DECENTRALIZED SPLIT FEDERATED LEARNING WITH ASYNCHRONOUS ROTATING COMMITTEES

Anonymous authors
Paper under double-blind review

ABSTRACT

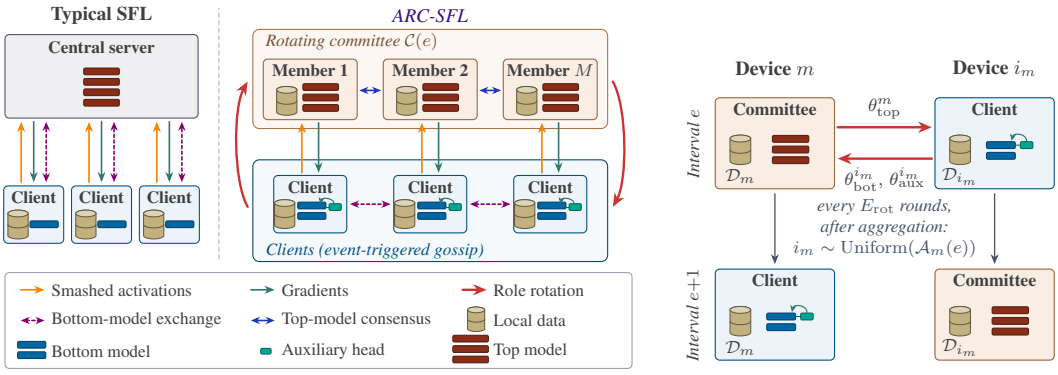
Split Federated Learning (SFL) enables collaborative training of machine learning models on resource-constrained IoT devices by combining model splitting with federated optimization. Despite its promise, existing SFL systems fundamentally lack scalability due to centralized server bottlenecks, synchronous training barriers, and reliance on fixed infrastructure, that are major limiting factors in IoT environments. We propose *ARC-SFL*, a fully asynchronous and decentralized framework for SFL that eliminates permanent servers while preserving accuracy and efficiency benefits. *ARC-SFL* distributes server-side computation across replicated committee members via our load balancing scheme, and supports asynchronous aggregation to remove straggler blocking. To further reduce communication overhead, *ARC-SFL* employs event-triggered gossip, enabling peer-to-peer information exchange only when local training has made progress and the learned representation has drifted. Across multiple vision, medical, and audio benchmarks, and on real homogeneous and heterogeneous clusters of IoT devices, we show that *ARC-SFL*: (i) scales linearly with the number of devices, reaching an order of magnitude higher throughput than synchronous SFL at 100 devices, while remaining robust to device and committee failures; (ii) significantly outperforms decentralized gossip learning on all six tasks by up to 31.2% and, without any centralized server, remains on par with the best centralized task-dependent baselines; and (iii) speeds up the convergence by $\approx 3.5\times$ and reduces gossip communication by up to 55% without degrading accuracy on compact models.

1 INTRODUCTION

The growing number of Internet of Things (IoT) and tiny edge devices typically generate a huge amount of data, often in privacy-sensitive domains, such as healthcare, finance, and smart homes. However, privacy regulations such as GDPR (European Parliament and Council of the European Union, 2016), stringent non-disclosure agreements (NDAs), and customer data protection policies, prohibit the transfer of raw data for training. Beyond regulatory constraints, enterprises often restrict proprietary data sharing even across internal divisions due to competitive and legal concerns (Lin et al., 2020c; Yin et al., 2022). Federated Learning (FL) addresses these restrictions by collaboratively training models across distributed devices while keeping all raw data local and only exchange model updates instead of the raw data (McMahan et al., 2017; Kairouz et al., 2021). Split Learning (SL) extends this idea with resource-constrained devices by deploying part of the model on client and servers where clients share only intermediate representations, creating an information bottleneck that not only splits work but also protects sensitive data (Gupta & Raskar, 2018; Vepakomma et al., 2018). Split Federated Learning (SFL) combines both (Thapa et al., 2022; Wu et al., 2023), enabling collaboration among many clients with split models, which supports training larger models under privacy and resource constraints and in the presence of non-IID data (Han et al., 2024).

Motivation. Existing SFL methods suffer from several shortcomings. Most approaches rely on a high performance and highly available central entity, e.g., a parameter server, to coordinate participating devices, manage model execution, and aggregate updates (Thapa et al., 2022; Wu et al., 2023; Oh et al., 2022; Liao et al., 2024). This introduces a single point of failure and assumes the availability of a fixed infrastructure that is often unrealistic in edge and IoT environments. In addition, many SFL methods adopt synchronous training protocols (Thapa et al., 2022; Wu et al., 2023; Shen et al., 2024), making overall progress sensitive to slow or straggling participants (Chai et al., 2020;

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.



(a) Typical SFL vs. *ARC-SFL*

(b) *ARC-SFL* Rotation of Committee.

Figure 1: *ARC-SFL* Overview

Wu et al., 2021), while significantly degrading training efficiency and increasing communication overhead. Decentralized FL removes the server but trains the full model on every device (Lian et al., 2017; Koloskova et al., 2020), and the few serverless SFL approaches rely on synchronous, communication-intensive consensus (Tedeschini et al., 2024).

Approach. We present *ARC-SFL*, a novel fully decentralized, asynchronous method that eliminates the need for a permanent server. We categorize tiny edge devices, which typically share similar computational characteristics, into clients or committee members. We then split a model between a client and a committee. We refer to the first split residing on the client as the *bottom-model* and the second split as the *top-model*. As illustrated in Fig. 1, we introduce a *load balancing mechanism* through the committee to distribute the top-model workload across multiple devices, and periodically rotate the committee so that every device eventually serves in it. To mitigate the straggler-blocking effect and make use of idle client time, we employ a *cycle stealing* approach, combining asynchronous aggregation with weighted averaging that accounts for model staleness (Zhang et al., 2016; Xie et al., 2019). Further, we enable clients to continue updating their local bottom models concurrently while awaiting committee members' responses or slower peers (Han et al., 2021; Belilovsky et al., 2020) with the help of an auxiliary local head. Lastly, we reduce communication via event-triggered gossip (George & Gurram, 2020; Zehtabi et al., 2022), in which clients share their models only when local progress is significant and representation drift exceeds a threshold.

Contribution. ① To our knowledge, *ARC-SFL* is the first serverless SFL method for tiny devices that trains fully asynchronously. ② we implement our method in simulation, and homogeneous as well as heterogeneous clusters of real devices, a significant step towards realizing fully decentralized SFL for the real-world use cases; ③ *ARC-SFL* is scalable and robust; it handles large client populations, stragglers, and node or link failures. We additionally provide analysis and ablation study to explain the impact of different conditions on performance; and ④ we employ event-triggered, gossip-based information exchange among participating devices, further reducing communication overhead.

In summary, *ARC-SFL* removes the server without giving up accuracy. Across six benchmarks, we not only outperform decentralized gossip learning by 3 to 31 points and but also reach to close proximity of the best centralized baselines, even though such a baseline has a global view of all updates that no decentralized method can have. At the same time, *ARC-SFL* reaches 11× the throughput of synchronous SFL at 100 devices, keeps training unaffected when 30% of the clients drop out, and reduces gossip traffic by up to 55%.

2 Background and Related Work

Decentralized Federated Learning. In FL, each client trains a model on its private data and shares only model updates with a central server for aggregation (McMahan et al., 2017; Konečný et al., 2015; Li et al., 2020b). The central server, however, is a bottleneck and a single point of failure, particularly in IoT environments prone to device and communication failures. Decentralized FL (DFL) therefore lets clients exchange updates directly with their peers (Lalitha et al., 2019; Lian et al., 2017; Koloskova

et al., 2020; Hegedűs et al., 2021), and asynchronous variants further remove the synchronization barrier, at the cost of stale updates that must be down-weighted during aggregation (Lian et al., 2018; Xie et al., 2019; Nguyen et al., 2022). In both cases, however, every device trains the full model.

Split Federated Learning. Models grow faster than the compute and memory resources of tiny IoT devices. Split Learning (SL) therefore divides the model into a client-side part that runs on the device and a server-side part that resides on a more capable server (Gupta & Raskar, 2018; Vepakomma et al., 2018). In each step, the client sends intermediate activations to the server and receives the corresponding gradients. Thapa et al. (2022) combine FL and SL into Split Federated Learning (SFL). In SFLv1, the server processes the server-side models of all clients in parallel, whereas SFLv2 processes the clients sequentially. Han et al. (2021) update the client-side model with a local loss, and RingSFL (Shen et al., 2024) arranges clients in a ring. Parallel variants reduce the latency of serving many clients by clustering or parallelizing them (Wu et al., 2023; Oh et al., 2022; Lin et al., 2024; Liao et al., 2024), yet all of these methods rely on a central server for the top model or for aggregation. To our knowledge, the only serverless SFL method, SCFL (Tedeschini et al., 2024), requires communication-intensive, synchronous, and sequential updates and lacks mechanisms for robustness against failures.

Distributed ML on Tiny Devices. Several works reduce the load on individual devices through computation optimization and split computing (Gao et al., 2022; Jin & Hong, 2019; Asadi & Goudarzi, 2024; Katebi et al., 2024; Lin et al., 2022), while others adapt FL to IoT devices and microcontrollers (Wulfert et al., 2023a;b; Asadi et al., 2025; Giménez et al., 2023; Kopparapu et al., 2022). RockNet (Gräfe et al., 2026) enables microcontrollers to collaboratively train time-series classifiers within their memory constraints. However, it shares raw data across devices and supports only linear layers. Appendix G discusses further related work.

What makes *ARC-SFL* different? As summarized in Table 1, while providing the aforementioned benefits, existing SFL methods often rely on central servers, which raises security concerns and introduces potential single points of failure. Furthermore, to our knowledge, prior art deploys the second split on servers that are significantly more capable; this contradicts the reality of decentralized learning scenarios, which typically comprise only IoT devices with similar capabilities. In contrast, our method, *ARC-SFL*, addresses these limitations by proposing an asynchronous rotating committee-based approach, as we describe next.

Table 1: Comparison of SFL frameworks (Thapa et al., 2022; Han et al., 2021; Shen et al., 2024; Arafeh et al., 2025; Tedeschini et al., 2024).

Method	Topol.	Server-less	Async	Strag. robust	Aux. loss	Role rot.
SplitFed	Star	×	×	×	×	×
Han et al.	Star	×	×	×	✓	×
RingSFL	Ring	×	×	✓	×	×
ClusterSFL	Cluster	×	✓	✓	×	×
SCFL	P2P	✓	×	×	×	×
ARC-SFL	P2P	✓	✓	✓	✓	✓

3 METHODOLOGY

In this section, we first define the system model and the SFL problem, and then explain the individual components of *ARC-SFL*, namely committee-based execution of the top model, asynchronous aggregation, and event-based communication, followed by a convergence analysis.

3.1 *ARC-SFL* OVERVIEW

We consider an SFL setting with clients $\mathcal{C}_{\text{all}}=\{1,\dots,N\}$, where client i owns a private local dataset $\mathcal{D}_i$ containing n_i samples. We define $p_i=n_i/n$, where $n=\sum_{i=1}^{N} n_i$. The global target objective is

$$\min_{\theta_{\text{bot}},\theta_{\text{top}}} F(\theta_{\text{bot}},\theta_{\text{top}}) := \sum_{i=1}^{N} p_i \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\left[\ell\left(f_{\theta_{\text{top}}}(f_{\theta_{\text{bot}}}(x)),y\right)\right]. \tag{1}$$

The parameters θ_{bot} and θ_{top} in Eq. (1) denote the shared target parameters. During execution, *ARC-SFL* maintains distributed replicas of these parameters. Client i stores a local bottom-model replica θ^i_{bot}, while each committee member m stores a local top-model replica θ^m_{top}. For an input x, client i computes the smashed activation $h_i=f_{\theta^i_{\text{bot}}}(x)$, and its assigned committee member evaluates the corresponding top-model replica. Client-side gossip and committee synchronization are coordination

mechanisms that aim to keep these replicas sufficiently close while allowing asynchronous local progress. Each standard client additionally maintains a local auxiliary head θ^i_{aux}. The auxiliary head is an optimization mechanism that allows the bottom model to continue training while split learning gradients are pending; it is not part of the target objective in Eq. (1). To optimize it without a permanent central server, we distribute the server-side role across a rotating committee of clients. Committee members maintain local replicas of the top model and periodically align them through committee consensus, while the rest train local bottom-model replicas that exchange information through gossip. The client↔committee interactions are asynchronous, while top-model replicas are periodically aligned at committee synchronization events. Figure 1 shows an overview of *ARC-SFL* compared to typical SFL variants. Since clients and committee members are devices of similar capability, we split the model more evenly than typical SFL.

3.2 Committee Formation and Rotation

In *ARC-SFL*, our server functionality is decentralized with a dynamically rotating committee $\mathcal{C}(e)\subseteq\mathcal{C}_{\text{all}}$ of fixed size M at interval e. We distribute the load of θ_{top} across the committee members $m \in \mathcal{C}(e)$, which temporarily take the server role and host replicas of the top model. Clients get randomly selected to take the committee member role in order to balance the request load across committee members. An important aspect of *ARC-SFL* is that the committee rotates every E_{rot} committee rounds and devices switch roles over time. By changing the clients represented in the committee, we improve participation coverage, let the local data of every device eventually shape the top model, and avoid permanently assigning the server-side computation to the same devices. Figure 1b illustrates the rotation process. Once a rotation round arrives, for each committee member m, we randomly select one assigned client to swap roles with. Let $\mathcal{A}_m(e)$ denote the assigned clients of committee member m at interval e. It samples the successor as $\text{D}i_m \sim \text{Uniform}(\mathcal{A}_m(e))$. During this swap, the current committee member transfers its top model θ^m_{top} to the selected client i_m, while i_m transfers its bottom model $\theta^{i_m}_{\text{bot}}$ together with its auxiliary network $\theta^{i_m}_{\text{aux}}$ to the committee member. Devices exchange the models after aggregation. The former committee member, now acting as a regular client, continues training using its local data and the newly received bottom and auxiliary networks.

3.3 Asynchronous Training and Aggregation

In contrast to the existing SFL methods, we enable asynchronous client–committee training. As soon as client i finishes its forward pass in local round r_i, it sends the smashed activation $h_i^{(r_i)}=f_{\theta^i_{\text{bot}}}(x)$ to the assigned committee member through an asynchronous remote procedure call $\mathcal{R}(\cdot)$ and requests the gradient $G^{(r_i)}_{\text{top}}:=\partial\mathcal{L}_{\text{global}}/\partial h_i^{(r_i)}$. While waiting for the gradients from the top model to return, client i continues training its local bottom model θ^i_{bot} using an auxiliary head θ^i_{aux}. This auxiliary network $f_{\theta^i_{\text{aux}}}$ is a small prediction head attached to the bottom model, which predicts $\hat{y}$ from the smashed activation $h_i^{(r_i)}$. The client minimizes the auxiliary loss $\mathcal{L}^{(r_i)}_{\text{aux}}=\ell(\hat{y},y)$ locally to update both θ^i_{bot} and θ^i_{aux}. The auxiliary network is lightweight, containing only a fraction of the top model, and overhead remains negligible. When client i receives the requested gradient $G^{(r_i)}_{\text{top}}$, it backpropagates it through its bottom model and combines it with the auxiliary gradient in a single update. Once a client has processed its local batches it enters a *reactive training* phase, in which it keeps iterating over its data and updates only θ^i_{bot} and θ^i_{aux} through the auxiliary loss, without getting blocked by slower peers. The committee closes the round as soon as a fraction (ρ) of its assigned clients has finished the learning phase, which ends reactive training. ρ regulates the asynchrony admitted within a committee group. As $\rho\to0$ the round closes at once, so clients start the gossip phase and then advance to the next round independently of the other clients' progress, whereas at $\rho=1$ it closes after every assigned client has finished, turning the round boundary into a synchronization barrier at the pace of the slowest client. After reactive training, client i then enters the gossip phase, where it requests $(\theta^k_{\text{bot}}, r_k, s_k)$ from K randomly selected clients and serves requests of its peers when the event trigger allows (§ 3.4).

During aggregation, client i combines its local bottom model with the bottom models received from peers $\mathcal{K}_i$, each weighted by its discount $\phi_\beta(\tau_k)$ and by the total number of samples s_k that peer k has

Algorithm 1 *ARC-SFL* Client Training Loop. At round r_i; $\ell_{\text{aux}} = \ell\big(f_{\theta^i_{\text{aux}}}(h_i), y_i\big)$ and $J_i = \partial h_i / \partial \theta^i_{\text{bot}}$.

Init.: $\theta^i_{\text{bot}}, \theta^i_{\text{aux}}, r_i \leftarrow 0, s_i \leftarrow 0$
while not converged **do**
  for batch $(X_i, y_i) \in \mathcal{D}_i$ **do** ▷ **A: split**
    $h_i \leftarrow f_{\theta^i_{\text{bot}}}(X_i)$
    request $G_{\text{top}} \leftarrow \mathcal{R}(h_i, y_i, r_i)$ ▷ async
    $g_{\text{aux}} \leftarrow 0$
    while G_{top} not received **do** ▷ ≤ 1 pass
      $g_{\text{aux}} \mathrel{+}= \nabla_{(\theta^i_{\text{bot}}, \theta^i_{\text{aux}})} \ell_{\text{aux}}$
    end while
    $g \leftarrow \lambda_{\text{aux}}\, g_{\text{aux}} + J_i^\top G_{\text{top}}$ ▷ on arrival
    $(\theta^i_{\text{bot}}, \theta^i_{\text{aux}}) \mathrel{-}= \eta\, g$
    $s_i \mathrel{+}= |X_i|$
  end for
  while round open: train on $\mathcal{L}_{\text{aux}}$ ▷ **B: reactive**
  serve peers iff Alg. 3 is true ▷ **C: gossip**
  pull $\{(\theta^k_{\text{bot}}, r_k, s_k)\}_{k \in \mathcal{K}_i}$ from peers
  $\theta^i_{\text{bot}} \leftarrow$ Eq. (2); $r_i \leftarrow r_i + 1$
end while

Algorithm 2 Committee Node Operation

Init.: θ^m_{top}, round $R \leftarrow 0$, finished set $\mathcal{F} \leftarrow \emptyset$
while committee member active **do**
  if $(h_i^{(r_i)}, y_i, r_i)$ from i **then** ▷ **serve**
    $\tau_i \leftarrow \max(0, R - r_i)$
    $\mathcal{L}_{\text{global}} \leftarrow \ell\big(f_{\theta^m_{\text{top}}}(h_i^{(r_i)}), y_i\big)$
    $G^{(r_i)}_{\text{top}} \leftarrow \nabla_{h_i^{(r_i)}} \mathcal{L}_{\text{global}}$
    $\theta^m_{\text{top}} \leftarrow \theta^m_{\text{top}} - \eta_{\text{top}}\, \phi_\alpha(\tau_i)\, \nabla_{\theta^m_{\text{top}}} \mathcal{L}_{\text{global}}$
    return $G^{(r_i)}_{\text{top}}$ to client i
    if last batch of i: $\mathcal{F} \leftarrow \mathcal{F} \cup \{i\}$
    $R \leftarrow \max(R, r_i)$ ▷ fastest client
  end if
  if $|\mathcal{F}| \geq \rho\, |\mathcal{A}_m|$ **then** ▷ **round close**
    broadcast close; $\mathcal{F} \leftarrow \emptyset$
  end if
  if $R \bmod E_{\text{sync}} = 0$: Eq. (3) ▷ **consensus**
  if $R \bmod E_{\text{rot}} = 0$: swap roles ▷ **rotation**
end while

processed so far:

$$\theta^{i,(r_i+1)}_{\text{bot}} = \frac{w_{\text{bot}}\, \theta^{i,(r_i)}_{\text{bot}} + \sum_{k \in \mathcal{K}_i} w_k\, \theta^k_{\text{bot}}}{w_{\text{bot}} + \sum_{k \in \mathcal{K}_i} w_k}, \quad w_k = s_k\, \phi_\beta(\tau_k), \quad w_{\text{bot}} = s_i\, \phi_\beta(0) = s_i. \tag{2}$$

The local model is never stale relative to client i. This rule favors fresh, data-rich updates and strongly down-weights stale ones, which stabilizes training in the asynchronous setting (Zhang et al., 2016; Xie et al., 2019; Wang et al., 2022). Algorithm 1 summarizes the client-side training procedure in *ARC-SFL*.

To aggregate the committee members' top-models, we compute a participation-weighted average:

$$\theta^{(R+1)}_{\text{top}} = \frac{\sum_{m \in \mathcal{C}(e)} s_m\, \theta^m_{\text{top}}}{\sum_{m \in \mathcal{C}(e)} s_m}, \tag{3}$$

where s_m is the total number of smashed activations processed by member m. Every E_{sync} committee rounds, one designated member initiates the exchange. The members then exchange their replicas, and each one applies the same merge rule via Eq. (3). This avoids global barrier for consensus. In contrast to the client side, it is crucial that the top models remain in close consensus; otherwise, committee members would gradually specialize on their own assigned clients, and this drift would propagate to the bottom models that train against them. Algorithm 2 summarizes the operation of a committee member m, including round closing, synchronization, and rotation.

3.4 Event-Triggered Gossip Mechanism

We further improve communication efficiency by limiting peer-to-peer transmission of bottom-model updates to situations where we expect these updates to be informative (Hsieh et al., 2017; Chen et al., 2018; Zehtabi et al., 2022; Singh et al., 2023). Specifically, each client decides locally to share its model only when local training has made progress and the learned representations have changed sufficiently. We measure progress through the local loss history $\{\mathcal{L}_i\}$ of client i. To be robust to noise, the client compares the mean loss of its $\mathcal{N}$ rounds, $\bar{\mathcal{L}}_i^{\text{new}}$, to the historical mean $\bar{\mathcal{L}}_i^{\text{old}}$, and shares if the loss has not increased. Further, our representation drift criterion detects significant changes in the client-side feature space. For client i we denote by $\bar{h}_i^{(r_i)}$ the mean of the smashed activations, a compact summary of that feature space, and measure drift as its cosine distance to a reference activation $\hat{h}_i$, $d_i^{(r_i)} = 1 - \cos\big(\bar{h}_i^{(r_i)}, \hat{h}_i\big)$, where $\hat{h}_i$ is the mean activation at the last round in which

client i shared its model. Each client maintains a history of its own drift values and shares only when $d_i^{(r_i)} \geq Q_q(\mathcal{D}_i)$ for the q-th empirical quantile. Therefore, exchange criteria is $d_i^{(r_i)} \geq Q_q(\mathcal{D}_i)$ and $\bar{\mathcal{L}}_i^{\text{new}} \leq \bar{\mathcal{L}}_i^{\text{old}}$. To avoid prolonged silence, we fallback on sharing after $N_{\max}$ silent local rounds, and the loss check is skipped during a short warm-up in which the loss history is still too short. Algorithm 3 in Appendix B gives the complete decision process.

3.5 Convergence and Complexity

We provide a conditional stationarity analysis for the shared SFL objective F in Eq. (1). We analyze the averaged iterate

$$\bar{w}_t = \left(\sum_{i=1}^{N} p_i \theta_{\text{bot},t}^i,\ \frac{1}{|\mathcal{C}(t)|} \sum_{m \in \mathcal{C}(t)} \theta_{\text{top},t}^m \right),$$

and let D_t denote the corresponding replica disagreement. Our analysis explicitly accounts for stale split gradients, auxiliary updates, and the potentially non-mean-preserving drift induced by decentralized coordination.

Theorem 3.1 (Conditional stationarity bound)**.** *Assume that F is lower bounded and smooth, stochastic-gradient variance and split-gradient delays are bounded, and that the stale-gradient bias is controlled by replica disagreement and a finite residual term. Further assume bounded auxiliary-update perturbations and that the coordination-induced average drift satisfies $\mathbb{E}[|\delta_t^{\text{coord}}|^2 \mid \mathcal{F}_t] \leq \eta^2(C_{\text{coord}} D_t + \varepsilon_{\text{coord}})$. Then, for a sufficiently small constant learning rate η, there exist constants $C_0, C_D, C_\tau, C_{\text{aux}}, C_{\text{coord}}, C_\sigma > 0$ such that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(\bar{w}_t)\|^2 \leq \frac{C_0}{\eta T} + \frac{C_D}{T} \sum_{t=0}^{T-1} \mathbb{E}[D_t] + C_\tau \varepsilon_\tau + C_{\text{aux}} \varepsilon_{\text{aux}} + C_{\text{coord}} \varepsilon_{\text{coord}} + C_\sigma \eta \sigma^2. \quad (4)$$

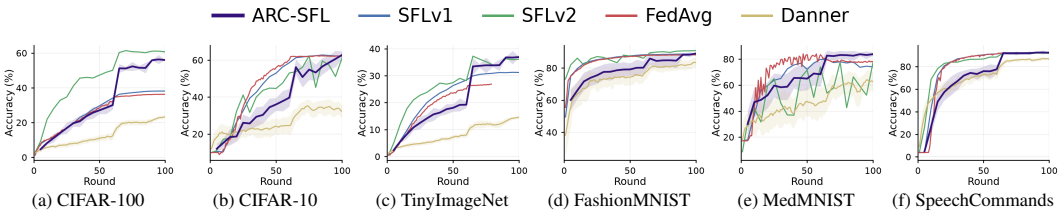
Thus, when replica disagreement and the residual errors caused by staleness, auxiliary training, and decentralized coordination remain controlled, the averaged model approaches a neighborhood of stationarity. The result is a conditional characterization rather than a convergence guarantee for arbitrary asynchronous executions; the precise assumptions, effective update model, and proof are given in Appendix C.

Under balanced assignment, each committee member handles approximately an $O(1/M)$ fraction of the online top-model workload of a centralized split learning server; Appendix C.8 provides the full complexity accounting.

4 Evaluation

Setup. We evaluate different methods on six benchmarks, covering vision, audio, and medical tasks on tiny devices, namely CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), TinyImageNet (Deng et al., 2009; Le & Yang, 2015), FashionMNIST (Xiao et al., 2017), MedMNIST (Yang et al., 2023), and Speech Commands (Warden, 2018). We pair them with ResNet-18 (He et al., 2016), MobileNetV2 (Sandler et al., 2018), EfficientNetB0 (Tan & Le, 2019), and FashionCNN, a lightweight two-layer convolutional network for FashionMNIST. Unless otherwise stated, we train all models from scratch on 50 devices for 100 rounds with Dirichlet label skew of concentration 0.5 (Hsu et al., 2019), a committee of three members and with K=16 gossip peers in every round; we study each separately in § 4.1. Appendix A provides further details.

Baselines. Since no decentralized SFL method is publicly available, we compare *ARC-SFL* with its natural decentralized competitor, gossip learning, and with three centralized methods. The latter aggregate all updates at a server with a complete and up-to-date view of the system; they therefore serve as a loose upper bound that a decentralized method could ideally approach. We compare with: ① `FedAvg` (Konečný et al., 2016; McMahan et al., 2017), vanilla FL with a central server that performs orchestration and synchronous aggregation; ② Gossip Learning (`GL`) (Ormándi et al., 2013; Tang et al., 2023) with the merging mechanism of Danner et al. (2023), proposed particularly for IoT devices; ③ `SFLv1` (Thapa et al., 2022), where a central server processes the top model for clients in parallel, while a server aggregates the bottom models via FedAvg; and ④ `SFLv2` (Thapa et al., 2022), where the server processes the clients sequentially.



(a) CIFAR-100 (b) CIFAR-10 (c) TinyImageNet (d) FashionMNIST (e) MedMNIST (f) SpeechCommands

Figure 3: Shaded bands show the spread across clients for *ARC-SFL* and Danner. SFLv1, SFLv2 and FedAvg evaluate a single global model per round, so no band can be drawn for them.

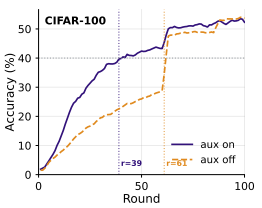


Figure 4: Auxiliary network; faster convergence while mitigating stragglers.

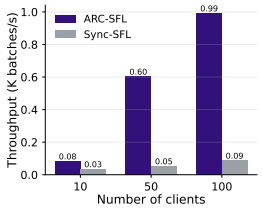


Figure 5: Throughput scaling: 11× `Sync-SFL` at 100 devices.

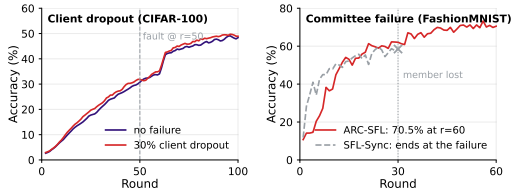


Figure 6: Left: 30% of the clients drop out at round 50 without visible effect. Right: one of three committee members fails at round 30; its clients re-attach to the survivors and training continues.

Real device Testbed and implementation. We not only implement an asynchronous simulation but also a real device testbed in `C` language, and evaluate *ARC-SFL* against the baselines on ESP32 family S3 microcontrollers with different clock speeds from 80 MHz to 240 MHz. We choose UCI-HAR and Fashion-MNIST as the representative tasks for measurement comparisons. We deploy two clusters tiny devices interconnected through a wireless router: 5 devices with one committee, and 10 devices with two committees (Fig. 2). The 5-device cluster is homogeneous (160 MHz); the 10-device cluster mixes 240, 160 and 80 MHz boards to introduce heterogeneity. We use a PC to orchestrate experiments, monitor, and collect logs over USB from each device. No learning or whatsoever happens on the PC.

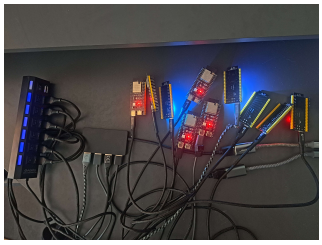


Figure 2: A partial overview of the measurement testbed.

4.1 Results

Can *ARC-SFL* achieve comparable accuracy? We have two goals: ① to clearly outperform the decentralized alternatives, and ② to close as much of the gap as possible to methods with a centralized view to their advantage. To have a fair comparison, we disable the auxiliary network and committee rotation since none of the baselines has an equivalent component. Both are evaluated separately.

As depicted in Fig. 3, *ARC-SFL* is the most consistent method across the six tasks (accuracies below are averaged over the last five rounds). It achieves the highest final accuracy on MedMNIST and TinyImageNet, and on average it remains close to the best method on each task, closer than any centralized baseline (SFLv2: 5.3%, SFLv1: 6.2%, FedAvg: 7%) and substantially better than the decentralized gossip-based `GL` (20.2%). *ARC-SFL* outperforms `GL` on all six datasets, by margins between 3% and 31%, indicating that routing updates through a committee of top models propagates information considerably more effectively than purely peer-to-peer model exchange. The centralized baselines reach higher peaks on individual tasks, but not consistently. SFLv2 leads on CIFAR-100 yet finishes last on MedMNIST, and SFLv1 and FedAvg lead on CIFAR-10 yet trail SFLv2 by more than 20% on CIFAR-100. The strongest baseline changes per task, whereas *ARC-SFL* remains comparable across all six tasks. Hence, *ARC-SFL* comes close to centralized accuracy without a central server, while clearly surpassing its decentralized counterpart.

Do auxiliary models give *ARC-SFL* an advantage? A major cost of synchronous baselines is idle time. Stragglers are one source where most clients finish their local training and then wait for the slowest devices. Split learning adds another since a client cannot update its bottom model until the gradients for its activations return. Beyond aggregating asynchronously, *ARC-SFL* fills both kinds

of idle time with local training through the auxiliary network. As shown in Fig. 4, this accelerates convergence. The auxiliary network reaches 30% accuracy in the initial 25 rounds instead of round 61, speeding up more than $2.5\times$. By continuing local training while waiting, the bottom model processes substantially more batches per unit of wall-clock time, which learns useful representations earlier. Additional results in Appendix D show the same pattern on two further settings. Our decentralized mechanism with the addition of the auxiliary network trade the cost of a small auxiliary head of at most 5-6% of the top model's parameters for a considerable reduction in the number of rounds required to converge to certain accuracy levels.

Performance on homogeneous and heterogeneous tiny real device clusters.

Figure 7 compares our measurement cluster with ten devices with our simulation. We set the committee roles to be static here, and no rotation takes place. The two curves follow each other closely over all 100 rounds. The cluster reaches 78.7% and the simulation 79.4%, with around 1% difference. Overall, on the real cluster, *ARC-SFL* shows similar performance as it does in our simulation environment.

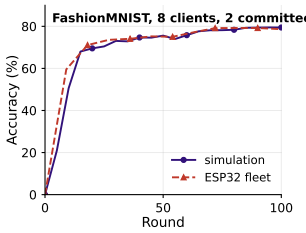


Figure 7: Measurement vs Simulation.

We also compare *ARC-SFL* with SplitFed v1 and v2 on UCI-HAR; the supporting figures with several additional metrics including communication patterns and comparison, wall-clock times, time to convergence, and events are present in Appendix in Fig. 14. In accuracy *ARC-SFL* starts slow and ends on par with its centralized counterparts SplitFed v1 (73.6% vs. 71.1% on 5 boards, 72.8% vs. 71.6% on 10 boards). Similarly to the simulation results, SplitFed v2 expectedly achieves slightly higher accuracy in both clusters.On the other hand, *ARC-SFL* significantly reduces the communication overhead by up to 34%. The wall-clock time is where *ARC-SFL* shows its full dominance. On the ten-device heterogeneous cluster, our method completes 100 rounds within 38 min, while the baselines require $\approx 3.5\times$ more time (128 to 132 min). On the real device clusters, *ARC-SFL* matches the parallel SplitFed baseline in accuracy with a third less communication and clearly faster runs.

How does it scale? We compare *ARC-SFL* against synchronous centralized aggregation (`Sync-SFL`) while varying the number of participants from 10 to 50 and eventually to 100 devices. Here, the committee scales with the deployment and is set to 10% of the clients; one, five, and 10 members, respectively. We measure system throughput, defined as the number of processed training batches per unit of time (bps) until all clients have finished. As depicted in Fig. 5, the centralized baseline scales poorly, going from 10 to 100 clients raises its throughput only from 35 to 87 bps *ARC-SFL* distributes the server-side workload across committee members and uses idle client time for auxiliary training, reaching nearly 1000 bps at 100 clients which is an $11\times$ improvement over `Sync-SFL` and a $12\times$ increase over its own throughput at 10 clients. The advantage does not stem from the auxiliary batches alone. Counting only split learning batches, which both systems perform, *ARC-SFL* still processes 249 bps against 87 for `Sync-SFL` at 100 clients, a $2.9\times$ improvement. At 10 clients the two are on par (32 vs. 35 bps): with a single committee member *ARC-SFL* degenerates to one server, so there is no parallelism to gain over the baseline. The benefit appears as soon as the committee grows, reaching $2.7\times$ at 50 clients with five members. All throughput numbers are measured on CIFAR-100 with ResNet-18.

How robust is *ARC-SFL* to device failures? When deploying on tiny devices in real-world settings, failures of both clients and committee members are inevitable. We therefore study two scenarios on CIFAR-100 with ResNet-18. ① *Client dropout.* At round 50, 30% of the clients permanently become unavailable (Fig. 6). The loss has no visible effect on *ARC-SFL*'s convergence: the run with the failure tracks its no-failure reference throughout and ends at 48.9% against 48.4%, a difference well inside the run-to-run noise. *ARC-SFL* does not wait for all clients before closing a round, so losing a third of them removes work but does not stall the remaining ones. Their contribution has moreover already propagated through gossip before the failure. ② *Committee member failure.* At round 30, one of three committee members is terminated while the other two keep running. The clients attached to the failed member detect the loss and re-attach to a surviving committee member. Client participation drops from 10 to 7 for two rounds and is back to 10 at round 33; training then continues to the end of the run without any lasting effect on accuracy. `Sync-SFL` cannot absorb the same failure: its single server is also its only server, so the run stops at round 30 and never resumes. The two scenarios probe different parts of the design. Client dropout tests whether the round can

close without every participant, which asynchrony provides. Committee failure tests whether the top-model role is replaceable, which the committee provides. Therefore, with more than one member, the failure of any single one is a temporary loss of capacity rather than the end of training.

How does *ARC-SFL* improve communication efficiency? We compare five strategies that differ in when a client serves its model: *Loss*, when its local loss improves; *Drift*, when a significant representation drift occurs; *Loss+Drift*, when both hold, as in *ARC-SFL*; *Random* to test whether communication is triggered meaningfully roughly or randomly three out four times; and *Always*, the default. Table 2 shows that *Loss+Drift* reduces the sent gossip volume the most in both settings, by 55% on MobileNetV2/CIFAR-10 and 56% on ResNet-18/TinyImageNet. On CIFAR-10 this comes at no cost as *Loss+Drift* reaches 60.7% against 59.2% for *Always*, the highest accuracy of all five strategies. While this may initially seem counter-intuitive, always gossiping also propagates noisy or stale updates that interfere with convergence, and the trigger filters them out. For the larger ResNet-18 the same aggressive filtering costs 2.1 points (27.5% vs. 29.6%), whereas *Drift* alone matches *Always* while still sending 26% less. Larger models therefore call for a more conservative trigger. *Random* is never competitive at a comparable volume, confirming that the savings come from deciding *when* to communicate rather than from communicating less often as such.

Table 2: Sent gossip volume, its reduction relative to *Always*, and final accuracy per communication trigger.

	CIFAR-10			TinyImageNet		
Trigger	Vol.	Δ	Acc.	Vol.	Δ	Acc.
Always	2076	–	59.2	10243	–	29.6
Random	1510	−27%	56.2	7555	−26%	28.7
Loss	1263	−39%	60.5	6043	−41%	27.8
Drift	1678	−19%	57.6	7628	−26%	**29.6**
Loss+Drift	**933**	−55%	**60.7**	**4498**	−56%	27.5

What does each component contribute? Since *ARC-SFL* is a hierarchical framework, we isolate its components on CIFAR-10: *SFL Sync* is the standard synchronized approach with a centralized server, *Sync Committee* remains synchronized but employs multiple committee members, and *ARC-SFL* without rotation adds the auxiliary network and staleness-aware updates. Figure 8 shows that these components buy speed in two distinct senses. In throughput, both *ARC-SFL* variants process three to four times as many batches per client and second as *SFL Sync* (18.5 and 20.9 against 5.7). In convergence, *ARC-SFL* without rotation reaches 45% accuracy at round 40, where both baselines need round 70, and 55% at round 70, where they need round 90 or later; at the midpoint of training it leads the better baseline by 11.6 points (46.9% against 35.3% at round 50). Its accuracy at round 90 is also the highest of all variants, 59.5% against 55.0% and 55.4% for the baselines. Rotation costs some of that lead while raising throughput further, since clients keep training their auxiliary networks while devices swap roles. Its purpose, however, is not speed but participation: rotating the committee role lets the data of every device shape the top model, instead of concentrating that influence in a fixed subset of the clients. Two further ablations, on the committee size and on the two staleness discounts, are reported in Appendix D.

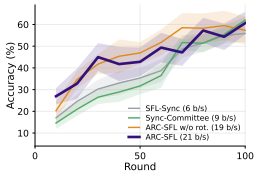


Figure 8: Component ablation on CIFAR-10. Accuracy over rounds; the legend gives each variant's throughput in batches per second and client.

5 Conclusion

We present *ARC-SFL*, a decentralized and asynchronous split federated learning with auxiliary networks for constrained IoT devices. *ARC-SFL* forms a committee from the same device pool and rotates its members over time. We show that our method significantly improves communication efficiency and scalability, while being robust to severe failure scenarios. Although fully decentralized, *ARC-SFL* outperforms gossip learning on every benchmark and performs on par with centralized FL and SFL, showing that decentralization does not always come at the expense of accuracy. We view our *ARC-SFL* and our implementation for simulation and real deployment as not only a theoretical but an important practical step towards achieving pervasive intelligence involving tiny devices. At the same time, we foresee a long way to go as one can build on top of *ARC-SFL* and further tackle adaptive communication strategies, committee selection mechanisms, and robustness to malicious participants as direct future works.

Reproducibility Statement

We specify all algorithms in Section 3 and Appendix B, give the full convergence analysis in Appendix C, and describe model splits, hyperparameters, and the simulation environment in Appendix A. We plan to make our code and artifacts available and provide step-by-step guidelines for reproducing the results and facilitating building on top of *ARC-SFL* upon acceptance.

AI Statement

We used generative AI tools to polish the writing, assist with literature retrieval and discovery, support code implementation through pair programming, and assist in developing proofs of mathematical claims. We did not use generative AI to generate synthetic datasets. The authors manually reviewed and verified the AI-assisted work, including the code, mathematical proofs, and references. We take responsibility for the final content of this paper and its accompanying artifacts, including all text, claims, and results produced with AI assistance.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. FedRolex: Model-Heterogeneous Federated Learning with Rolling Sub-Model Extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Mohamad Arafeh, Mohamad Wazzeh, Hani Sami, Hakima Ould-Slimane, Chamseddine Talhi, Azzam Mourad, and Hadi Otrok. Efficient Privacy-Preserving ML for IoT: Cluster-Based Split Federated Learning Scheme for Non-IID Data. *Journal of Network and Computer Applications*, 236:104105, 2025.

Navidreza Asadi and Maziar Goudarzi. Variant Parallelism: Lightweight Deep Convolutional Models for Distributed Inference on IoT Devices. *IEEE Internet of Things Journal*, 11(1):345–352, 2024.

Navidreza Asadi, Halil İbrahim Bengu, Lars Wulfert, Hendrik Wöhrle, and Wolfgang Kellerer. Gist – Optimizing Segmentation for Decentralized Federated Learning on Tiny Devices. In *Proceedings of the Workshop on Federated Learning and Edge AI for Privacy and Mobility (FLEdge-AI)*, pp. 77–84, 2025.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic Gradient Push for Distributed Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 344–353, 2019.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to Backdoor Federated Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2938–2948, 2020.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy Layerwise Learning Can Scale to ImageNet. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 583–593, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled Greedy Learning of CNNs. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 736–745, 2020.

Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*, 2020.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1175–1191, 2017.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 1, 2019.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized Gossip Algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. TiFL: A Tier-Based Federated Learning System. In *Proceedings of the International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, pp. 125–136, 2020.

Zheng Chai, Yujing Chen, Ali Anwar, Liang Zhao, Yue Cheng, and Huzefa Rangwala. FedAT: A High-Performance and Communication-Efficient Federated Learning System with Asynchronous Tiers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–16, 2021.

Chunjiang Che, Xiaoli Li, Chuan Chen, Xiaoyu He, and Zibin Zheng. A Decentralized Federated Learning Framework via Committee Mechanism with Convergence Guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4783–4800, 2022.

Tianyi Chen, Georgios Giannakis, Tao Sun, and Wotao Yin. LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Zhipeng Cheng, Xiaoyu Xia, Minghui Liwang, Xuwei Fan, Yanglong Sun, Xianbin Wang, and Lianfen Huang. CHEESE: Distributed Clustering-Based Hybrid Federated Split Learning over Edge Networks. *IEEE Transactions on Parallel and Distributed Systems*, 34(12):3174–3191, 2023.

Gábor Danner, István Hegedűs, and Márk Jelasity. Improving Gossip Learning via Limited Model Merging. In *Advances in Computational Collective Intelligence (ICCCI)*, Communications in Computer and Information Science, pp. 351–363. Springer, 2023.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *International Conference on Learning Representations (ICLR)*, 2021.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union, L 119, 2016.

Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. A General Theory for Federated Optimization with Asynchronous and Heterogeneous Clients Updates. *Journal of Machine Learning Research*, 24(110):1–43, 2023.

Yansong Gao, Minki Kim, Chandra Thapa, Alsharif Abuadbba, Zhi Zhang, Seyit Camtepe, Hyoungshick Kim, and Surya Nepal. Evaluation and Optimization of Distributed Machine Learning Techniques for Internet of Things. *IEEE Transactions on Computers*, 71(10):2538–2552, 2022.

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 (NeurIPS)*, volume 33, 2020.

Jemin George and Prudhvi Gurram. Distributed Stochastic Gradient Descent with Event-Triggered Communication. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7169–7178, 2020.

Nil Llisterri Giménez, Joan Miquel Solé, and Felix Freitag. Embedded Federated Learning over a LoRa Mesh Network. *Pervasive and Mobile Computing*, 93:101819, 2023.

Alexander Gräfe, Fabian Mager, Marco Zimmerling, and Sebastian Trimpe. RockNet: Distributed Learning on Ultra-Low-Power Devices. *ACM Transactions on Cyber-Physical Systems*, 10(1): 1–28, 2026.

Hongyu Guo, Navidreza Asadi, Giovanni Bartolomeo, Patrick Laufer, Jörg Ott, and Wolfgang Kellerer. Comparative Performance and Cost Analysis of Computer Vision in Edge-Cloud Continuum. In *Proceedings of the ACM CoNEXT Workshop on Edge-Cloud Collaboration for AI*, pp. 30–37, 2025.

Otkrist Gupta and Ramesh Raskar. Distributed Learning of Deep Neural Network over Multiple Agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Dong-Jun Han, Hasnain Irshad Bhatti, Jungmoon Lee, and Jaekyun Moon. Accelerating Federated Learning with Split Learning on Locally Generated Losses. In *ICML Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2021.

Dong-Jun Han, Do-Yeon Kim, Minseok Choi, Christopher G. Brinton, and Jaekyun Moon. SplitGP: Achieving Both Generalization and Personalization in Federated Learning. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–10, 2023.

Pengchao Han, Chao Huang, Geng Tian, Ming Tang, and Xin Liu. Convergence Analysis of Split Federated Learning on Heterogeneous Data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 14068–14080, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model Inversion Attacks Against Collaborative Inference. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 148–162, 2019.

István Hegedűs, Gábor Danner, and Márk Jelasity. Decentralized Learning Works: An Empirical Comparison of Gossip Learning and Federated Learning. *Journal of Parallel and Distributed Computing*, 148:109–124, 2021.

Samuel Horváth, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. FjORD: Fair and Accurate Federated Learning under Heterogeneous Targets with Ordered Dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 629–647, 2017.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The Non-IID Data Quagmire of Decentralized Machine Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4387–4398, 2020.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *arXiv preprint arXiv:1909.06335*, 2019.

Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pp. 1423–1431, 2019.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*, 2018.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Zhouyuan Huo, Bin Gu, Qian Yang, and Heng Huang. Decoupled Parallel Backpropagation with Convergence Guarantee. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2098–2106, 2018.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces Using Synthetic Gradients. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1627–1635, 2017.

Tian Jin and Seokin Hong. Split-CNN: Splitting Window-Based Operations in Convolutional Neural Networks for Memory System Optimization. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 835–847, 2019.

Peter Kairouz, H. Brendan McMahan, et al. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 615–629, 2017.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143, 2020.

Hossein Katebi, Navidreza Asadi, and Maziar Goudarzi. FullPack: Full Vector Utilization for Sub-Byte Quantized Matrix-Vector Multiplication on General Purpose CPUs. *IEEE Computer Architecture Letters*, 23(2):142–145, 2024.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3301–3310, 2019.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-Based Computation of Aggregate Information. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 482–491, 2003.

Janis Keuper and Franz-Josef Pfreundt. Asynchronous Parallel Stochastic Gradient Descent: A Numeric Core for Scalable Distributed Machine Learning Algorithms. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments (MLHPC)*, pp. 1–11, 2015.

Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3478–3487, 2019.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A Unified Theory of Decentralized SGD with Changing Topology and Local Updates. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5381–5393, 2020.

Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Sharper Convergence Guarantees for Asynchronous SGD for Distributed and Federated Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated Optimization: Distributed Optimization Beyond the Datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

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

Lingjing Kong, Tao Lin, Anastasia Koloskova, Martin Jaggi, and Sebastian Stich. Consensus Control for Decentralized Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5686–5696, 2021.

Kavya Kopparapu, Eric Lin, John G. Breslin, and Bharath Sudharsan. TinyFedTL: Federated Transfer Learning on Ubiquitous Tiny IoT Devices. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 79–81, 2022.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient Federated Learning via Guided Participant Selection. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 19–35, 2021.

Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-Peer Federated Learning on Graphs. *arXiv preprint arXiv:1901.11173*, 2019.

Stefanos Laskaridis, Stylianos I. Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D. Lane. SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 1–15, 2020.

Ya Le and Xuan Yang. Tiny ImageNet Visual Recognition Challenge. Technical report, Stanford University, CS231N Course Report, 2015.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified Robustness to Adversarial Examples with Differential Privacy. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 656–672, 2019.

En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Transactions on Wireless Communications*, 19(1): 447–457, 2020a.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label Leakage and Protection in Two-Party Split Learning. In *International Conference on Learning Representations (ICLR)*, 2022.

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

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 2, 2020c.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-IID Data. In *International Conference on Learning Representations (ICLR)*, 2020d.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous Decentralized Parallel Stochastic Gradient Descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3043–3052, 2018.

Yunming Liao, Yang Xu, Hongli Xu, Zhiwei Yao, Liusheng Huang, and Chunming Qiao. ParallelSFL: A Novel Split Federated Learning Framework Tackling Heterogeneity Issues. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 845–860, 2024.

Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. MCUNet: Tiny Deep Learning on IoT Devices. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020a.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-Device Training Under 256KB Memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 22941–22954, 2022.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble Distillation for Robust Model Fusion in Federated Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020b.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations (ICLR)*, 2018.

Zheng Lin, Guangyu Zhu, Yiqin Deng, Xianhao Chen, Yue Gao, Kaibin Huang, and Yuguang Fang. Efficient Parallel Split Learning over Resource-Constrained Wireless Edge Networks. *IEEE Transactions on Mobile Computing*, 23(10):9224–9239, 2024.

Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, pp. 464–483, 2020c.

Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an End to End-to-End: Gradient-Isolated Learning of Representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, 2017.

Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake Woodworth. Asynchronous SGD Beats Minibatch SGD Under Arbitrary Delays. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pp. 1–15, 2019.

Angelia Nedić and Asuman Ozdaglar. Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

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 International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pp. 3581–3607, 2022.

Arild Nøkland and Lars Hiller Eidnes. Training Neural Networks with Local Error Signals. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4839–4850, 2019.

Seungeun Oh, Jihong Park, Praneeth Vepakomma, Sihun Baek, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. LocFedMix-SL: Localize, Federate, and Mix for Improved Scalability, Convergence, and Latency in Split Learning. In *Proceedings of the ACM Web Conference (WWW)*, pp. 3347–3357, 2022.

Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip Learning with Linear Models on Fully Distributed Data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the Tiger: Inference Attacks on Split Learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 2113–2129, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Benjamin Recht, Christopher Ré, Stephen Wright, and Feng Niu. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011.

Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive Federated Optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Max Ryabinin, Eduard Gorbunov, Vsevolod Plokhotnyuk, and Gennady Pekhimenko. Moshpit SGD: Communication-Efficient Decentralized Training on Heterogeneous Unreliable Devices. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. SWARM Parallelism: Training Large Models Can Be Surprisingly Communication-Efficient. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29416–29440, 2023.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Jinglong Shen, Nan Cheng, Xiucheng Wang, Feng Lyu, Wenchao Xu, Zhi Liu, Khalid Aldubaikhy, and Xuemin Shen. RingSFL: An Adaptive Split Federated Learning Towards Taming Client Heterogeneity. *IEEE Transactions on Mobile Computing*, 23(5):5462–5478, 2024.

Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi. SPARQ-SGD: Event-Triggered and Compressed Communication in Decentralized Optimization. *IEEE Transactions on Automatic Control*, 68(2):721–736, 2023.

Amirreza Sokhankhosh, Khalid Hassan, and Sara Rouhani. Enhancing split learning with sharded and blockchain-enabled splitfed approaches, 2025. URL https://arxiv.org/abs/2509.25555.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Tao Sun, Dongsheng Li, and Bao Wang. Decentralized Federated Averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2023.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, 2019.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized Training over Decentralized Data. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4848–4856, 2018.

Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. GossipFL: A Decentralized Federated Learning Framework with Sparsified and Adaptive Communication. *IEEE Transactions on Parallel and Distributed Systems*, 34(3):909–922, 2023.

Bernardo Camajori Tedeschini, Mattia Brambilla, and Monica Nicoli. Split Consensus Federated Learning: An Approach for Distributed Training and Inference. *IEEE Access*, 12:119535–119549, 2024.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Distributed Deep Neural Networks over the Cloud, the Edge and End Devices. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 328–339, 2017.

Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. SplitFed: When Federated Learning Meets Split Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8485–8493, 2022.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split Learning for Health: Distributed Deep Learning without Sharing Raw Patient Data. *arXiv preprint arXiv:1812.00564*, 2018.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Luping Wang, Wei Wang, and Bo Li. CMFL: Mitigating Communication Overhead for Federated Learning. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 954–964, 2019a.

Qiyuan Wang, Qianqian Yang, Shibo He, Zhiguo Shi, and Jiming Chen. AsyncFedED: Asynchronous Federated Learning with Euclidean Distance Based Adaptive Weight Aggregation. *arXiv preprint arXiv:2205.13797*, 2022.

Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019b.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting Locally Supervised Learning: An Alternative to End-to-End Training. In *International Conference on Learning Representations (ICLR)*, 2021.

Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep Learning via Semi-Supervised Embedding. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1168–1175, 2008.

Wen Wu, Mushu Li, Kaige Qu, Conghao Zhou, Xuemin Shen, Weihua Zhuang, Xu Li, and Weisen Shi. Split Learning over Wireless Networks: Parallel Design and Resource Management. *IEEE Journal on Selected Areas in Communications*, 41(4):1051–1066, 2023.

Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. SAFA: A Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2021.

Lars Wulfert, Navidreza Asadi, Wen-Yu Chung, Christian Wiede, and Anton Grabmaier. Adaptive Decentralized Federated Gossip Learning for Resource-Constrained IoT Devices. In *Proceedings of the International Workshop on Distributed Machine Learning (DistributedML)*, pp. 27–33, 2023a.

Lars Wulfert, Christian Wiede, and Anton Grabmaier. TinyFL: On-Device Training, Communication and Aggregation on a Microcontroller for Federated Learning. In *Proceedings of the IEEE Interregional NEWCAS Conference (NEWCAS)*, pp. 1–5, 2023b.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous Federated Optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2 – A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification. *Scientific Data*, 10(1):41, 2023.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5650–5659, 2018.

Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical GAN-Based Synthetic IP Header Trace Generation Using NetShare. In *Proceedings of the ACM SIGCOMM Conference*, pp. 458–472, 2022.

Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Ré, and Ce Zhang. Decentralized Training of Foundation Models in Heterogeneous Environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Shahryar Zehtabi, Seyyedali Hosseinalipour, and Christopher G. Brinton. Decentralized Event-Triggered Federated Learning with Heterogeneous Communication Thresholds. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 4680–4687, 2022.

Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-Aware Async-SGD for Distributed Deep Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2350–2356, 2016.

Zhuoran Zhao, Kamyar Mirzazad Barijough, and Andreas Gerstlauer. DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2348–2359, 2018.

Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous Stochastic Gradient Descent with Delay Compensation. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 4120–4129, 2017.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

A Implementation Details

All simulated experiments run on a single multi-GPU machine. We use PyTorch (Paszke et al., 2019) for model training and Flower (Beutel et al., 2020) for federated life-cycle orchestration; every client and every committee member runs as a separate process. For the inter-device communication layer, we implement a custom gRPC-based protocol with two distinct data flows: (1) a client-committee link for the exchange of smashed activations and corresponding gradients, and (2) a decentralized peer-to-peer (P2P) network for gossip-based model exchange. Each committee member runs its own Flower server over its share of the clients, and Flower enforces a round barrier within each committee: a round ends only once all of its clients have returned, with fast clients training their auxiliary head while they wait. Committees are not synchronized with each other; they advance at their own pace and exchange top models without waiting, so the staleness weighting acts on gossip across committees. On hardware there is no such barrier since after its split-learning batches and a fixed number of auxiliary steps, each client advances to the next round on its own, and the committee accepts activations from any round.

Tables 3 and 4 list the settings of the accuracy comparison in Fig. 3. The optimizer settings are identical for every method and dataset, so that differences in performance mainly reflect the learning method rather than optimizer choices. Non-IID label skew is induced with a Dirichlet concentration of 0.5. The steps in the accuracy curves at round 60 stem from the learning rate schedule.

Table 3: Training hyperparameters of Fig. 3, shared by every method and dataset.

Hyperparameter	**Value**
Batch size	32
Learning rate (bottom / top)	0.01 / 0.01
Optimizer	SGD (momentum 0.9, weight decay 5×10^{-4})
LR schedule	MultiStep at rounds 60/85/95, $\gamma = 0.1$
Dirichlet concentration	0.5
Number of clients	50
Communication rounds	100
Evaluation interval	every 5 rounds
Seeds	1

Hyperparameters. Beyond the usual optimizer settings, *ARC-SFL* exposes three groups of hyperparameters, one per mechanism: the committee (M, E_{rot}, E_{sync}, ρ), staleness handling (α, β), and gossip (K, δ, $N_{\max}$). We keep all of them fixed across datasets and models; Table 5 groups them by the mechanism they control and the rationale behind. We fix all of them across tasks and tune none per dataset; the sensitivity studies in Section 4 and Appendix D shows how each behave.

While clients handle bottom-model aggregation as described in Eq. (2), committee members address staleness caused by delayed smashed activations. Since clients send activations at different speeds, gradients computed from older activations can destabilize training. To mitigate this, the committee damps its local top-model update with the same discount family, using the exponent α:

$$\theta^m_{\text{top}} \leftarrow \theta^m_{\text{top}} - \eta_{\text{top}}\, \phi_\alpha(\tau_i)\, \nabla_{\theta^m_{\text{top}}} \mathcal{L}_{\text{global}}, \qquad \tau_i = \max(0, R - r_i), \tag{5}$$

where the committee round R is the latest local round reported by any of the member's clients, so that τ_i measures how far client i lags behind its fastest peers. In contrast to the client side ($\beta = 2$),

Table 4: Models and method-specific settings of Fig. 3. Gradient clipping is applied in *ARC-SFL* only; the baselines run without it.

Model per dataset	
CIFAR-100, TinyImageNet	ResNet-18
CIFAR-10, Speech Commands	MobileNetV2
MedMNIST	EfficientNet-B0
FashionMNIST	4-layer CNN
ARC-SFL	
Committee size	3
Gossip fanout K	16
Auxiliary network / rotation	disabled
Gossip trigger	every round
Gossip staleness exponent β	2.0
Gradient damping exponent α	0.5
Round-close threshold	0.7 of a committee's clients
Gradient clipping (max. norm)	1.0
Baselines	
SFLv1	–
SFLv2	client shuffling
FedAvg	–
`GL`	gossip fanout $K = 16$, momentum reset on exchange

we set $\alpha = 1/2$ and therefore penalize stale gradients more conservatively. We use this milder penalty because slightly stale gradients still preserve their descent direction (Keuper & Pfreundt, 2015; Zheng et al., 2017), whereas stale model parameters have a stronger effect during direct weight averaging. The committee returns the activation gradient $G_{\text{top}}^{(r_i)}$ itself to the client unscaled, so that the bottom model receives the full split learning signal.

Table 5: *ARC-SFL*-specific hyperparameters, grouped by mechanism, with the values used in all experiments. Exceptions: the committee-size sweep varies M; the scalability study uses $M = 0.1 * N$; the robustness studies use $M = 3$ and $\rho = 0.4$ to recover quickly from failures, with the auxiliary network enabled and, for client dropout, rotation enabled; the committee-failure experiment additionally runs on FashionMNIST with $N = 10$ for 60 rounds; the gossip hyperparameters δ and $N_{\max}$ apply only to the event-trigger study (§ 4.1), where the loss check uses three-round averages after a warm-up of six rounds. Furthermore, The event-trigger study (§ 4.1) uses a committee of four.

Mechanism	**Symbol**	**Role**	**Value**
Committee	M	committee size	3 (6%)
	E_{rot}	committee rounds between role rotations	20
	E_{sync}	committee rounds between consensus steps	1
	ρ	fraction of finished clients closing a round	0.7
Staleness	α	exponent of ϕ_α for stale gradients	$1/2$
	β	exponent of ϕ_β for stale gossiped models	2
Gossip	K	peers contacted per gossip phase	16
	$N_{\max}$	silent rounds before sharing is forced	10

Rationale. *Committee.* The committee size trades throughput for accuracy (Figure 11): each member serves N/M clients, so a larger M shortens queues but lets clients on different members drift apart. We use $M = 3$ and find 5–10% of the clients to be a good operating range; consensus among the M replicas is cheap and runs every round. *Staleness.* Both discounts come from the family $\phi_\gamma(\tau) = (1 + \tau)^{-\gamma}$, with $\gamma = 0$ recovering staleness-agnostic training. We penalize stale models more strongly than stale gradients ($\beta > \alpha$): a slightly stale gradient still points in a descent direction (Keuper & Pfreundt, 2015; Zheng et al., 2017), whereas averaging in a stale model pulls the parameters back directly. The sweep with real stragglers in Figure 12 bears this out: the model discount is what matters, as $\beta = 0$ collapses to 13% while $\beta \geq 1$ stays close to the no-straggler range.

The gradient discount is secondary but not irrelevant: with the weaker model discount $\beta = 1$, $\alpha = 1$ lifts accuracy back into the no-straggler range where the other values fall short, which indicates that damping stale gradients helps when stale models are discounted less; with $\beta = 2$ the choice of α no longer matters.

Model details and partitioning. We partition each model into a client-side bottom model, a committee-side top model, and a local auxiliary head for asynchronous training (Table 6). We select the split points relatively deep in the network, so that approximately 15–25% of the total parameters reside on the client. This balances the hardware constraints of tiny IoT devices with the need for high-level feature abstraction for better privacy preservation. Furthermore, deeper partitioning helps the auxiliary networks receive sufficiently rich representations to generate useful local updates during idle time. We deliberately keep the auxiliary head lightweight, at about 5% of the top model's parameters or less, to ensure minimal computational overhead for the constrained IoT devices.

Table 6: Parameter distribution and split-point configuration. Client % denotes the share of the whole model on the client, and Aux/Top % the size of the auxiliary network relative to the top model.

Base Architecture	Split Point Layer	Bottom (P)	Top (P)	Aux (P)	Client %	Aux/Top %
FashionCNN	Post-Conv2 Pool	13.2K	66.9K	2.1K	16.4%	3.1%
MobileNetV2	14th Block	542K	1.69M	70K	24.2%	4.1%
EfficientNet-B0	6th Block	851K	3.16M	161K	21.2%	5.1%
ResNet-18	Post-Layer 3	2.77M	8.44M	235K	24.7%	2.8%

B Event-Triggered Gossip Decision

Algorithm 3 summarizes the event-trigger logic referenced in Algorithm 1. It first increments the silent counter and checks whether the client has withheld its model for too many consecutive rounds. In that case, sharing is forced to guarantee liveness. Next, it compares the mean loss of the last three rounds with that of the three rounds before and returns false if the loss has increased, since such an update is unlikely to benefit peers; during the first six rounds, this check is skipped. Otherwise, it computes the mean smashed activation for the current round and its cosine distance to the reference activation. The drift threshold is not a fixed constant but the q-th percentile of the client's own drift history $\mathcal{D}_i$, so that the criterion adapts to the scale of each client's representation changes and suppresses only the rounds in which the representation barely moves. The reference activation is updated whenever the drift exceeds this threshold. When the algorithm returns false, the client does not serve its model to requesting peers, while it still requests and aggregates the models of others. The silent counter is reset whenever the client shares its model.

C Convergence Analysis

This appendix provides a conditional stationarity analysis for *ARC-SFL*. The result is not an unconditional convergence guarantee for arbitrary asynchronous executions. Instead, it characterizes the averaged optimization dynamics under explicit assumptions on stale gradients, local auxiliary updates, replica disagreement, and the average drift introduced by non-mean-preserving coordination operations.

C.1 Target Objective and Distributed Replicas

The optimization target is the shared SFL objective from Eq. (1),

$$F(\theta_{\text{bot}}, \theta_{\text{top}}) = \sum_{i=1}^{N} p_i \mathbb{E}_{(x,y)\sim\mathcal{D}_i} \left[\ell \left(f_{\theta_{\text{top}}} \left(f_{\theta_{\text{bot}}}(x) \right), y \right) \right], \qquad p_i = \frac{n_i}{\sum_{j=1}^{N} n_j}. \tag{6}$$

The variables θ_{bot} and θ_{top} denote shared target parameters. The distributed execution maintains local replicas θ^i_{bot} at clients and θ^m_{top} at committee members. The auxiliary heads θ^i_{aux} are local optimization aids and are not parameters of the target objective in Eq. (6).

Algorithm 3 *ARC-SFL* Event-Based Trigger Decision (the caller resets $z_i \leftarrow 0$ whenever the client shares)

Require: round losses $\{\mathcal{L}_i^{(t)}\}_{t \le r_i}$, current activations $h_i^{(r_i)}$, reference activation $\hat{h}_i$, drift history $\mathcal{D}_i$, silent counter z_i, drift percentile q, maximum silent rounds $N_{\max}$

$z_i \leftarrow z_i + 1$
if $z_i > N_{\max}$ **then** ▷ 1. Liveness guarantee
  return `True`
end if
if $r_i \ge 6$ **and** $\mathrm{mean}(\mathcal{L}_i^{(r_i-2:r_i)}) > \mathrm{mean}(\mathcal{L}_i^{(r_i-5:r_i-3)})$ **then** ▷ 2. Loss check
  return `False`
end if
$\bar{h}_i^{(r_i)} \leftarrow \frac{1}{|h_i^{(r_i)}|} \sum_{a \in h_i^{(r_i)}} a$ ▷ 3. Drift check
$d_i^{(r_i)} \leftarrow 1 - \cos\left(\bar{h}_i^{(r_i)}, \hat{h}_i\right)$
$\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{d_i^{(r_i)}\}$
$\delta_i \leftarrow \mathrm{Quantile}_q(\mathcal{D}_i)$
if $d_i^{(r_i)} \ge \delta_i$ **then**
  $\hat{h}_i \leftarrow \bar{h}_i^{(r_i)}$
  return `True`
end if
return `False`

The analysis abstracts away optimizer momentum and studies the base asynchronous stochastic-gradient dynamics. Consequently, the theorem below should be interpreted as a characterization of the underlying optimization mechanism rather than as a convergence theorem for every optimizer detail used in the experiments.

C.2 Averaged Iterates and Replica Disagreement

Let $\mathcal{F}_t$ denote the history of the asynchronous execution up to the analysis event t. The event index is a logical analysis index and need not coincide with any client's local round r_i or the committee round R.

For the bottom-model replicas, define the sample-weighted average

$$\bar{\theta}_{\mathrm{bot},t} = \sum_{i=1}^{N} p_i \theta_{\mathrm{bot},t}^{i}. \tag{7}$$

For the currently active committee $\mathcal{C}(t)$, define the arithmetic average of its top-model replicas

$$\bar{\theta}_{\mathrm{top},t} = \frac{1}{|\mathcal{C}(t)|} \sum_{m \in \mathcal{C}(t)} \theta_{\mathrm{top},t}^{m}. \tag{8}$$

We collect both blocks in

$$\bar{w}_t = (\bar{\theta}_{\mathrm{bot},t}, \bar{\theta}_{\mathrm{top},t}). \tag{9}$$

The corresponding replica disagreements are

$$D_t^{\mathrm{bot}} = \sum_{i=1}^{N} p_i \|\theta_{\mathrm{bot},t}^{i} - \bar{\theta}_{\mathrm{bot},t}\|^2, \tag{10}$$

$$D_t^{\mathrm{top}} = \frac{1}{|\mathcal{C}(t)|} \sum_{m \in \mathcal{C}(t)} \|\theta_{\mathrm{top},t}^{m} - \bar{\theta}_{\mathrm{top},t}\|^2, \tag{11}$$

$$D_t = D_t^{\mathrm{bot}} + D_t^{\mathrm{top}}. \tag{12}$$

No exact mean-preservation property is assumed for receiver-side gossip or for the participation-weighted committee synchronization. Their effect on the averaged iterate is modeled explicitly below.

C.3 Effective Averaged Update

We decompose one event-indexed change of the averaged state as

$$\bar{w}_{t+1} = \bar{w}_t - \eta\big(\widetilde{g}_t + a_t\big) + \delta_t^{\mathrm{coord}}. \tag{13}$$

Here, $\widetilde{g}_t$ denotes the effective stochastic update associated with the main SFL objective, including the effect of delayed smashed activations and the committee-side staleness damping. The vector a_t denotes the averaged bottom-model update induced by local auxiliary training, expressed in units of the base learning rate η. Finally, $\delta_t^{\mathrm{coord}}$ denotes the displacement of the averaged state that is caused by coordination operations rather than by a gradient step. This term includes, in particular, receiver-side bottom-model gossip and any change of the top-model average caused by participation-weighted committee consensus or role rotation.

Equation (13) is a decomposition of the average state change; the substantive assumptions are the bounds imposed on its three non-ideal components below. In particular, we do not assume that receiver-side gossip preserves the sample-weighted average.

C.4 Assumptions

Assumption C.1 (Smoothness and lower boundedness)**.** The target objective F is lower bounded by $F^\star > -\infty$ and has an L-Lipschitz-continuous gradient:

$$\|\nabla F(u) - \nabla F(v)\| \le L\|u - v\| \quad \text{for all } u, v.$$

Assumption C.2 (Conditional variance of the main stochastic update)**.** Let

$$g_t := \mathbb{E}[\widetilde{g}_t \mid \mathcal{F}_t], \qquad \xi_t := \widetilde{g}_t - g_t.$$

Then

$$\mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0, \qquad \mathbb{E}[\|\xi_t\|^2 \mid \mathcal{F}_t] \le \sigma^2.$$

Assumption C.3 (Effective stale-gradient and replica bias)**.** Split-gradient delays are bounded by a finite deployment-dependent constant. Moreover, the conditional bias of the effective main-objective update satisfies

$$\|g_t - \nabla F(\bar{w}_t)\|^2 \le C_D D_t + \varepsilon_\tau, \tag{14}$$

for some $C_D \ge 0$ and $\varepsilon_\tau \ge 0$. The term $C_D D_t$ captures evaluation of gradients at non-consensus local replicas, while ε_τ captures residual error due to stale smashed activations and the asymmetric treatment of staleness in the top- and bottom-model updates.

Assumption C.4 (Bounded auxiliary perturbation)**.** The average update induced by local auxiliary training satisfies

$$\mathbb{E}[\|a_t\|^2 \mid \mathcal{F}_t] \le \varepsilon_{\mathrm{aux}} \tag{15}$$

for a finite $\varepsilon_{\mathrm{aux}} \ge 0$. This assumption deliberately does not identify auxiliary training with the gradient of a fixed composite objective: the number of reactive auxiliary steps can vary across clients and over time.

Assumption C.5 (Controlled coordination-induced average drift)**.** The average displacement caused by non-gradient coordination operations obeys

$$\mathbb{E}\big[\|\delta_t^{\mathrm{coord}}\|^2 \mid \mathcal{F}_t\big] \le \eta^2 \left(C_{\mathrm{coord}} D_t + \varepsilon_{\mathrm{coord}}\right) \tag{16}$$

for constants $C_{\mathrm{coord}} \ge 0$ and $\varepsilon_{\mathrm{coord}} \ge 0$. This is an effective mixing condition. It is not implied solely by the event trigger or by row-normalization of receiver-side gossip weights. Rather, it states explicitly the additional property required for a stationarity bound on the averaged iterate when the implemented coordination step is not mean preserving.

Remark C.6 (Event-trigger liveness). The fallback rule in Algorithm 3 forces a client to share after at most $N_{\max}$ silent local rounds. Hence, event-triggered communication cannot be suppressed indefinitely. This liveness property is useful for maintaining interaction among replicas, but it does not by itself establish Assumption C.5 or a uniform bound on D_t.

C.5 One-Step Descent

Lemma C.7 (Conditional expected descent)**.** *Under Assumptions C.1–C.5, if $\eta \le 1/(16L)$, then*

$$\mathbb{E}[F(\bar{w}_{t+1}) \mid \mathcal{F}_t] \le F(\bar{w}_t) - \frac{3\eta}{8}\|\nabla F(\bar{w}_t)\|^2 + 3\eta(C_D + C_{\mathrm{coord}})D_t + 3\eta\varepsilon_\tau + 3\eta\varepsilon_{\mathrm{aux}} + 3\eta\varepsilon_{\mathrm{coord}} + 4L\eta^2\sigma^2. \tag{17}$$

Proof. Let

$$G_t := \nabla F(\bar{w}_t), \qquad b_t := g_t - G_t.$$

By Assumption C.2, $\widetilde{g}_t = G_t + b_t + \xi_t$ with $\mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0$. Using Eq. (13), define

$$\Delta_t := \bar{w}_{t+1} - \bar{w}_t = -\eta(G_t + b_t + \xi_t + a_t) + \delta_t^{\mathrm{coord}}.$$

By L-smoothness,

$$F(\bar{w}_{t+1}) \le F(\bar{w}_t) + \langle G_t, \Delta_t\rangle + \frac{L}{2}\|\Delta_t\|^2. \tag{18}$$

Taking conditional expectation and using the zero conditional mean of ξ_t gives

$$\mathbb{E}[\langle G_t, \Delta_t\rangle \mid \mathcal{F}_t] = -\eta\|G_t\|^2 - \eta\langle G_t, b_t\rangle - \eta\,\langle G_t, \mathbb{E}[a_t \mid \mathcal{F}_t]\rangle + \left\langle G_t, \mathbb{E}[\delta_t^{\mathrm{coord}} \mid \mathcal{F}_t]\right\rangle. \tag{19}$$

Young's inequality, Jensen's inequality, and $\|\mathbb{E}[X \mid \mathcal{F}_t]\|^2 \le \mathbb{E}[\|X\|^2 \mid \mathcal{F}_t]$ yield

$$-\eta\langle G_t, b_t\rangle \le \frac{\eta}{8}\|G_t\|^2 + 2\eta\|b_t\|^2, \tag{20}$$

$$-\eta\,\langle G_t, \mathbb{E}[a_t \mid \mathcal{F}_t]\rangle \le \frac{\eta}{8}\|G_t\|^2 + 2\eta\mathbb{E}[\|a_t\|^2 \mid \mathcal{F}_t], \tag{21}$$

$$\left\langle G_t, \mathbb{E}[\delta_t^{\mathrm{coord}} \mid \mathcal{F}_t]\right\rangle \le \frac{\eta}{8}\|G_t\|^2 + \frac{2}{\eta}\mathbb{E}[\|\delta_t^{\mathrm{coord}}\|^2 \mid \mathcal{F}_t]. \tag{22}$$

Therefore,

$$\mathbb{E}[\langle G_t, \Delta_t\rangle \mid \mathcal{F}_t] \le -\frac{5\eta}{8}\|G_t\|^2 + 2\eta\|b_t\|^2 + 2\eta\mathbb{E}[\|a_t\|^2 \mid \mathcal{F}_t] + \frac{2}{\eta}\mathbb{E}[\|\delta_t^{\mathrm{coord}}\|^2 \mid \mathcal{F}_t]. \tag{23}$$

For the quadratic smoothness term, using $\|u+v\|^2 \le 2\|u\|^2 + 2\|v\|^2$ and $\|x_1+x_2+x_3+x_4\|^2 \le 4\sum_{j=1}^4 \|x_j\|^2$ gives

$$\mathbb{E}[\|\Delta_t\|^2 \mid \mathcal{F}_t] \le 8\eta^2\left(\|G_t\|^2 + \|b_t\|^2 + \sigma^2 + \mathbb{E}[\|a_t\|^2 \mid \mathcal{F}_t]\right) + 2\mathbb{E}[\|\delta_t^{\mathrm{coord}}\|^2 \mid \mathcal{F}_t]. \tag{24}$$

Substituting Eqs. (23) and (24) into Eq. (18) yields

$$\mathbb{E}[F(\bar{w}_{t+1}) \mid \mathcal{F}_t] \le F(\bar{w}_t) - \left(\frac{5\eta}{8} - 4L\eta^2\right)\|G_t\|^2 + (2\eta + 4L\eta^2)\|b_t\|^2 + (2\eta + 4L\eta^2)\mathbb{E}[\|a_t\|^2 \mid \mathcal{F}_t] + \left(\frac{2}{\eta} + L\right)\mathbb{E}[\|\delta_t^{\mathrm{coord}}\|^2 \mid \mathcal{F}_t] + 4L\eta^2\sigma^2. \tag{25}$$

If $\eta \le 1/(16L)$, then $5\eta/8 - 4L\eta^2 \ge 3\eta/8$, $2\eta + 4L\eta^2 \le 3\eta$, and $(2/\eta + L)\eta^2 \le 3\eta$. Applying Assumptions C.3, C.4, and C.5 gives Eq. (17). □

C.6 Proof of Theorem 3.1

Proof. Taking total expectation in Lemma C.7, summing from $t = 0$ to $T-1$, and using $F(\bar{w}_T) \geq F^\star$ gives

$$\frac{3\eta}{8} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(\bar{w}_t)\|^2 \leq F(\bar{w}_0) - F^\star + 3\eta(C_D + C_{\text{coord}}) \sum_{t=0}^{T-1} \mathbb{E}[D_t] + 3\eta T\left(\varepsilon_\tau + \varepsilon_{\text{aux}} + \varepsilon_{\text{coord}}\right) + 4L\eta^2 T\sigma^2. \tag{26}$$

Dividing by $3\eta T/8$ yields the explicit bound

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(\bar{w}_t)\|^2 \leq \frac{8(F(\bar{w}_0) - F^\star)}{3\eta T} + 8(C_D + C_{\text{coord}}) \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[D_t] + 8\varepsilon_\tau + 8\varepsilon_{\text{aux}} + 8\varepsilon_{\text{coord}} + \frac{32L}{3}\eta\sigma^2. \tag{27}$$

This is Eq. (4) after renaming constants. □

C.7 Interpretation and Scope

Equation (27) separates five sources of residual error. The first term is the finite-horizon optimization term and decays as $O(1/(\eta T))$. The second term is the time-averaged disagreement among distributed bottom- and top-model replicas. The third term captures the bias caused by stale and locally evaluated split gradients. The fourth term captures the fact that local auxiliary updates need not be gradients of the main SFL objective. The fifth coordination term captures average drift from non-mean-preserving gossip, participation-weighted committee consensus, and role changes. Finally, stochastic gradient noise yields the usual constant-step residual proportional to $\eta\sigma^2$.

In particular, if the time-averaged replica disagreement is bounded by $\overline{D}$, then

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla F(\bar{w}_t)\|^2 \leq 8(C_D + C_{\text{coord}})\overline{D} + 8(\varepsilon_\tau + \varepsilon_{\text{aux}} + \varepsilon_{\text{coord}}) + \frac{32L}{3}\eta\sigma^2.$$

Thus, under the stated effective conditions, the averaged iterate is driven to a residual neighborhood of stationarity whose size is explicitly controlled by disagreement, stale-gradient bias, auxiliary-update perturbations, coordination-induced average drift, and stochastic gradient noise.

The analysis does *not* establish that the particular event-trigger rule alone guarantees Assumption C.5, nor does it establish a uniform disagreement bound for every asynchronous network schedule. Proving such properties would require additional assumptions on the time-varying communication graph and, for exact average preservation, a mean-preserving mixing mechanism such as a doubly-stochastic construction or an appropriate push-sum correction. The purpose of the present result is therefore to state precisely the optimization conditions under which the implemented asynchronous architecture admits a stationarity guarantee, without attributing properties to the receiver-side gossip operator that it does not generally possess.

C.8 Complexity Analysis

Let P_{bot}, P_{top}, and P_{aux} denote the numbers of bottom-, top-, and auxiliary-head parameters, and let d_h be the smashed-activation dimension per sample. For batch size B, each client–committee interaction communicates $O(Bd_h)$ activations and $O(Bd_h)$ activation gradients. Since each client contacts K peers per gossip phase, the communication per shared model is $O(KP_{\text{bot}})$ per client. A committee synchronization among M members costs $O(MP_{\text{top}})$ to $O(M^2P_{\text{top}})$, depending on the synchronization primitive. A role rotation transfers $O(P_{\text{top}} + P_{\text{bot}} + P_{\text{aux}})$ parameters per swap. Under balanced assignment, each committee member serves approximately N/M clients and therefore handles an $O(1/M)$ fraction of the online top-model workload of a centralized split

learning server, i.e., its bandwidth load drops from $O(N \cdot Bd_h)$ to $O(\frac{N}{M} \cdot Bd_h)$ per round. Since the committee rotates, this workload is further spread across all N devices over time instead of being concentrated on the same M devices, which may also help slow down battery drainage on specific devices.

D Additional Results

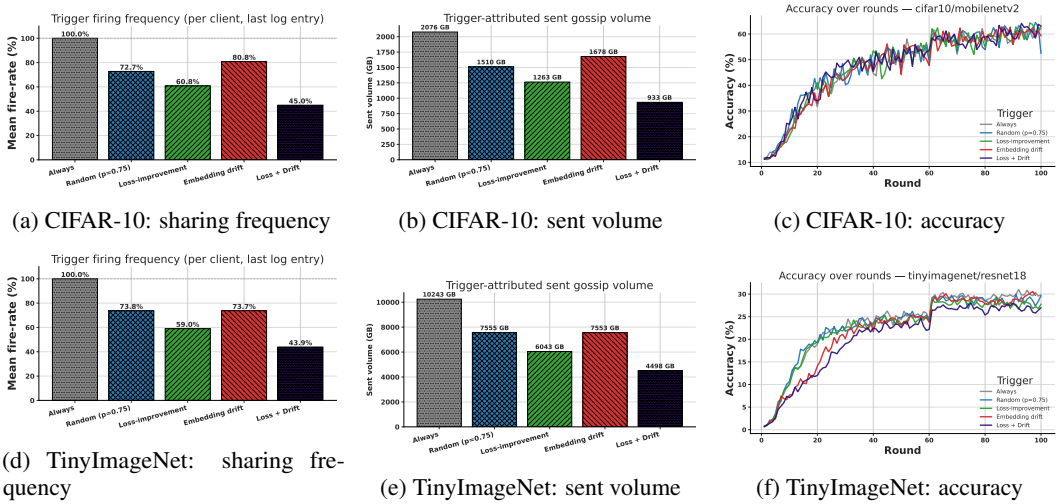
Auxiliary network. Figure 9 complements Figure 4 with the two remaining settings. On CIFAR-10 with MobileNetV2 the auxiliary network both accelerates training, reaching 30% accuracy 2.5× earlier, and improves the final accuracy by 3.6 points. On MedMNIST with EfficientNet-B0 the acceleration is the strongest we observe 70% accuracy at round 14 instead of round 61 — while the final accuracy ends 1.9 points lower. The benefit of the mechanism is therefore consistent in convergence speed across all three settings, whereas its effect on final accuracy depends on the task.



(a) CIFAR-10 (b) MedMNIST

Figure 9: Impact of the auxiliary network.

Communication efficiency. Figure 10 shows, per trigger, the fraction of rounds in which a client shares its model, the resulting sent gossip volume, and the accuracy over rounds. The ordering of the triggers in terms of communication is consistent across the smaller MobileNetV2 on CIFAR-10 and the larger ResNet-18 on TinyImageNet, while their effect on accuracy differs as discussed in § 4.1.



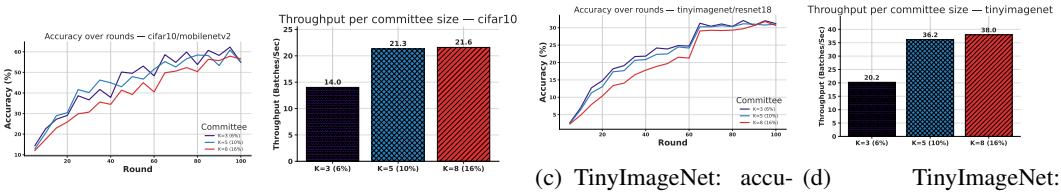
(a) CIFAR-10: sharing frequency (b) CIFAR-10: sent volume (c) CIFAR-10: accuracy

(d) TinyImageNet: sharing frequency (e) TinyImageNet: sent volume (f) TinyImageNet: accuracy

Figure 10: Communication triggers on CIFAR-10 (MobileNetV2, top) and TinyImageNet (ResNet-18, bottom).

Committee size. Figure 11 shows the corresponding curves for varying committee sizes.

Staleness factors. Figure 12 shows the full sweep of the staleness exponents under severe stragglers. The green band marks the accuracy range without stragglers.



(a) CIFAR-10: accuracy (b) CIFAR-10: throughput (c) TinyImageNet: accuracy (d) TinyImageNet: throughput

Figure 11: Sensitivity of *ARC-SFL* to the committee size.

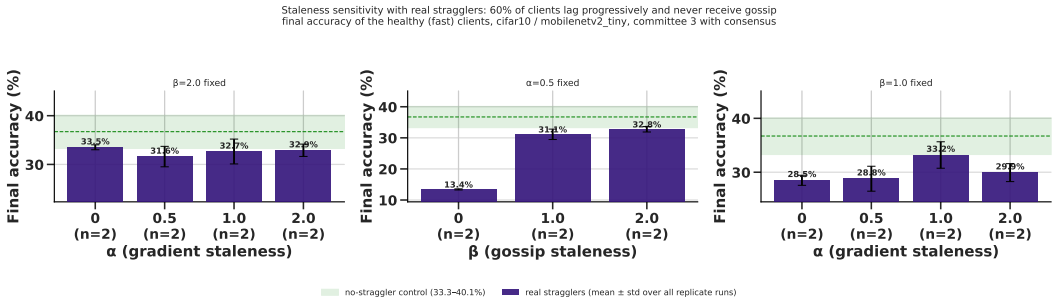


Figure 12: Sensitivity to the gradient (α, left) and gossip (β, right) staleness exponents of Eq. (**??**) with 60% of clients being severe stragglers.

E Privacy Analysis

Since *ARC-SFL* utilizes a rotating committee of peers, a potential threat arises if a malicious device is elected to the committee. To quantify the privacy leakage of the transmitted smashed activations, we conduct a Feature Reconstruction Attack (Model Inversion). We assume a semi-honest adversary acting as a committee member who trains a decoder network D to reconstruct the raw input X from the received smashed activations A, similar to model inversion attacks on collaborative inference and split learning (He et al., 2019; Pasquini et al., 2021), minimizing the Mean Squared Error (MSE) $\mathcal{L}_{\text{rec}} = \|X - D(A)\|^2$. Table 7 reports the Structural Similarity Index (SSIM) between original and reconstructed images, where a lower SSIM indicates better privacy preservation. Our results on MedMNIST with EfficientNetB0 and CIFAR-10 with MobileNetV2 indicate that while the smashed activations in *ARC-SFL* retain some high-level structural information that the top model needs to complete its task correctly, they successfully obfuscate fine-grained private details, likely due to the bottom model's non-linear transformations. Figure 13 provides visual inspection.

Table 7: Privacy leakage analysis.

Dataset	**MSE** ↓	**SSIM** ↓
CIFAR-10	0.7971	0.1750
MedMNIST	0.1016	0.1441

F Discussion, Limitations, and Broader Impact

Through our experiments, we conclude that *ARC-SFL* provides a fully decentralized and asynchronous approach to split federated learning. It achieves accuracy comparable to centralized methods across multiple datasets and model architectures, is communication-efficient, and is robust to both committee and client failures, which eliminates single points of failure and allows the system to recover quickly from transient disruptions. By distributing the top-model workload across multiple committee members, *ARC-SFL* achieves near-linear throughput scaling, in contrast to the minimal gains of a single centralized server without load balancing. The auxiliary network is in line with the cycle

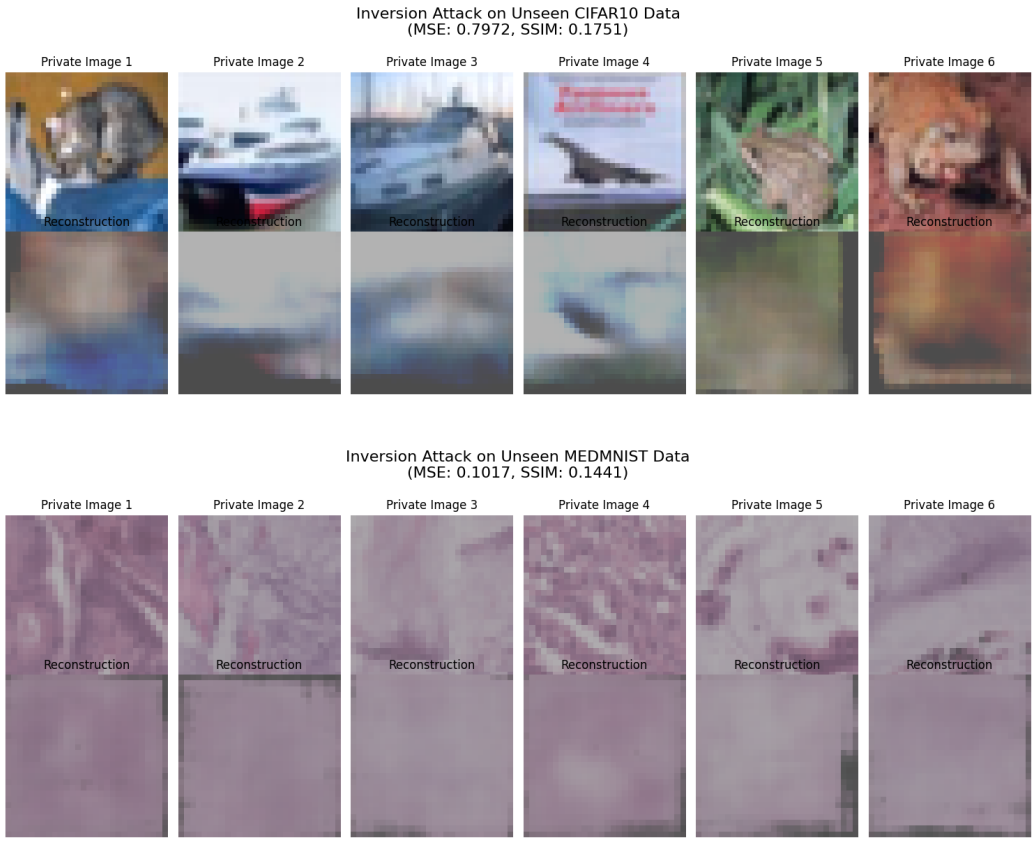
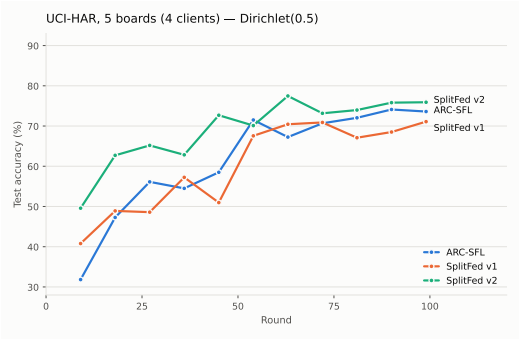


Figure 13: Visual reconstruction of client data by a malicious committee member. Top row: original images; bottom row: reconstructed images from smashed activations.

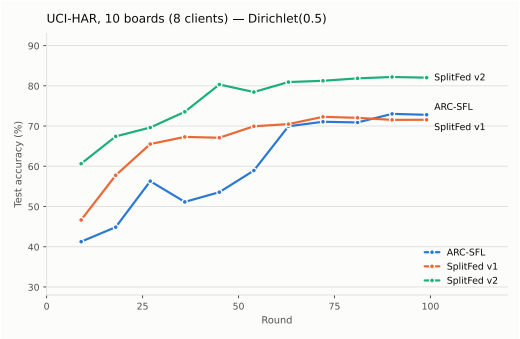
stealing concept; by effectively utilizing idle client time, *ARC-SFL* utilizes available computational resource more efficiently at an overhead of only $\sim 5\%$ of the top model.

However, these design choices also introduce trade-offs. Temporary accuracy drops occur during committee failures, although the system consistently recovers. Larger committees raise throughput but reduce accuracy, so we choose the committee size according to the application. Event-triggered gossip reduces communication substantially for compact models, but larger models tolerate less aggressive filtering, which calls for adaptive triggers. Further, *ARC-SFL* has several hyperparameters, e.g., the staleness exponents, the round-close threshold, and the rotation period, and reducing this sensitivity would make it easier to apply across use cases. Finally, communication remains the main bottleneck on tiny devices, which motivates exchanging only selected model segments instead of full bottom models.

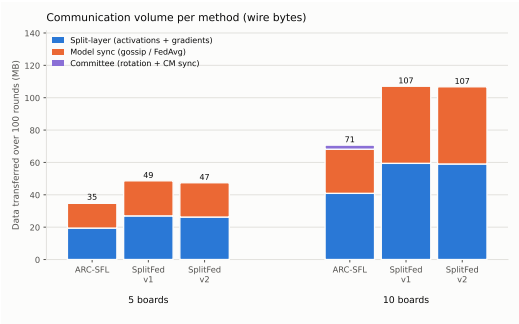
Broader impact. *ARC-SFL* facilitates the training of machine learning models on resource-constrained IoT devices without the need for centralized data aggregation. ⊕ Our approach enables many verticals such as healthcare and smart-city applications in areas with poor connectivity or strict privacy regulations to learn from other data sources across sectors and improve their accuracy without the need to share raw private data, instead of relying only on their own local available data. By utilizing idle computation, i.e., auxiliary training, our method further improves convergence speed, and potentially energy efficiency, compared to synchronous protocols that suffer from waiting times. ⊖ We eliminate the need for a central server that aggregates and orchestrates. Instead we use a group of devices to form a committee and take on the responsibility for aggregation. In Appendix E, we empirically show that *ARC-SFL* preserves privacy, mainly because we integrate split learning and instead of sharing the raw data, share an embedded representation with compressed information mostly useful for performing the particular task. Further, since the compute capability



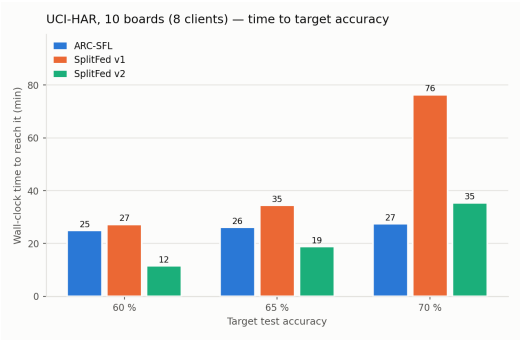
(a) Accuracy, 5 boards



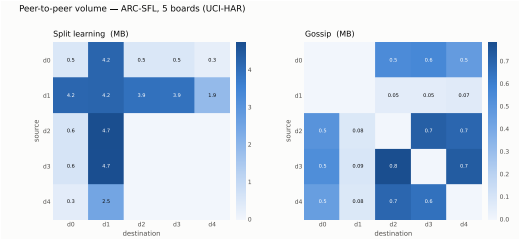
(b) Accuracy, 10 boards



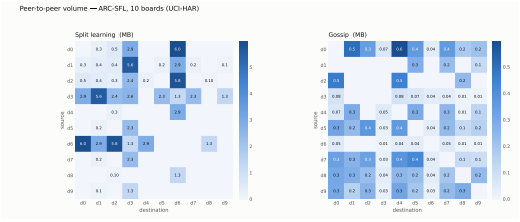
(c) Communication volume over 100 rounds



(d) Committee/server wall-clock to round 100

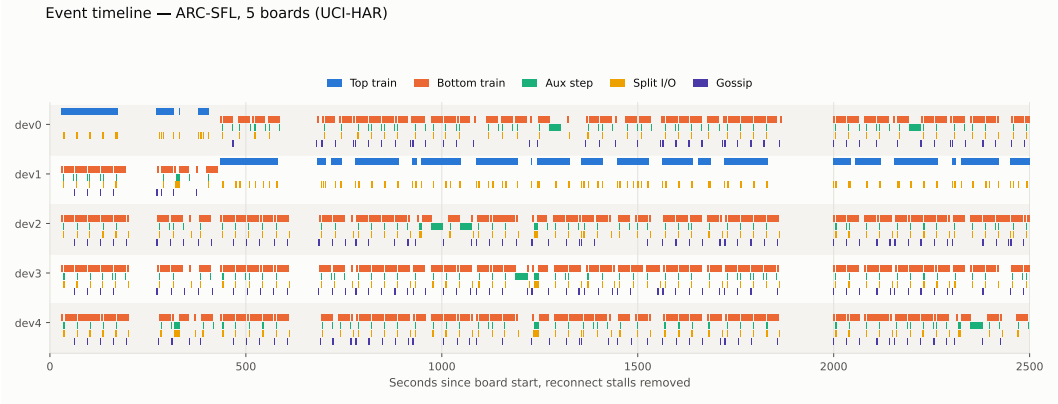


(e) *ARC-SFL* peer-to-peer traffic, 5 boards

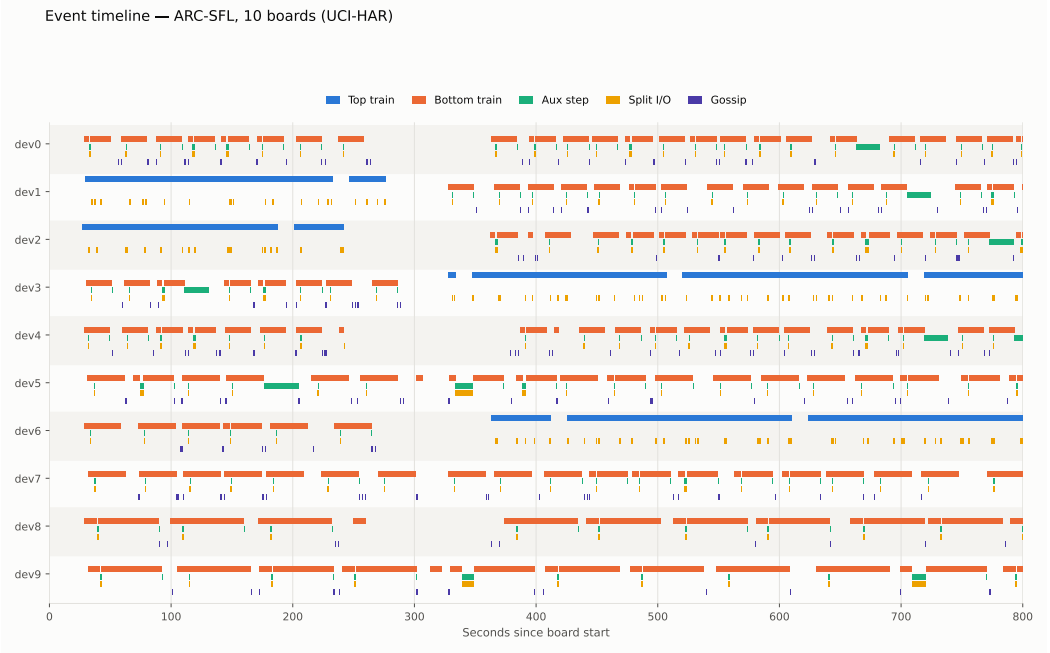


(f) *ARC-SFL* peer-to-peer traffic, 10 boards

Figure 14: *ARC-SFL* against SplitFed v1 and v2 on the ESP32-S3 cluster, UCI-HAR with a Dirichlet($\alpha = 0.5$) split. Accuracy is that of the averaged bottom model paired with the averaged top model on the full 2 947-sample test set. One run per cell, selected by a fault audit (no reboots, no panics).



(a) *ARC-SFL* event timeline, 5 boards (reconnect stalls removed)



(b) *ARC-SFL* event timeline, 10 boards

Figure 15: Gantt chart of events on real devices.

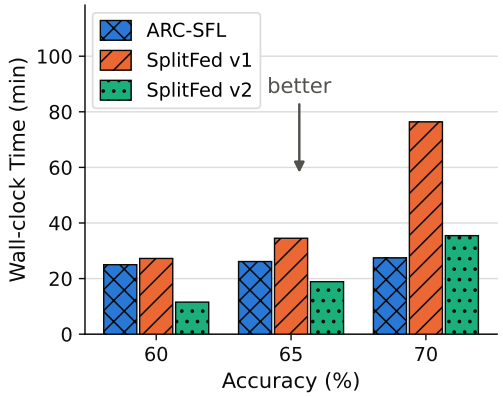


Figure 16: Wall-clock time for the committee / server to first reach an accuracy threshold on the 10-board fleet (UCI-HAR, averaged bottom model with averaged top model, interpolated between checkpoints).

of clients and committee member is similar, the split is more balanced, which leads to even more abstract intermediate representation. However, compared to scenarios where the server is trusted and fully secure, which is a weak argument and unrealistic in the majority of real-world cases, there might be malicious actors that could potentially exploit the lack of a central authority to poison models (Bagdasaryan et al., 2020) or reconstruct private data (Pasquini et al., 2021). Nevertheless, our rotation mechanism mitigates specific targeting. Future work should investigate this further.

G Extended Related Work

This section complements Section 2 and positions *ARC-SFL* with respect to the broader literature.

Federated optimization under heterogeneity. FedAvg (McMahan et al., 2017) performs several local SGD steps between aggregations, which saves communication but lets local models drift apart under non-IID data. Li et al. (2020d) establish the convergence of FedAvg on non-IID data and show that a decaying learning rate is necessary. FedProx (Li et al., 2020c) limits local drift with a proximal term, SCAFFOLD (Karimireddy et al., 2020) corrects it with control variates, FedNova (Wang et al., 2020) normalizes the updates of clients that run different numbers of local steps, and Reddi et al. (2021) apply adaptive optimizers on the server side. Bonawitz et al. (2019) describe the design of a production FL system, including device availability and dropouts. All these methods assume a central server holding the global model, and every client trains the full model.

Decentralized and gossip learning. Decentralized optimization builds on distributed subgradient methods (Nedić & Ozdaglar, 2009) and randomized gossip averaging (Kempe et al., 2003; Boyd et al., 2006). Lian et al. (2017) show that decentralized parallel SGD can match the convergence rate of its centralized counterpart while removing the communication hotspot at the server. Follow-up work covers changing topologies and local updates (Koloskova et al., 2020), directed and time-varying graphs (Assran et al., 2019), compressed gossip (Koloskova et al., 2019), non-IID data (Tang et al., 2018; Hsieh et al., 2020), and federated averaging with momentum over graphs (Sun et al., 2023). Kong et al. (2021) identify the consensus distance between replicas as the quantity that decides whether decentralized training reaches centralized accuracy, which corresponds to the replica-disagreement term D_t in our analysis. Gossip learning (Ormándi et al., 2013; Hegedűs et al., 2021; Danner et al., 2023) and Moshpit SGD (Ryabinin et al., 2021) target unreliable peers. Closest to our committee, Che et al. (2022) elect a committee of clients that screens local updates for aggregation in serverless FL and is periodically re-elected. However, all these methods replicate the full model on every device, and the committee of Che et al. (2022) acts as a filter against Byzantine clients rather than executing part of the model. Fully decentralized SFL exists, but to our knowledge only in blockchain-based form (Sokhankhosh et al., 2025).

Asynchronous training and staleness. Asynchronous SGD removes synchronization barriers in shared-memory (Recht et al., 2011) and parameter-server training (Dean et al., 2012), and converges for non-convex objectives under bounded delays (Lian et al., 2015). Zhang et al. (2016) scale the learning rate of each gradient inversely with its staleness, the same principle behind our gradient discount ϕ_α, whereas delay compensation (Zheng et al., 2017) corrects stale gradients through a Taylor approximation. Recent analyses show that asynchronous SGD converges at rates that do not depend on the maximum delay (Mishchenko et al., 2022; Koloskova et al., 2022). In FL, FedAsync (Xie et al., 2019) mixes each arriving update with a staleness-dependent weight, FedBuff (Nguyen et al., 2022) buffers updates before aggregation, and Fraboni et al. (2023) provide a unified theory for asynchronous and heterogeneous client updates. Tier-based and semi-asynchronous protocols group clients by speed to mitigate stragglers (Chai et al., 2020; 2021; Wu et al., 2021), and Oort (Lai et al., 2021) selects participants by their statistical and system utility. All of them rely on a server to buffer, tier, or select updates. *ARC-SFL* applies staleness discounts on both sides of the split without such a server.

Communication-efficient and event-triggered exchange. Compression reduces the size of each message through quantization (Alistarh et al., 2017), sparsification with error feedback (Stich et al., 2018; Lin et al., 2018), or sparsified gossip (Koloskova et al., 2019; Tang et al., 2023). An orthogonal line of work decides *whether* to communicate at all. Gaia (Hsieh et al., 2017) synchronizes only significant updates across data centers, LAG (Chen et al., 2018) reuses outdated gradients when new ones change little, and CMFL (Wang et al., 2019a) skips client updates that do not align with the global trend. Event-triggered methods transmit only when the local state deviates sufficiently from the last shared one (George & Gurram, 2020; Zehtabi et al., 2022; Singh et al., 2023), and Wang et al. (2019b) adapt the aggregation frequency to the resource budget of edge devices. *ARC-SFL* follows the event-triggered paradigm but combines a loss-progress check with a representation-drift criterion on the smashed activations, which is specific to the split setting and complementary to compression.

Split learning and split federated learning. Beyond vanilla SL (Gupta & Raskar, 2018; Vepakomma et al., 2018) and SplitFed (Thapa et al., 2022), whose variants can additionally integrate differential privacy methods such as PixelDP (Lecuyer et al., 2019), a large body of work reduces the latency of serving many clients. CPSL (Wu et al., 2023) trains clients in parallel within a cluster and sequentially across clusters, EPSL (Lin et al., 2024) aggregates last-layer activation gradients to shrink server-side backpropagation, and LocFedMix-SL (Oh et al., 2022) combines local parallelism, federated averaging, and mixup augmentation. ParallelSFL (Liao et al., 2024) and ClusterSFL (Arafeh et al., 2025) group clients into clusters to address heterogeneity, RingSFL (Shen et al., 2024) arranges clients in a ring, and CHEESE (Cheng et al., 2023) splits the model across neighboring devices via device-to-device links before aggregating at a base station. SplitGP (Han et al., 2023) trains the client-side model for personalization and the server-side model for generalization, and Han et al. (2024) analyze the convergence of SFL on heterogeneous data. To decouple clients from server gradients, Han et al. (2021) update the client-side model with a local auxiliary loss, and FedGKT (He et al., 2020) distills small client models into a large server-side model. However, all these methods depend on a permanent server or base station, and the only serverless variant we are aware of (Tedeschini et al., 2024) relies on synchronous, sequential consensus among devices.

Local learning and auxiliary objectives. Training parts of a network with local losses avoids waiting for the backward pass of later layers. Weston et al. (2008) add auxiliary embedding losses to intermediate layers, synthetic gradients (Jaderberg et al., 2017) predict the missing error signal, and decoupled parallel backpropagation (Huo et al., 2018) uses delayed gradients with convergence guarantees. Greedy and decoupled greedy training with auxiliary classifiers scales to ImageNet (Belilovsky et al., 2019; 2020), and local error signals (Nøkland & Eidnes, 2019) and gradient-isolated contrastive objectives (Löwe et al., 2019) train blocks independently. Wang et al. (2021) show that purely local objectives tend to discard task-relevant information in early layers. *ARC-SFL* therefore uses its auxiliary head only to fill idle time and combines its gradient with the split learning gradient instead of replacing it.

Split computing and distributed inference. Split computing partitions DNN inference between devices and edge or cloud servers (Kang et al., 2017; Hu et al., 2019; Li et al., 2020a), often together with early exits that provide auxiliary outputs at the end of the first split to improve latency and throughput (Teerapittayanon et al., 2016; Huang et al., 2018; Kaya et al., 2019; Laskaridis et al., 2020). Although early exits may lower accuracy, confidence estimation lets inputs traverse the full model when necessary (Guo et al., 2025). DDNN (Teerapittayanon et al., 2017) and DeepThings (Zhao

et al., 2018) distribute inference across hierarchies and clusters of IoT and edge devices. These works target inference, whereas *ARC-SFL* targets collaborative training.

Training on resource-constrained devices. MCUNet (Lin et al., 2020a) co-designs network architectures and inference engines for microcontrollers, and on-device training reduces memory by updating only biases and lightweight residual modules (Cai et al., 2020) or sparse subsets of layers (Lin et al., 2022). In FL, model-heterogeneous methods let weak clients train sub-models of the global model, e.g., width-reduced subnetworks (Diao et al., 2021), ordered dropout (Horváth et al., 2021), or rolling sub-model extraction (Alam et al., 2022), and ensemble distillation fuses heterogeneous client models on the server (Lin et al., 2020b). These methods shrink the model that a client trains and still require a server, while SL keeps the full model and offloads its later layers. Further works evaluate FL and SL on IoT devices (Gao et al., 2022) and deploy FL on microcontrollers (Wulfert et al., 2023b;a; Kopparapu et al., 2022; Giménez et al., 2023; Asadi et al., 2025).

Decentralized model-parallel training. Pipeline parallelism splits a model across accelerators (Huang et al., 2019; Narayanan et al., 2019). Closer to our setting, Yuan et al. (2022) schedule model-parallel training of foundation models over heterogeneous, geo-distributed devices, and SWARM parallelism (Ryabinin et al., 2023) routes activations through randomized pipelines of unreliable peers and rebalances them at runtime. Similar to *ARC-SFL*, these systems let peers host parts of the model and tolerate failures. However, they target large models on GPU-class peers without data-privacy constraints, whereas *ARC-SFL* keeps data local on tiny devices and combines model splitting with federated aggregation.

Privacy and robustness. Shared gradients in FL can leak training data (Zhu et al., 2019; Geiping et al., 2020). In SL, a curious or malicious server can reconstruct inputs from smashed activations (He et al., 2019; Pasquini et al., 2021), and labels can leak through the gradients exchanged at the cut layer (Li et al., 2022). Differential privacy (Abadi et al., 2016) and secure aggregation (Bonawitz et al., 2017) are standard defenses and are orthogonal to *ARC-SFL*. Byzantine-robust aggregation (Blanchard et al., 2017; Yin et al., 2018) and defenses against backdoor attacks (Bagdasaryan et al., 2020) become relevant once committee members may be malicious, which we leave for future work (Appendix F).