging and largely underexplored.

Why existing efficiency tools fall short in FL. Existing FL efficiency methods sidestep the data-redundancy question rather than answering it. Network pruning approaches sparsify the model on each client to lower compute Jiang et al. (2022); Meng et al. (2023); Huang et al. (2024), but introduce architectural divergence that complicates aggregation. Coreset and data-valuation methods for FL Sivasubramanian et al. (2024); Chanda et al. (2025); Wei et al. (2025) prune the data, but typically rely on static importance scores computed before training, on server-held validation sets, or on aggregated class centroids, all of which conflict with FL's privacy model and ignore evolving training dynamics. Centralized data pruning, whether static Toneva et al. (2018); Paul et al. (2021) or dynamic Qin et al. (2024); He et al. (2024); Raju et al. (2021), assumes continuous access to a full, independent, and identically distributed (IID) dataset with globally consistent statistics; none of these assumptions survive once training is fragmented across non-IID, partially-attended clients. The closest FL counterpart, the recent FedCS coreset method Hao et al. (2025), uses a distance-contrast score with double-pruning and removes data at the sample level, which is known to amplify class imbalance under heterogeneity. A recent line of work, batch pruning by activation stability (B-PAS) Alam et al. (2026), provides an appealing starting point in the centralized setting. B-PAS shows that the per-batch mean standard deviation of hidden activations stabilizes once a batch's representation converges, and leverages this signal to permanently prune saturated batches without auxiliary networks, gradient inspection, or validation labels. Activation stability is particularly

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.

attractive for FL because it is computed during the forward pass and is purely local, requiring no cross-client communication or server-side data. However, B-PAS is developed under centralized, IID, and fully-attended training, and its extension to FL is non-trivial. In FL, (i) local models evolve under global drift, so previously "saturated" batches may become informative again; (ii) intermittent client participation leads to stale per-batch histories Rodio & Neglia (2024); (iii) skewed local distributions raise concerns about class imbalance from permanent pruning; and (iv) layer-wise aggregation may mask ongoing learning in deeper representations. These factors make it unclear whether activation stability remains a reliable redundancy signal in federated settings.

Our position and framework. This paper addresses the following question: *Does activation stability remain a valid signal of data redundancy under federated, non-IID, partially-participating training, and if so, how can it be exploited without auxiliary communication or server-side data?* Rather than assuming a direct transfer, we empirically study how activation stability interacts with key FL dynamics, including client drift, data heterogeneity, and stale local state, and translate these observations into a practical on-device pruning framework. Our framework performs batch pruning driven entirely by activation-stability on each client, using only signals produced during the local forward pass and a small persistent history per-client. Some key findings underwrite both the design and its reliability in federated settings. First, despite the challenges outlined above, activation stability remains a robust and reliable signal in federated settings. In particular, activation stability captures local representational convergence and is largely unaffected by heterogeneity, while client drift captures global disagreement and evolves on a different time scale. This decoupling explains why stability-based pruning can be applied locally without coordination and remain effective under non-IID conditions. Second, batch-level pruning is empirically class-agnostic under heterogeneous partitions, mitigating the concern that permanent local pruning amplifies class imbalance. Third, the multi-layer mean stability score is robust to layer choice: across 11 tracked-layer configurations, accuracy varies by less than 0.7%, and excluding early layers does not improve performance, indicating that layer aggregation does not mask ongoing learning. We operationalize these findings via per-client persistent stability histories that tolerate intermittent participation, a smoothly scheduled pruning threshold that transfers across datasets with minimal calibration, and a plug-in pruning step compatible with FedAvg McMahan et al. (2017) and standard FL optimizers without architectural changes or additional communication.

Our contributions are threefold. **A federated reformulation of activation-stability-based pruning.** We adapt activation-stability-based batch pruning to federated learning with a staleness-aware mechanism, and systematically characterize its behavior under high heterogeneity and partial participation. To the best of our knowledge, this is the first work to establish activation stability as a practical on-device pruning signal in federated learning under these settings. **An analysis disentangling stability from client drift.** We show that activation stability and client drift evolve on different timescales and respond differently to heterogeneity. The decoupling explains why stability remains a robust local signal under non-IID conditions and justifies pruning entirely on-device rather than via global coordination. **Scalability and energy impact.** We demonstrate the framework from CIFAR-scale Krizhevsky et al. (2009a;b) to ImageNet-1K Krizhevsky et al. (2012) and from ResNet-18 through CvT Wu et al. (2021), achieving up to **45%** data reduction at minimal to no accuracy cost, with corresponding savings in client compute and energy.

2 Related work

Research on efficient training spans both centralized data pruning and federated data pruning, each aiming to reduce redundancy and cost while preserving model performance. In centralized setting, static methods select representative subsets prior to training using geometric or diversity-based criteria such as core-sets and k-Center Sener & Savarese (2018); Agarwal et al. (2020); Welling (2009), or scoring functions based on uncertainty Coleman et al. (2019); Gal et al. (2017), forgetting events Toneva et al. (2018), adversarial influence Ducoffe & Precioso (2018), and gradient-based importance (GraNd, EL2N) Paul et al. (2021). More principled approaches use bilevel optimization and submodular objectives Killamsetty et al. (2021); Iyer et al. (2021), while ensemble and coverage-based methods (e.g., Moderate, Herding) emphasize representativeness Xia et al. (2022); Welling (2009); Zheng et al. (2023). Dynamic pruning leverages training-time signals Raju et al. (2021); Qin et al. (2024); He et al. (2024), and dataset distillation synthesizes compact datasets via gradient, distribution, or trajectory matching Cazenavette et al. (2022); Liu et al. (2023); Guo et al. (2024); Wang et al. (2022). These centralized approaches, however, do not directly transfer to federated learning due to privacy and decentralization constraints.

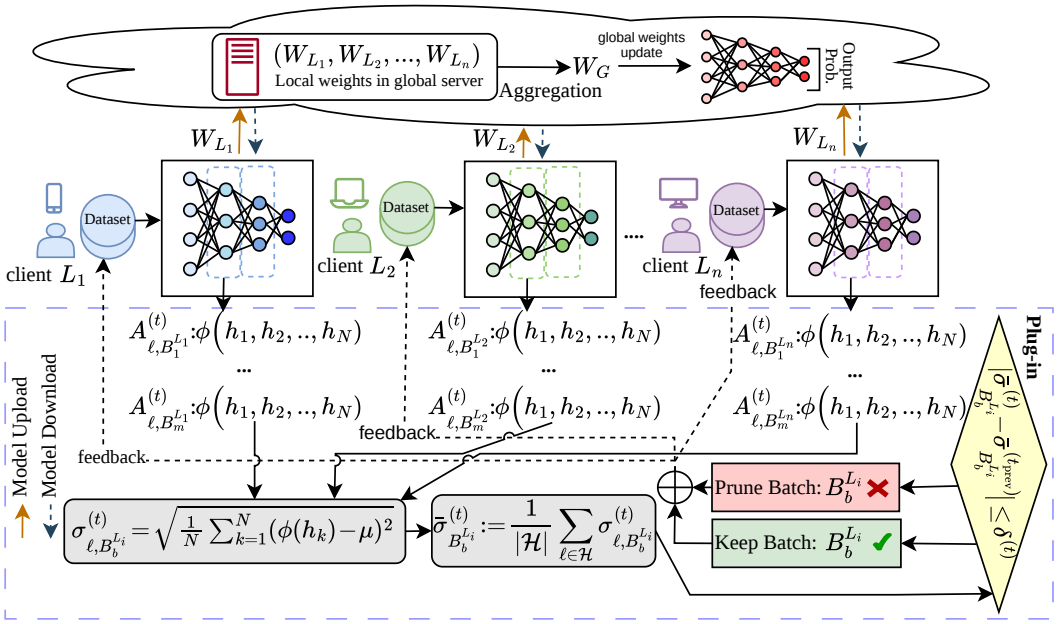


Figure 1: Framework of batch pruning by activation stability in federated learning. Each client i trains its local model L_i on private data and transmits the updated parameters to the global server, where they are aggregated to form the global model. During local training, each batch $B_b^{L_i}$ ($b \in 1, \ldots, m$) is monitored through the mean standard deviation of its hidden-layer activation tensors across iterations. The flattened activation tensors ($\phi(h_1, h_2, .., h_N)$) for the b-th batch at layer ℓ and iteration t are denoted as $A^{(t)}_{\ell,B_b^{L_i}} : \phi(h_1, h_2, \ldots, h_N)$, where $\ell \in \mathcal{H}$ and $|\mathcal{H}|$ represents the total number of tracked layers. For all networks in the figure, $|\mathcal{H}| = 2$. The per-layer standard deviation of the activations is expressed as $\sigma^{(t)}_{\ell,B_b^{L_i}}$, and the mean standard deviation across layers for batch $B_b^{L_i}$ is given by $\bar{\sigma}^{(t)}_{B_b^{L_i}}$. A batch is permanently pruned if the absolute change in its mean standard deviation across consecutive iterations satisfies $|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{\text{prev}})}_{B_b^{L_i}}| \le \delta^{(t)}$, where $\delta^{(t)}$ is a time-varying pruning threshold. This pruning process, governed by Algorithm 1, progressively eliminates redundant batches while retaining those that contribute meaningful feature representation updates, thereby improving data efficiency and reducing local computation across federated training rounds.

Federated data pruning must handle heterogeneity and communication constraints. Gradient-alignment coresets Sivasubramanian et al. (2024) select per-client subsets by matching server validation gradients via greedy OMP, while FedCS Hao et al. (2025) uses a Distance-Contrast score with Double-Pruning. Bayesian approaches such as CORESET-PFEDBAYES Chanda et al. (2025) learn client-specific weights with minimax guarantees. Model-level pruning methods (FedTiny Huang et al. (2023), FedDUAP Zhang et al. (2022)) focus on structured sparsity but introduce architectural heterogeneity. Other directions include valuation-based sampling Wei et al. (2025), hierarchical selection (FedHDS) Qin et al. (2025), and imbalance-aware resampling strategies Düsing et al. (2025).

3 Methodology

Here, we introduce activation-stability-based batch pruning as a plug-in framework for federated learning. Figure 1 illustrates the generalized adaptation of the approach in FL, while Algorithm 1 formalizes the pruning procedure with the integrated staleness-aware mechanism. Together, they demonstrate the complete workflow of the proposed framework. FL is a distributed machine learning paradigm in which model training is performed locally on client devices and only model weights are communicated with a central server. As illustrated in Figure 1, each client maintains a local model, L_i with weights, $W_{L_i} = \{w_1^{L_i}, w_2^{L_i}, \ldots\}$. During each iteration, clients update their local model parameters based on their private data and send the updated parameters to the global server. The server then aggregates these updates (e.g., using federated averaging), producing a new global parameter set W_G, which is subsequently broadcast back to all clients. This iterative process continues over multiple rounds until convergence. Within this standard FL pipeline, we incorporate a plug-in batch

pruning mechanism guided by activation stability. At each iteration, the dataset of every client is partitioned into batches, and the activation tensors produced by passing these batches through the network's hidden layers are monitored. For each batch, we compute the mean standard deviation of these activation tensors across all hidden layers. This value reflects the stability of a batch's feature representations across training rounds. If the change in mean standard deviation between two iterations is negligible, the batch is considered less informative and is permanently removed from subsequent training. This dynamically updates each client's dataset, prioritizing batches that induce meaningful representational change.

Pruning. The pruning mechanism follows Algorithm 1, adapting the activation-stability–based tracking and pruning framework of Alam et al. (2026) to operate locally in federated settings with staleness-aware updates. Consider the hidden layers of a neural network, where activations are represented as flattened tensors. For image data, each batch comprises multiple images whose pixel or feature map activations form these tensors. As illustrated in Figure 1, for a given local model L_1, the batches are denoted as $B_1^{L_1}, \ldots, B_m^{L_1}$, where m is the total number of batches in that client's dataset. During local training, the algorithm monitors the stability of activations within each batch. Specifically, for every batch, the standard deviation of its activation tensors is computed layer-wise. If the network has $|\mathcal{H}|$ hidden layers, then each batch produces $|\mathcal{H}|$ standard deviation values, one per layer. The mean of these values defines a batch stability score, capturing the overall representational change within the batch. A batch is considered converged and is permanently pruned when the absolute difference between its stability scores at two iterations, t and t_{prev}, falls below a value. Here, an iteration corresponds to one local epoch. The reference iteration t_{prev} denotes the most recent iteration in which the client participated, coinciding with $t-1$ under continuous participation, and with the last active round otherwise. This dynamic pruning continuously refines each client's local dataset, ensuring that only batches inducing meaningful feature updates remain active across iterations.

Algorithm 1: Pruning Algorithm

Require: Clients $i = 1{:}n$ (local models L_i); batches per client m; tracked layers $\mathcal{H}$; iterations T; threshold schedule $\{\delta^{(t)}\}_{t=1}^{T}$; staleness threshold k

Initialize: For all clients i, set
  $\mathcal{S}_i^{(1)} \leftarrow \{B_1^{L_i}, \ldots, B_m^{L_i}\}$; $\mathcal{P}_i^{(1)} \leftarrow \varnothing$; staleness counter $c_i \leftarrow 0$; stability history $\bar{\sigma}_i^{(\mathrm{t_{prev}})} \leftarrow \texttt{None}$
for $t = 1$ **to** T **do**
  for each participating client i **do**
    if $c_i > k$ **then**
      Reset $\bar{\sigma}_i^{(\mathrm{t_{prev}})} \leftarrow \texttt{None}$ ▷ discard stale history
    Compute staleness factor:
      $f_i \leftarrow \max\big(0.5,\ 1 - \frac{c_i}{2k}\big)$
    for each active batch $B_b^{L_i} \in \mathcal{S}_i^{(t)}$ **do**
      Forward pass on $B_b^{L_i}$; collect $\{A^{(t)}_{\ell, B_b^{L_i}}\}_{\ell \in \mathcal{H}}$
      Compute per-layer std:
      $\sigma^{(t)}_{\ell, B_b^{L_i}} \leftarrow \mathrm{Std}(A^{(t)}_{\ell, B_b^{L_i}})$ for all $\ell \in \mathcal{H}$
      Mean across layers:
      $\bar{\sigma}^{(t)}_{B_b^{L_i}} \leftarrow \frac{1}{|\mathcal{H}|} \sum_{\ell \in \mathcal{H}} \sigma^{(t)}_{\ell, B_b^{L_i}}$
    if $\bar{\sigma}_i^{(\mathrm{t_{prev}})} \neq \texttt{None}$ **then**
      $\mathcal{P}_i^{(t)} \leftarrow \big\{ B_b^{L_i} \in \mathcal{S}_i^{(t)} \ :\ \big|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(\mathrm{t_{prev}})}_{B_b^{L_i}}\big| \le f_i \cdot \delta^{(t)} \big\}$
      Permanently remove pruned batches:
      $\mathcal{S}_i^{(\mathrm{t_{next}})} \leftarrow \mathcal{S}_i^{(t)} \setminus \mathcal{P}_i^{(t)}$
    else
      $\mathcal{S}_i^{(\mathrm{t_{next}})} \leftarrow \mathcal{S}_i^{(t)}$ ▷ skip pruning
    $\bar{\sigma}_i^{(\mathrm{t_{prev}})} \leftarrow \bar{\sigma}_i^{(t)}$ ▷ store current as prior
    Reset staleness counter $c_i \leftarrow 0$
  Increment c_j for all non-participating clients j

Formally, let $i \in \{1, \ldots, n\}$ index local clients (model L_i), $\ell \in \mathcal{H}$ denote the tracked hidden layers, $b \in \{1, \ldots, m\}$ represent the batch index, and $t \in \{1, \ldots, T\}$ denote the training iteration. For client L_i, the b-th batch is expressed as $B_b^{L_i}$. A forward pass generates layer activations $A^{(t)}_{\ell, B_b^{L_i}}$ from flattened activation tensors $\phi(h_1, h_2, \ldots, h_N)$ for each $\ell \in \mathcal{H}$, where N is the number of scalar activations the batch produces at layer ℓ, from which we compute the per-layer activation standard deviation: $\sigma^{(t)}_{\ell, B_b^{L_i}} := \mathrm{Std}\big(A^{(t)}_{\ell, B_b^{L_i}}\big), \quad \ell \in \mathcal{H}$. The mean standard deviation across layers defines the per-batch activation stability score: $\bar{\sigma}^{(t)}_{B_b^{L_i}} := \frac{1}{|\mathcal{H}|} \sum_{\ell \in \mathcal{H}} \sigma^{(t)}_{\ell, B_b^{L_i}}$.

To identify negligible representational changes, a hyperparameter threshold schedule $\delta^{(t)} > 0$ is introduced, which varies over iterations. A batch is permanently pruned when its stability variation between two consecutive participations is small: $\left|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{\text{prev}})}_{B_b^{L_i}}\right| \leq \delta^{(t)}$, where t_{prev} denotes the last iteration at which client i participated, rather than the immediately preceding global round. We denote by $\mathcal{S}_i^{(t)} \leftarrow \{B_1^{L_i}, \ldots, B_m^{L_i}\}$ the set of active batches for client i at iteration t, and by $\mathcal{P}_i^{(t)}$ the set of batches pruned at iteration t. Once a batch is pruned, it is excluded from all future iterations, leading to a progressively smaller yet more informative dataset that enhances energy and data efficiency across federated rounds. If all batches across all clients are pruned, training is terminated.

Since activation stability scores are maintained locally, a client that has been absent for several rounds holds statistics computed under an earlier global model. Directly comparing these stale scores against freshly computed ones risks incorrect pruning: a batch may appear stable not because it has converged, but because the outdated reference no longer reflects the current model state. We address this with a per-client staleness counter c_i, tracking the number of rounds since client i last participated. Upon rejoining, the counter governs a two-tier mechanism. For moderate staleness ($0 < c_i \leq k$), the pruning threshold, δ is scaled by a per-client factor, $f_i = \max\left(0.5,\ 1 - \frac{c_i}{2k}\right)$, yielding an adjusted threshold schedule $f_i \cdot \delta^{(t)}$ and a correspondingly stricter pruning condition: $\left|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{\text{prev}})}_{B_b^{L_i}}\right| \leq f_i \cdot \delta^{(t)}$. The factor, f_i, decays linearly from 1.0 (no absence; original schedule recovered) to 0.5 (at the reset boundary $c_i = k$), making pruning progressively more conservative as staleness increases, while the floor of 0.5 ensures pruning remains active rather than halting entirely. For severe staleness ($c_i > k$), the stored stability scores $\bar{\sigma}_i^{(t_{\text{prev}})}$ are discarded, the client trains for one round without pruning to establish a fresh baseline ($\mathcal{S}_i^{t_{\text{next}})} = \mathcal{S}_i^{(t)}$), and pruning resumes from the subsequent participation using the newly recorded scores as reference. After each participation, the counter is reset to $c_i \leftarrow 0$, and the current scores are stored for future comparison; non-participating clients have their counters incremented. In the absence of a partial participation issue or staleness, pruning follows the standard condition $\left|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{\text{prev}})}_{B_b^{L_i}}\right| \leq \delta^{(t)}$, with no staleness-aware adjustments or history resets applied. For clarity, (1) if $\mathcal{P}_i^{(t)} = \varnothing$ for a client, its dataset remains unchanged, $\mathcal{S}_i^{(t_{\text{next}})} = \mathcal{S}_i^{(t)}$. (2) Pruning is permanent: once $B_b^{L_i}$ is removed at line 18, it never reappears in later iterations. (3) A time-varying $\delta^{(t)}$ can be used without changing the algorithmic structure.

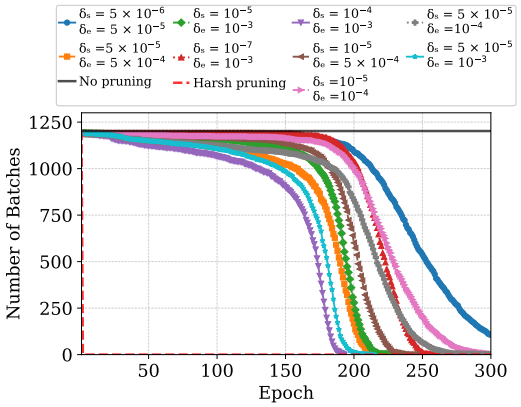


Figure 2: Effect of the pruning threshold schedule on batch removal in federated condition. Each curve shows a different (δ_s, δ_e) setting in the exponential schedule $\delta^{(t)}$. Small thresholds (e.g., 5×10^{-6}–5×10^{-5}) lead to conservative pruning, while large thresholds (e.g., 10^{-4}–10^{-3}) prune aggressively and may collapse training. A balanced choice such as $(5{\times}10^{-5}, 5{\times}10^{-4})$ achieves steady, stable pruning. The horizontal line denotes no pruning, and sharp early drops indicate overly aggressive configurations.

Dynamic schedule for pruning. A tolerance hyperparameter δ determines whether a batch is removed by thresholding changes in activation variance across iterations. Rather than enforcing exact invariance, δ provides a practical margin for detecting variance saturation. Its magnitude controls pruning behavior: small values lead to conservative pruning, while large values cause aggressive removal that may discard informative batches and destabilize training. To balance these extremes, a smoothly increasing schedule is employed: $\delta^{(t)} = \delta_s e^{\alpha t}, \quad \alpha = \frac{1}{T}\ln\left(\frac{\delta_e}{\delta_s}\right)$, where δ_s and δ_e denote the initial and final threshold, t is the current iteration, and T is the total number of iterations. This schedule yields conservative pruning in early rounds, when representations evolve rapidly, and increases aggressiveness as training stabilizes. At iteration t, a batch $B_b^{L_i}$ is pruned when $\left|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{prev})}_{B_b^{L_i}}\right| \leq \delta^{(t)}$. Figure 2 illustrates the pruning behavior for nine (δ_s, δ_e) configurations on ImageNet-1K with ResNet-50 in federated condition. Extremely 'aggressive pruning' (harsh pruning)

removes all batches by iteration 2 which is portrayed by the dotted vertical line and 'no pruning' is portrayed by the horizontal line. Lower threshold ranges such as $[5 \times 10^{-6}, 5 \times 10^{-5}]$ behave conservatively, and higher ranges such as $[10^{-4}, 10^{-3}]$ prune too aggressively and often halt training prematurely. A balanced setting of $\delta_s = 5 \times 10^{-5}$ and $\delta_e = 5 \times 10^{-4}$ enables smooth and stable pruning without disrupting optimization. Although effective, the choice of δ remains task dependent and can be tuned according to dataset difficulty, architecture, and resource constraints.

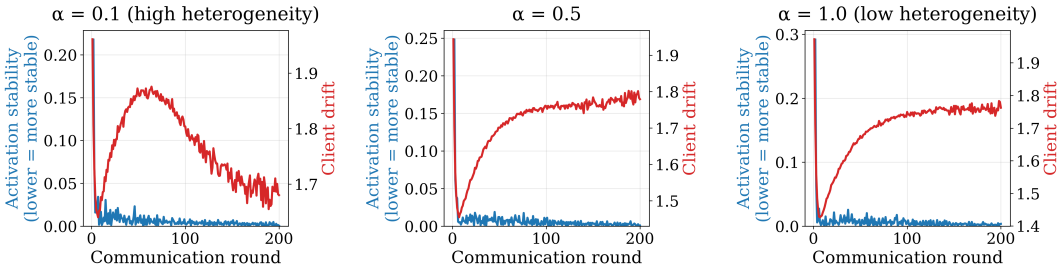


Figure 3: Evolution of activation stability and client drift across heterogeneity levels. Activation stability rapidly saturates early, while client drift evolves over longer horizons and varies with heterogeneity, highlighting their distinct roles.

4 Activation Stability under Heterogeneous Data

Federated learning operates under non-IID data distributions that introduce client drift and complicate global optimization. Although our pruning framework is local, it is important to understand how heterogeneity affects activation stability. We analyze this empirically across varying heterogeneity levels. We simulate heterogeneity using a Dirichlet distribution with concentration parameter $\alpha \in \{0.1, 0.5, 1.0\}$ on CIFAR-10 with 40 probe batches, where a smaller α corresponds to stronger non-IID partitions. Probe batches are a fixed subset of randomly selected training batches tracked across communication rounds to monitor activation stability and client drift under different heterogeneity levels. They are treated identically during training and pruning, serving only as a consistent basis for analysis. Activation stability is computed as in Section 3. To contextualize its behavior, we also measure client drift, defined as the mean L2 norm of each client's local parameter update relative to the global model, averaged across all participating clients, capturing cross-client disagreement.

Stability vs. drift. Figure 3 compares activation stability ($\left|\bar{\sigma}^{(t)}_{B_b^{L_i}} - \bar{\sigma}^{(t_{prev})}_{B_b^{L_i}}\right|$) with client drift. Activation stability saturates within the first $\sim$150 communication rounds (with one local epoch) and remains tightly bounded within a narrow stability range ($\pm$0.001). Client drift, although strongly influenced by heterogeneity in earlier stages, also stabilizes after a similar number of rounds, converging to a consistent drift range ($\pm$0.05) across all heterogeneity levels. This indicates a mild temporal alignment in their saturation behavior, as both signals eventually stabilize, but their magnitude of variation differs. Under high heterogeneity ($\alpha = 0.1$), after $\sim$150 rounds, client drift exhibits noticeably larger fluctuations, whereas for low heterogeneity ($\alpha = 0.5$ or 1.0), the fluctuations persist but remain comparatively smaller. In contrast, activation stability remains consistently stable across all heterogeneity levels. Drift captures *global disagreement across clients* and is sensitive to heterogeneity, whereas activation stability captures *local representational convergence* and remains largely unaffected by the change in client drift due to heterogeneity.

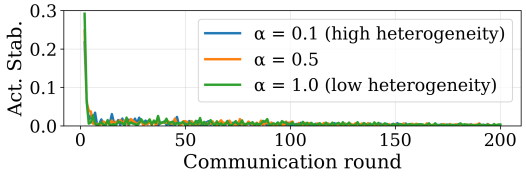


Figure 4: Activation stability under varying heterogeneity levels (α). Overlapping trajectories indicate consistent stability dynamics across α.

Consistency across heterogeneity. Figure 4 shows that stability trajectories closely overlap across all heterogeneity levels, reaching similar saturation levels at nearly identical rounds, even under strong non-IID settings ($\alpha = 0.1$). We validate this using statistical tests: ANOVA ($p = 0.93$) and Kruskal-Wallis ($p = 0.40$) do not provide evidence of statistically significant differences across heterogeneity levels (details in Appendix A.2). This indicates that activation stability remains consistent across varying data distributions and is largely insensitive to heterogeneity. These observations explain why activation-stability-guided pruning transfers effectively to FL. Since pruning depends only

on local activation dynamics, it is significantly unaffected by global disagreement. Consequently, pruning can be applied uniformly across clients without heterogeneity-aware adjustments or additional communication. Overall, heterogeneity does not alter local representation stabilization. Activation stability, therefore, remains a reliable signal for identifying redundant batches locally, explaining the consistent pruning behavior and performance observed in Section 5.

5 Evaluations and Discussion

We evaluate our framework on CIFAR-10, CIFAR-100, SVHN, and ImageNet-1K using ResNet-18 for CIFAR-10/100/SVHN, and ResNet-50 and CvT Wu et al. (2021) for ImageNet-1K. Unless specified, all experiments use 10 federated clients with FedAvg McMahan et al. (2017). We train for 200 communication rounds on CIFAR-10/100 and SVHN, and 300 on ImageNet-1K (100 for CvT due to resource constraints). Each round contains one local epoch, corresponding to one pruning iteration, though the method naturally extends to multi-epoch local training. We evaluate under both IID and Non-IID settings, using a Dirichlet coefficient $\alpha = 0.1$ to simulate high heterogeneity. Standard augmentations (normalization, random crop, horizontal flip) are applied. Unless otherwise stated, we use $\delta \in [10^{-7}, 10^{-3}]$ for CIFAR-10/100/SVHN and $\delta \in [5 \times 10^{-5}, 5 \times 10^{-4}]$ for ImageNet-1K. Consistent with B-PAS Alam et al. (2026), δ can be calibrated using a small subset of data and transfers well across datasets and architectures (Appendix A.4), with tuning effort comparable to standard hyperparameters.

Table 1: Effect of threshold range on training efficiency and accuracy for ImageNet-1K with ResNet-50. Each setting is defined by the starting (δ_s) and ending (δ_e) thresholds. DSI denotes the Data Savings Index, and "Node Hour Saved" measures computation saved relative to full-data training. Accuracy values are statistically consistent across runs, remaining within $\pm 0.2\%$ variation.

δ_s	δ_e	DSI (%)	Acc (%)	Early Stop Round	Node Hour	Node Hour Saved (%)
5×10^{-6}	5×10^{-5}	18	75.10	–	117.80	12.63
5×10^{-5}	5×10^{-4}	42	73.32	231	101.08	25.03
10^{-5}	10^{-3}	38	73.88	231	101.23	24.92
10^{-7}	10^{-3}	28	74.80	263	104.02	22.85
10^{-4}	10^{-3}	49	72.49	193	73.75	45.30
10^{-5}	5×10^{-4}	35	74.36	246	93.04	30.99
10^{-5}	10^{-4}	26	75.06	–	108.99	19.16
5×10^{-5}	10^{-4}	33	74.38	–	100.98	25.10
5×10^{-5}	10^{-3}	45	73.02	215	80.84	40.04
Full Data	–	–	73.92	–	134.83	–

We report top-1 accuracy alongside two efficiency metrics. The data savings index (DSI) Alam et al. (2026) measures the fraction of training data avoided due to pruning relative to full-data training; a higher DSI indicates greater data reduction. GPU node-hours quantify computational cost (number of GPUs × training time in hours), and percentage savings are reported relative to full-dataset baselines. To ensure consistent variance tracking, mini-batches are fixed at initialization, while intra-batch shuffling is retained to maintain stochasticity. More implementation details are in Appendix A.1.

Result analysis

Performance analysis on ImageNet-1K. Table 1 summarizes the effect of different pruning-threshold schedules on ImageNet-1K with ResNet-50 in the IID setting. The full-data baseline attains 73.92% accuracy for 134.83 node-hours. All pruning configurations substantially reduce node-hours while maintaining comparable accuracy, demonstrating the viability of activation-stability-guided pruning at a large scale. Conservative schedules such as $[5\times10^{-6}, 5\times10^{-5}]$ prune cautiously, yielding higher-than-baseline accuracy (75.10%) and a 12.63% reduction in node-hours. More aggressive settings, e.g., $[10^{-4}, 10^{-3}]$, remove nearly half the training data (DSI=49%) and save 45.30% node-hours but lower accuracy to 72.49%,

Table 2: Comparison of accuracy between FedCS and our approach on CIFAR-10 with ResNet-18 under Dirichlet $\alpha = 0.1$. For FedCS, we use the overall pruning rate p_i: (p_i^f, p_i^l) reported in their ablation study and interpret it as the DSI, since pruning a fraction of the data directly corresponds to saving that fraction. To ensure fairness, we align the two methods by matching the closest DSI values, allowing for a direct comparison at similar effective data usage levels.

DSI (%)	FedCS (Acc)	Our (Acc)	Our (δ_s, δ_e)	FedCS (p_i^f, p_i^l)
≈ 44	59.16	82.27	$(5 \times 10^{-5}, 1 \times 10^{-4})$	(0.5, 0.1)
≈ 30	59.62	81.80	$(5 \times 10^{-6}, 1 \times 10^{-3})$	(0.3, 0.1)

reflecting the risk of prematurely removing informative batches. Moderately aggressive schedules such as $[5\times10^{-5}, 5\times10^{-4}]$ strike the best balance, achieving 73.32% accuracy, pruning 42% of the data, and reducing node-hours by 25.03%. A similar trend appears for $[1\times10^{-5}, 5\times10^{-4}]$, which improves accuracy to 74.36% while saving 30.99% node-hours. These results indicate that activation-stability-based pruning can also act as a regularizer, improving generalization while efficiently reducing data usage. Overall, the table shows that well-chosen threshold schedules can outperform full-data training in both accuracy and efficiency. This improvement arises because full-data training can overfit, whereas activation-stability-guided pruning implicitly regularizes the model by removing redundant batches, leading to better generalization. Additionally, a comparison between centralized (B-PAS Alam et al. (2026)) and federated pruning is in Appendix A.3.

Comparison with SOTA method. FedCS Hao et al. (2025) establishes a strong and comprehensive baseline by benchmarking against several state-of-the-art federated version of coreset selection methods, including EL2N Paul et al. (2021), Moderate Xia et al. (2022), GradND Paul et al. (2021), and Forgetting Toneva et al. (2018). However, due to the lack of publicly available implementation and reproducibility details, we were unable to implement FedCS for a controlled comparison directly. To ensure a fair evaluation, we instead adopt the same experimental setup reported for FedCS, using an identical dataset, model architecture, and hyperparameter configuration. As shown in Table 2, under $\alpha = 0.1$ on CIFAR-10 with ResNet-18, our method consistently outperforms FedCS at comparable levels of data reduction. The core limitation of FedCS is its reliance on static, representation-based sample pruning computed from only a few rounds of weak pretraining. Under strong non-IID data, these early representations are unstable, leading FedCS to prune informative decision-boundary samples and to further amplify client imbalance, ultimately yielding a low-performing global model. In contrast, our approach performs dynamic batch-level pruning observed throughout training, which preserves high-impact batches, reduces gradient noise, and stabilizes the aggregation process. As a result, at a similar DSI region (30–44%), our method achieves higher accuracy than FedCS, demonstrating substantially better pruning quality in heterogeneous federated environments.

Table 3: Cross-architecture and dataset robustness on CIFAR-10/100, SVHN with ResNet-18 and ImageNet-1K with ResNet-50 and CvT in IID and Non-IID settings. Our approach preserves accuracy while reducing data usage.

	CIFAR-10		CIFAR-100		SVHN		ImageNet-1K		
	IID	N-IID	IID	N-IID	IID	N-IID	IID	N-IID	CvT(IID)
Full	93.49	85.71	74.54	69.68	97.00	96.10	73.92	65.84	73.44
Our	93.16	85.48	73.28	68.16	97.07	96.00	73.32	67.40	72.74
DSI(%)	16	17	25	26	17	23	42	45	45

Architecture and dataset robustness. Table 3 summarizes the cross-dataset and cross-architecture performance of the proposed pruning framework under both IID and Dirichlet $\alpha = 0.1$ non-IID splits. Across CIFAR-10, CIFAR-100, SVHN, and ImageNet-1K using ResNet-18, ResNet-50, and CvT, accuracy remains close to the full-data baseline despite discarding a substantial fraction of training batches. For CIFAR and SVHN, accuracy drops are minimal. On ImageNet-1K, both ResNet-50 and CvT retain competitive accuracy (sometimes better) while yielding the largest data savings, which is particularly important since large-scale training incurs significantly higher computational and energy costs than small-scale datasets. Overall, the method generalizes across architectures, scales effectively to large datasets, and remains stable under non-IID conditions, demonstrating activation-stability-guided pruning as a practical and impactful strategy for data-efficient FL.

Table 4: Accuracy (%) under partial client participation on Non-IID CIFAR-10 ($\alpha = 0.1$) with ResNet-18. Full participation compares full-data training with pruning; partial participation reports pruning for two staleness thresholds k.

	Participation rate		
Method	100%	75%	50%
Full data	85.71	–	–
Pruning ($k=5$)	85.48	86.95	84.16
Pruning ($k=20$)		82.08	80.83

Partial client participation. We test how pruning handles intermittent participation on Non-IID CIFAR-10 ($\alpha=0.1$) with ResNet-18. In each round, 100%, 75%, or 50% of the clients are sampled at random, following the standard FedAvg protocol. Table 4 first anchors the method at full participation, where pruning stays within 0.23 points of full-data training. At full participation, k has no effect since no client is ever stale. With $k=5$, pruning holds its accuracy as participation drops (86.95% at 75%, 84.16% at 50%), so sampling only half of the clients costs about one point. With $k=20$, accuracy falls to 82.08% and 80.83%. This follows from the staleness rule: a larger k barely tightens

the threshold for returning clients and rarely resets their histories, so some batches are pruned by comparing against scores computed under an older global model. A small k keeps pruning decisions tied to the current model. Appendix A.5 ablates the staleness mechanism and scales the setup to 50 clients.The slight gain at 75% over full participation lies within the run-to-run variation we observe under $\alpha = 0.1$, and we do not treat it as a systematic effect.

Ablation Study

Performance with different federated algorithms. Table 5 compares FedAvg, FedProx, and FedNova under pruning. All achieve nearly identical pruning accuracy ($\approx 93\%$) while producing similar DSI values (14–16%). FedProx and FedNova perform comparably to FedAvg despite incorporating proximal or normalization corrections, showing that our method is compatible with diverse FL optimization dynamics and does not rely on a specific aggregation strategy.

Table 5: Comparison of Federated Algorithms under Pruning and Full-Data using CIFAR-10 with ResNet-18.

	Pruning DSI (%)	Pruning Acc (%)	Full Dataset Acc (%)
FedProx	16	93.00	93.25
FedNova	14	93.00	93.53
FedAvg	16	93.16	93.49

Random pruning vs our. To assess activation stability further, we compare with a controlled random pruning baseline (Table 6). For fairness, the baseline prunes the same number of batches randomly as selected by our method (i.e., identical DSI). Across CIFAR-10 and CIFAR-100, our method consistently outperforms random pruning in both IID and Non-IID settings.

Table 6: Comparison of our and random pruning under IID and Non-IID settings for CIFAR-10 and CIFAR-100 in terms of Accuracy.

	CIFAR-10		CIFAR-100	
	IID	Non-IID	IID	Non-IID
Random	92.26	82.46	70.68	63.25
Our	93.16	85.48	73.28	68.16

Layer aggregation: An ablation study across 11 layer configurations shows minimal sensitivity: accuracy ranges only 0.68% (92.55%–93.23%) across all configurations. Notably, early layers (first two layers: 93.23%) outperform deeper layers (last two layers: 92.78%) and all layers combined (93.16%). This contradicts the concern that averaging hides deep-layer instability, if early-layer stabilization were masking useful gradients from still-learning deeper layers, excluding early layers should perform better, yet we observe the opposite. The consistent performance across configurations indicates that batch stability is an intrinsic sample property rather than layer-specific. Tabular details in Appendix A.3.

Additional empirical observations. Further analyses are provided in Appendix A.3. Learning rate influences pruning behavior: larger step sizes increase gradient variability, leading to more aggressive pruning and reduced accuracy, while moderate rates stabilize activation dynamics and yield more reliable pruning. Consistent with B-PAS Alam et al. (2026), normalization is critical: batch normalization smooths activation trajectories, enabling higher data savings with preserved accuracy, whereas removing normalization destabilizes activations and limits pruning (though higher savings can still be achieved with aggressive δ). The method remains robust across optimizers (SGD, Adam, AdaGrad, RMSProp), indicating insensitivity to update dynamics. We also verify that pruning does not induce class imbalance at either global or local levels, with class distributions remaining nearly unchanged after training. Appendix A.5 further includes results on stronger heterogeneity ($\alpha = 0.05$), batch assignment sensitivity, tracking overhead (1.2% wall-clock), and an energy estimate.

6 Conclusion

We presented data pruning at the batch level by activation stability in federated learning, the first framework to establish activation stability as a reliable, training-aware, purely on-device redundancy signal under decentralized, non-IID settings. By disentangling local representational convergence from global client drift, we showed why a centralized signal can be exported to FL without any cross-client coordination, server-held data, or architectural modification. Per-client persistent histories and a staleness-aware mechanism turn this insight into a drop-in plug-in for FL frameworks. Across four vision benchmarks and three backbones, the method removes up to 45% of training batches with almost no loss in accuracy, and sometimes improves it. It beats the strongest coreset baseline at equal data savings under high heterogeneity. Importantly, reducing data usage and GPU node-hours directly lowers energy consumption and makes large-scale training more accessible, highlighting a positive societal impact. A current limitation is the use of empirically chosen δ schedules; developing data-driven, adaptive thresholding mechanisms is an important direction for future work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we include the codebase, experimental settings, and instructions in the supplementary materials. The full code and documentation will be publicly released upon acceptance.

AI USE STATEMENT

Generative AI tools (LLMs and AI agents) were used to assist in manuscript preparation, including improving clarity, refining wording, and structuring sections, as well as supporting codebase organization. All AI-assisted outputs were carefully reviewed and verified by the authors.

REFERENCES

Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. Contextual diversity for active learning. In *Computer Vision–ECCV 2020: Proceedings*. Springer, 2020.

Md. Mustakin Alam, Tanjim Ahmed, Meraz Hossain, Mehedi Hasan Emo, Md. Kausar Islam Bidhan, Md. Tanzim Reza, Md. Golam Rabiul Alam, Mohammad Mehedi Hassan, Francesco Pupo, and Giancarlo Fortino. Federated ensemble-learning for transport mode detection in vehicular edge network. *Future Generation Computer Systems*, 149:89–104, 2023. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2023.07.022. URL https://www.sciencedirect.com/science/article/pii/S0167739X23002674.

Md Mustakin Alam, Shaker Islam, and Aminul Islam. Batch pruning by activation stability. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TUADW7db5n.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR Workshops*, 2022.

Prateek Chanda, Shrey Modi, and Ganesh Ramakrishnan. Bayesian coreset optimization for personalized federated learning, 2025. URL https://arxiv.org/abs/2511.01800.

Junming Chen, Meirui Jiang, Qi Dou, and Qifeng Chen. Federated domain generalization for image recognition via cross-client style transfer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 361–370, 2023.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint*, 2019. arXiv:1906.11829.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. In *arXiv preprint*, 2018. arXiv:1802.09841.

Christoph Düsing, Philipp Cimiano, and Benjamin Paaßen. Leveraging local data sampling strategies to improve federated learning. *International Journal of Data Science and Analytics*, 20(3):2659–2678, 2025.

Eros Fanì, Marco Ciccone, and Barbara Caputo. Feddrive v2: an analysis of the impact of label skewness in federated semantic segmentation for autonomous driving. *arXiv preprint arXiv:2309.13336*, 2023.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. PMLR Vol. 70.

Ziyao Guo et al. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *arXiv preprint*, 2024. arXiv:2310.05773.

Chenhe Hao, Weiying Xie, Daixun Li, Haonan Qin, Hangyu Ye, Leyuan Fang, and Yunsong Li. Fedcs: Coreset selection for federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15434–15443, June 2025.

Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. Large-scale dataset pruning with dynamic uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 7713–7722, June 2024.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Distributed pruning towards tiny neural networks in federated learning, 2023. URL https://arxiv.org/abs/2212.01977.

Hong Huang, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. Fedmef: Towards memory-efficient federated dynamic pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27548–27557, 2024.

Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, 2021. PMLR Vol. 132.

Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386, 2022.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *AAAI Conference on Artificial Intelligence*, 2021.

Nikolas Koutsoubis, Yasin Yilmaz, Ravi P Ramachandran, Matthew Schabath, and Ghulam Rasool. Privacy preserving federated learning in medical imaging with uncertainty estimation. *arXiv preprint arXiv:2406.12815*, 2024.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009a. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). 2009b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Wei Jiang, and Yang You. Dream: Efficient dataset distillation by representative matching. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17268–17278. IEEE, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Mark Huasong Meng, Sin G Teo, Guangdong Bai, Kailong Wang, and Jin Song Dong. Enhancing federated learning robustness using data-agnostic model pruning. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 441–453. Springer, 2023.

Mansheej Paul, Surya Ganguli, and Gintare K. Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, 2021.

Zhen Qin, Zhaomin Wu, Bingsheng He, and Shuiguang Deng. Federated data-efficient instruction tuning for large language models, 2025. URL https://arxiv.org/abs/2410.10926.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, Yang You, et al. Infobatch: Lossless training speed up by unbiased dynamic data pruning. In *The Twelfth International Conference on Learning Representations*, 2024.

Ravi S. Raju, Kyle Daruwalla, and Mikko H. Lipasti. Accelerating deep learning with dynamic data pruning. *CoRR*, abs/2111.12621, 2021. URL https://arxiv.org/abs/2111.12621.

Angelo Rodio and Giovanni Neglia. Fedstale: leveraging stale updates in federated learning. In *ECAI 2024-27TH European Conference on Artificial Intelligence*, volume 392, pp. 3071–3078, 2024.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Advances in Neural Information Processing Systems*, 2018.

Durga Sivasubramanian, Lokesh Nagalapatti, Rishabh Iyer, and Ganesh Ramakrishnan. Gradient coreset for federated learning, 2024. URL https://arxiv.org/abs/2401.06989.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint*, 2018. arXiv:1812.05159.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Yang You, et al. Cafe: Learning to condense dataset by aligning features. In *Proceedings of CVPR*, 2022.

Shuyue Wei, Yongxin Tong, Zimu Zhou, Tianran He, and Yi Xu. Efficient data valuation approximation in federated learning: A sampling-based approach, 2025. URL https://arxiv.org/abs/2504.16668.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 22–31, 2021.

Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, Tongliang Liu, et al. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Hong Zhang, Ji Liu, Juncheng Jia, Yang Zhou, Huaiyu Dai, and Dejing Dou. Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server, 2022. URL https://arxiv.org/abs/2204.11536.

Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coreset selection for high pruning rates. *arXiv preprint*, 2023. arXiv:2210.15809.

A Appendix

A.1 Experimental Setup

A.1.1 Models

We evaluate both convolutional and transformer-based vision models.

ResNet architectures. We adopt standard ResNet-18 and ResNet-50 backbones with residual connections. ResNet-18 uses `BasicBlock` units (expansion factor 1) with configuration $[2, 2, 2, 2]$, while ResNet-50 employs `Bottleneck` units (expansion factor 4) with configuration $[3, 4, 6, 3]$. Both models consist of an initial 7×7 convolution and max pooling, followed by four residual stages, global average pooling, dropout (0.2), and a fully connected classifier. Batch Normalization is applied after each convolution, with ReLU activations throughout. For activation stability, we track outputs from each residual stage (layer1–layer4).

CvT architecture. We additionally evaluate CvT-13 (Wu et al., 2021), which integrates convolutional projections into transformer blocks. The model comprises three stages (Table 7) with embedding dimensions $[64, 192, 384]$, depths $[1, 2, 10]$, and attention heads $[1, 3, 6]$, respectively. Each stage includes convolutional embedding, convolutional multi-head self-attention, and MLP blocks with GELU activation. Activations are collected at stage outputs and reshaped into spatial form (B, C, H, W) for variance computation. The model concludes with layer normalization, global average pooling, and a linear classifier.

A.1.2 DATASETS

We conduct experiments on CIFAR-10, CIFAR-100, SVHN, and ImageNet-1K. For CIFAR-10/100 and SVHN, we create a validation split of 5,000 samples from the original training set, and use the original test set for testin. For ImageNet-1K, we reserve 10,000 samples from the training set for validation and 50,300 samples from the training set for testing. Standard data augmentations are applied, including normalization, random cropping, and horizontal flipping.

Table 7: Model specifications.

Model	Block Type	Depth	Channels / Embedding
ResNet-18	BasicBlock	[2, 2, 2, 2]	[64, 128, 256, 512]
ResNet-50	Bottleneck	[3, 4, 6, 3]	[256, 512, 1024, 2048]
CvT-13	Conv-Attn + MLP	[1, 2, 10]	[64, 192, 384]

Table 8: Dataset splits used in our experiments.

Dataset	Training	Validation	Test
CIFAR-10	45,000	5,000	10,000
CIFAR-100	45,000	5,000	10,000
SVHN	68,257	5,000	26,032
ImageNet-1K	1,220,867	10,000	50,300

A.1.3 TRAINING DETAILS

Table 9 summarizes hyperparameters across datasets and models. Unless otherwise specified, CNNs are trained using SGD with momentum 0.9. CIFAR and SVHN use cosine annealing, while ImageNet with ResNets adopts `MultiStepLR`. CvT models are optimized with AdamW.

For ImageNet, training is distributed across 4 GPUs, with linear scaling of batch size and learning rate (total batch size 256×4, base learning rate 0.4).

Table 9: Training hyperparameters.

Dataset / Model	Rounds	Batch	LR	WD	Scheduler
CIFAR-10 (ResNets)	200	128	0.05	5×10^{-4}	Cosine
CIFAR-100 (ResNets)	200	128	0.10	5×10^{-4}	Cosine
SVHN (ResNets)	200	128	0.10	5×10^{-4}	Cosine
ImageNet (ResNets)	300	256×4	0.1×4	1×10^{-4}	MultiStep
ImageNet (CvT)	100	128×4	1×10^{-3}	0.05	Cosine

A.1.4 HARDWARE

Experiments on CIFAR-10/100 and SVHN are conducted on a workstation with an AMD Ryzen 9 CPU, NVIDIA Titan RTX GPU (24GB), and 128GB RAM. ImageNet experiments are performed on a cluster with dual Intel Xeon Platinum CPUs and 4× NVIDIA A100 GPUs with NVLink, supported by 512GB RAM.

Table 10: Hardware (CIFAR/SVHN).

CPU	AMD Ryzen 9 7900X
GPU	NVIDIA Titan RTX (24GB)
RAM	128 GB DDR5

Table 11: Hardware (ImageNet).

CPU	Dual Intel Xeon Platinum 8358
GPU	4× NVIDIA A100 (NVLink)
RAM	512 GB

A.1.5 SOFTWARE

All experiments are implemented in Python 3.8.18 using PyTorch and Torchvision. Supporting libraries include NumPy, Matplotlib, Scikit-learn, Seaborn, Pandas, and Pillow. CIFAR and SVHN experiments are conducted in Jupyter Notebook, while ImageNet experiments are executed as standalone Python scripts.

A.2 STATISTICAL TESTS FOR STABILITY UNDER HETEROGENEITY

To quantitatively assess whether activation stability varies across different heterogeneity levels, we perform two standard statistical tests: one-way ANOVA and the Kruskal–Wallis test.

Table 12: Software environment.

Component	Details
Python	3.8.18
Libraries	PyTorch, Torchvision, NumPy, etc.
Platform	Jupyter / Python scripts

Setup. For each Dirichlet coefficient $\alpha \in \{0.1, 0.5, 1.0\}$, we collect activation stability values across communication rounds (and/or probe batches), treating each α as a separate group. The goal is to test whether the distribution of stability differs across these groups.

One-way ANOVA. ANOVA (Analysis of Variance) tests the null hypothesis that the group means are equal: $H_0 : \mu_{\alpha=0.1} = \mu_{\alpha=0.5} = \mu_{\alpha=1.0}$. It computes an F-statistic by comparing variance *between* groups to variance *within* groups. A small p-value (typically < 0.05) indicates that at least one group mean differs significantly.

Kruskal–Wallis test. The Kruskal–Wallis test is a non-parametric alternative to ANOVA that does not assume normality. It ranks all samples across groups and tests whether the rank distributions differ. The null hypothesis is that all groups are drawn from the same distribution.

Table 13: Training ImageNet-1K with ResNet-50 under different learning rates.

LR	DSI(%)	Acc(%)
0.2	63	72.22
0.1	42	73.32

Results and interpretation. In our experiments, ANOVA yields $p = 0.93$ and Kruskal–Wallis yields $p = 0.40$. Both values are well above the standard significance threshold (0.05), and therefore we do not reject the null hypothesis in either test. This indicates that we do not observe statistically significant differences in activation stability across heterogeneity levels. This supports our observation that activation stability remains consistent across varying data distributions.

A.3 MORE RESULTS

Learning Rate. Table 13 shows that the learning rate influences pruning behavior. A high LR (=0.2) amplifies gradient variability, causing aggressive pruning (DSI 63%) and a sharp drop in accuracy (72.22%). Reduced LR (0.1) stabilizes activation dynamics, yielding a more controlled pruning rate (DSI 42%) and higher accuracy (73.32%). This indicates that large updates can destabilize activation statistics and trigger premature batch removal, whereas moderate learning rates support more reliable pruning.

Layer Ablation: We conducted an ablation study across 11 layer configurations (Table 14). The results show minimal sensitivity: the precision ranges only 0.68% (92.55%–93.23%) across all configurations. In particular, early layers (first two layers: 93.23%) outperform deeper layers (last two layers: 92.78%) and all layers combined (93.16%). This contradicts the concern that averaging hides deep-layer instability, if early-layer stabilization were

Table 15: Effect of training iterations on ImageNet-1K.

Iter	DSI(%)	Acc(%)
100	5	66.23
300	42	73.32

Table 14: Effect of tracked layer configurations on performance and training efficiency. Results show minimal sensitivity to layer choice, with consistent accuracy across configurations.

Config	#Layers	Description	Acc (%)
all_layers	4	All residual layers (default)	92.75
last_3_layers	3	Last 3 layers (deep features)	92.72
last_2_layers	2	Last 2 layers (deepest features)	92.78
last_layer_only	1	Final layer only	92.79
first_3_layers	3	First 3 layers (early features)	92.95
first_2_layers	2	First 2 layers (earliest features)	93.23
first_layer_only	1	First layer only	92.55
alternating_13	2	Layers 1 & 3	92.76
alternating_24	2	Layers 2 & 4	92.99
middle_layers	2	Middle layers only	93.01
all_with_conv1	5	All layers + conv1	93.16

masking useful gradients from still-learning deeper layers, excluding early layers should perform better, yet we observe the opposite. The consistent performance across configurations indicates that batch stability is an intrinsic sample property rather than a layer-specific property, and averaging does not cause premature pruning.

Effect of training iterations. As shown in Table 15, shorter training schedules (100 iterations) offer minimal pruning opportunity, yielding low DSI and reduced accuracy due to insufficient representation learning.

Extending training to 300 iterations increases both pruning effectiveness (DSI=42%) and accuracy (73.32%), indicating that pruning benefits from mature activation patterns that emerge over longer optimization trajectories. This confirms that pruning requires sufficient representational evolution to reliably detect saturation.

Table 16: Effect of Batch Normalization (BN) in CIFAR-10 with ResNet-18.

	Full Dataset		Pruning	
	Acc (%)	DSI (%)	Acc (%)	DSI (%)
Without BN	82.57	0	82.73	5
With BN	93.49	0	93.16	16

Table 17: Class distribution across all clients (CIFAR-10, ResNet-18).

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Initial (%)	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
After 100 rounds (%)	9.95	10.14	9.94	9.99	9.99	10.07	10.00	10.00	10.03	9.89

Effect of Batch Normalization. Table 16 demonstrates the importance of Batch Normalization (BN) for activation-stability pruning. Without BN, unstable activation dynamics limit effective pruning, resulting in only 5% DSI and substantially lower accuracy. Incorporating BN regularizes activation statistics, enabling more reliable stability estimation and significantly higher pruning efficiency (16% DSI) while maintaining strong performance (93.16% accuracy). These results indicate that normalized feature distributions are important for robust and effective stability-based pruning.

Performance with Different Optimizers. Table 18 shows that our method remains stable across SGD, Adam, AdaGrad, and RMSProp. Although these optimizers have different update dynamics and curvature sensitivities, pruning accuracy remains consistent with the full-dataset baseline, and DSI values remain in a narrow range. This confirms that activation-stability pruning is optimizer-agnostic and resilient to differences in gradient noise, adaptivity, and step size scaling.

Table 18: The approach preserves accuracy across optimizers on CIFAR-10 with ResNet-18, demonstrating robustness to optimization dynamics.

	SGD	Adam	AdaGrad	RMSProp
Full Dataset	93.49	91.55	88.35	87.82
Our	93.16	91.27	86.72	86.79
DSI (%)	16	14	21	22

Class Distribution: Table 17 shows that the aggregate class distribution remains stable after 100 rounds of pruning, with per-class proportions deviating negligibly (9.89%–10.14% vs. the initial 10%).

To verify that this balance holds at the *local* level, where Dirichlet heterogeneity ($\alpha = 0.1$) creates highly skewed client distributions, we measure the Jensen–Shannon Divergence (JSD) between each client's original and post-pruning class distributions. JSD quantifies the similarity between two probability distributions by computing the average KL divergence of each distribution from their mixture: $\mathrm{JSD}(P\|Q) = \frac{1}{2}\mathrm{KL}(P\|M) + \frac{1}{2}\mathrm{KL}(Q\|M)$, where $M = \frac{1}{2}(P+Q)$. Unlike KL divergence, JSD is symmetric and bounded in $[0, 1]$, where 0 indicates identical distributions and values closer to 1 indicate complete divergence. In our analysis, P is a client's original class distribution and Q is its class distribution after pruning; a low JSD therefore confirms that pruning did not alter the relative class proportions within that client's data. Across all clients, the mean JSD is 0.055 (max 0.298), confirming that pruning preserves each client's local class proportions despite removing 84% of all batches. The maximum JSD is attributable to clients with extremely small data allocations under Dirichlet sampling (e.g., 108 samples), where the removal of even a few batches causes a proportionally larger distributional shift. For clients with moderate to large allocations, JSD remains below 0.06. This is expected: since batches are constructed by random assignment of samples, each batch contains a representative mix of the client's local classes. Activation stability reflects representational convergence of the batch as a whole rather than the frequency of any individual label, causing redundant batches to be removed class-agnostically without introducing systematic imbalance at either the local or global level.

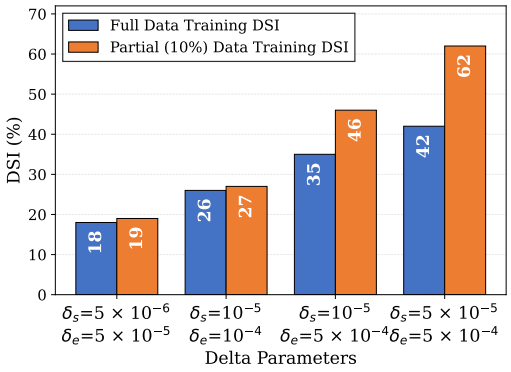


Figure 5: Comparison of DSI between full-data training and partial (10%) data runs on ImageNet-1K (ResNet-50). The consistent ordering across thresholds demonstrates that small subsets reliably capture pruning behavior, enabling efficient δ calibration.

Empirical Analysis on CIFAR-10 under IID and Non-IID Distributions. We evaluate the proposed pruning mechanism on CIFAR-10 using ResNet-18 across ten distinct threshold configurations for both IID and Non-IID settings. As shown in Figure 7 and Figure 8, the threshold range plays a critical role in balancing pruning aggressiveness and accuracy. With very small thresholds, pruning is conservative: accuracy remains high, but data savings (DSI) remain low because most batches are retained.

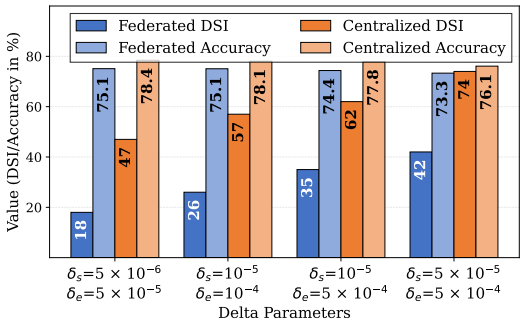


Figure 6: Comparison of DSI and accuracy between centralized (B-PAS Alam et al. (2026)) and federated training across four threshold settings in ImageNet-1K.

Conversely, larger thresholds trigger aggressive pruning, leading to premature removal of informative batches and, in many cases, early termination of training. This behavior yields high DSI but significantly degrades accuracy. For example, experiments eight through ten achieve competitive accuracy yet exhibit low data savings, reflecting minimal pruning.

In contrast, experiments one through three prune heavily beginning at early epochs, resulting in substantial data reduction but sharp accuracy drops. Across both IID and Non-IID configurations, the most stable and well-balanced behavior emerges within the range $\delta \in [10^{-7}, 10^{-3}]$. This configuration consistently achieves strong accuracy while maintaining DSI values near 15%, indicating effective pruning without sacrificing learning quality.

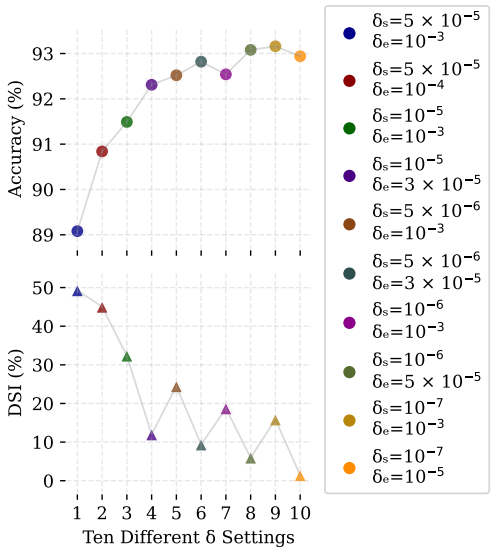


Figure 7: CIFAR-10 (IID): Effect of pruning thresholds on accuracy and data savings. Smaller thresholds retain most batches and preserve accuracy, while larger thresholds prune aggressively and reduce performance. Balanced behavior appears within $\delta \in [10^{-7}, 10^{-3}]$.

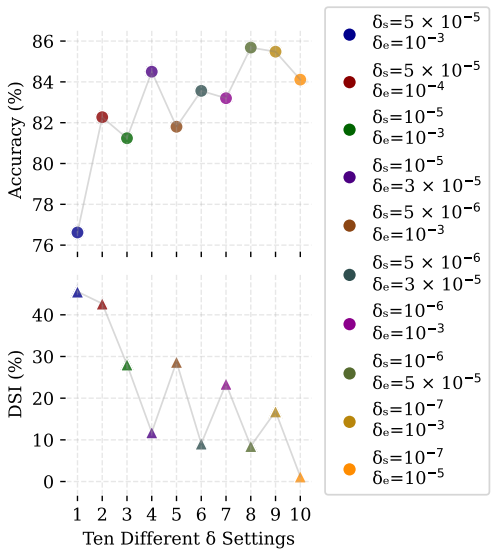


Figure 8: CIFAR-10 (Non-IID): Effect of pruning thresholds on accuracy and data savings. Smaller thresholds retain most batches and preserve accuracy, while larger thresholds prune aggressively and reduce performance.

These results highlight a clear relationship between pruning thresholds, accuracy, and data efficiency. For CIFAR-10 and other 32×32 image datasets, this threshold interval serves as a reliable default choice, as further demonstrated in Table 3.

Comparison of Centralized and Federated Pruning. Figure 6 compares the data savings index (DSI) and accuracy between centralized (B-PAS Alam et al. (2026)) and federated training for four ascending δ settings. Centralized training consistently achieves higher DSI, as global gradients and uniformly mixed data lead to faster stabilization of activation statistics, enabling earlier and more aggressive pruning. In contrast, federated training shows lower DSI because heterogeneous client data, asynchronous local updates, and round-based synchronization slow the convergence of activation variance. As a result, identical (δ_s, δ_e) schedules prune substantially fewer batches in FL, making federated pruning inherently more conservative. Although accuracy in FL is also lower, consistent with known gaps between centralized and decentralized optimization, the relationship between pruning aggressiveness and accuracy degradation remains similar across both settings: higher DSI generally leads to reduced accuracy.

A.4 Fast δ calibration with small data subsets

The pruning threshold δ can be calibrated efficiently without full-scale training. Consistent with prior findings in B-PAS Alam et al. (2026), we observe that the relationship between δ, data saving index (DSI), and accuracy is preserved even when using only a small subset (10%) of the training data (Figure 5). The relative ordering of pruning strength remains consistent, with larger thresholds yielding more aggressive pruning and smaller thresholds producing conservative behavior, closely matching full-data trends.

This consistency enables reliable δ selection using lightweight partial runs, significantly reducing tuning cost. Moreover, once a suitable threshold schedule is identified for a dataset family, it transfers effectively across related datasets and architectures (e.g., CIFAR-10 to CIFAR-100/SVHN across CNN backbones). In practice, tuning δ requires effort comparable to standard hyperparameters such as learning rate or weight decay, and does not introduce a significant overhead.

A.5 Additional Experiments

This section reports further experiments on the staleness mechanism, scale, heterogeneity, the choice of pruning criterion, and cost. Unless stated otherwise, all runs use CIFAR-10 with ResNet-18, 10 clients, Dirichlet $\alpha = 0.1$, and 200 communication rounds.

Table 19: Ablation of the staleness-aware mechanism. "Without" keeps a persistent history with no threshold scaling or reset, so k has no effect and both 50% settings share one run. Accuracy (%) with DSI (%) in parentheses.

Participation, k	With	Without
75%, $k=5$	86.95 (19.2)	83.26 (16.8)
50%, $k=5$	84.16 (12.3)	78.34 (11.2)
50%, $k=20$	80.83 (11.5)	

Table 20: Pruning accuracy as the number of clients grows and per-round participation falls.

Clients	Participation	Acc (%)
10	50%	84.16
20	25%	60.97
50	10%	54.63

Staleness-aware mechanism. To isolate the effect of the staleness rule, we replace it with a naive persistent history that always compares against the last stored scores. Table 19 shows that the mechanism raises accuracy by 3.7 points at 75% participation and by 5.8 points at 50%, while also pruning slightly more data. The gain is smaller with $k = 20$ (2.5 points), since a large k rarely triggers the stricter threshold or the reset. Comparing scores across distant global models therefore leads to poor pruning decisions, and correcting for staleness is needed in FL.

Client scale. Table 20 increases the number of clients from 10 to 50 while lowering participation from 50% to 10%. Accuracy drops as the same training set is split across more clients under a fixed budget of 200 rounds, so each client trains less often. Pruning still removes 10–14% of the data in every setting. Under sparse participation, clients often return after more than k rounds; the reset then skips pruning for that round, so the method becomes more conservative instead of acting on stale scores.

Stronger heterogeneity. With $\alpha = 0.05$ and full participation, pruning reaches 77.18% against 76.94% for full-data training while removing 22.2% of the data. The method remains stable under heterogeneity well beyond the $\alpha = 0.1$ used in the main experiments.

Input-space similarity baseline. We test whether a data-space score could replace the forward-pass signal. The baseline removes the same number of batches in the same rounds as our method, but selects them by input-space cosine redundancy. Table 21 shows it comes close to ours and clearly beats random pruning. This suggests that, once the schedule is fixed, a reasonable redundancy score recovers most of the benefit, since each batch is a random mix of the client's samples. The schedule itself, deciding *when* and *how much* to prune, is what activation stability provides. A static similarity score has no notion of training progress and cannot produce this schedule; here it inherits it from our method.

Table 21: Pruning criteria at a matched number of pruned batches (Non-IID CIFAR-10, full participation).

Criterion	Acc (%)
Random	82.46
Input-space similarity	85.14
Activation stability (ours)	85.48

Sensitivity to batch assignment. Since batches are fixed at initialization, we repeat the Non-IID CIFAR-10 experiment with three independent random batch assignments. Final accuracy varies by ± 0.54 points (maximum spread 1.1 points), so the results do not depend on a particular batching.

When pruning happens. The first batch is pruned around round 80, and the median pruned batch is removed at round 161 of 187. Most pruning thus occurs in the last third of training, after most of the learning has taken place, which limits the risk of permanently removing batches that are still useful.

Tracking overhead. Computing the per-layer standard deviation is a single $O(N)$ reduction over an activation tensor that the forward pass already produces. With tracking on and off, a CIFAR-10 round takes 23.48 s and 23.21 s, an overhead of 1.2%. This cost shrinks as batches are pruned, and all reported node-hours already include it.

Energy estimate. We did not measure power directly, so we convert node-hours into an estimate. Each ImageNet node has 4×A100 GPUs at 400 W TDP (1.6 kW), and we assume a datacenter PUE of 1.4, giving energy = node-hours × 1.6 kW × 1.4. Full-data training in Table 1 then uses about 302 kWh, and the most aggressive schedule $[10^{-4}, 10^{-3}]$ uses about 165 kWh, saving roughly 137 kWh per run. Across all schedules, savings range from 38 to 137 kWh (12.6–45.3%). This is a TDP-based upper bound that excludes host CPU and memory, and real draw varies with utilization.