FreezeFL: Accelerating Federated Learning in Heterogeneous Edge Devices by Layer Freezing

Yu-Min Chou, Fu-Chiang Chang, Jerry Chou
Computer Science Department
National Tsing Hua University
Hsinchu, Taiwan

***Abstract*—Federated learning is an emerging paradigm that enables edge devices to collaboratively train a model without sharing private data. There are two key challenges in the setting. 1) system heterogeneity, the significant variability in hardware resources on edge devices. 2) statistical heterogeneity, non-identically distributed data on edge devices. In this work, we propose a framework, FreezeFL, which alleviates the system heterogeneity by applying layer freezing and a fairness guaranteed client selection algorithm. We further show that FreezeFL does not exacerbate statistical heterogeneity while solving system heterogeneity and keeps client-oblivious; namely, clients do not need to share information besides model weights. Extensive experiment results on FEMNIST and Shakespeare show that FreezeFL outperforms various existing methods. Relative to FedAvg, FreezeFL shortens round length by 33% and 46%, respectively, while not hurting test accuracy.**

***Index Terms*—Federated learning, Statistical heterogeneity, system heterogeneity, Layer freezing, partial model training**

I. Introduction

Federated Learning (FL) [26] is a distributed framework for deep learning without centralizing data and with privacy by default. It enables edge devices such as mobile phones or IoT sensors to participate in training without exposing private data. During the training, each client (i.e., edge device) uses local private data to train a local model and only exchanges model updates with servers, which aggregate all updates to form a single global model. Recently, several successful applications such as improving query suggestions of Google keyboard [42] and next word prediction [11] are derived under FL.

However, as deployments move from the data center to the edge devices, FL encounters two key challenges that differentiate it from traditional distributed learning: 1) *system heterogeneity*, the differences in terms of the communication and computation capabilities of the edge devices, and 2) *statistical heterogeneity*, the heterogeneity of data distributions where training data are not independent and identically distributed (Non-IID) [21], [32], [44] on the local devices. System heterogeneity causes a lower training efficiency because all the clients must wait for the slower clients, known as stragglers, to complete each training round. On the other hand, statistical heterogeneity causes a lower training accuracy due to statistical bias, such as client overfitting. To mitigate system heterogeneity, techniques have been proposed to reduce the participation of stragglers, such as dropping the clients that fail to complete local training tasks in time [2]. Nevertheless, these techniques could further exacerbate statistical heterogeneity and harm the training performance, especially when the stragglers hold high-quality data. Therefore, despite recent attempts [20], [25], [30], designing a FL algorithm being the best of both worlds remains an open problem.

In this work, we propose **FreezeFL**, which applies the layer freezing technique to tackle the aforementioned problem. Freezing layer can effectively reduce the communication and computing cost of stragglers because the weight of frozen layers can be eliminated from the backward computation and the client-server model exchange. Although the idea is simple and it has been applied to other learning objectives, such as transfer learning [14], [24], surprisingly, no existing work has applied it to solve the straggler problem in FL. The reason is that previous studies mainly focus on the freezing [8], [34] and pruning [16], [27] techniques in the units of neurons instead of layers due to the general belief (as stated by [8]) that coarse granularity (i.e., layer) cannot provide efficient solutions. To the best of our knowledge, only a few works [4], [23] applied layer freezing for fine tuning models, but they are not designed for FL learning environment and not aiming to solve system heterogeneity problem. Therefore, our main objective is to explore the applicability and efficiency of layer freezing technique for solving the system and statistical heterogeneity problem in FL. Our research not only provides a positive answer, but also achieves the following four important requirements in FL environment by carefully designing the algorithm for making the freezing decision during the training process.

(1) Low algorithm complexity: Unlike traditional centralized training, scalability is a critical requirement for FL as the number of clients can grow quickly, and the computing power of clients could be limited, especially in the cross-device setting [35]. Therefore, it is essential to keep the learning method lightweight and non-intrusive. Our approach avoids additional overhead or complexity by simply adjusting the number of frozen layers in client training without changing the model architecture or learning behavior. Also, by answering how many layers instead of which neuron to be frozen, our problem complexity is significantly reduced.

(2) Adaptation: Applying layer freezing in FL is challenging because we have to consider other concerns, such as client availability, fairness, and statistical heterogeneity, and the interaction or impact of these factors are complicated and

dynamic. To overcome these obstacles, both our client selection and layer freezing decisions are made adaptively according to the runtime information (i.e., local model weight and training time). By quantifying and considering the statistical importance and training time of each layer, our layer freezing algorithm is able to keep stragglers' contribution as much as possible while reducing stragglers' workload. By carefully controlling client selection probability with the warm-start technique, we can ensure the fairness of client participation and improve training efficiency. Hence, we can strike a better balance between system and statistical heterogeneity and be more resilient to client variability.

(3) Client oblivious: FL is a collaborative learning environment, so the server should have as little information about the clients as possible for security, privacy, and scalability reasons. Unfortunately, all previous system heterogeneity solutions [10], [13], [27] require explicit client knowledge about the local training time or computing capability. In contrast, FreezeFL avoids such requirements by deferring decisions to the local side and utilizing the layer freezing information. Therefore, our approach is client oblivious, which means the server does not need to know any status of clients that influences the local training time, such as the number of available resources and the battery level.

(4) Compatibility: FL often has to tackle multiple learning objectives (e.g., fairness, security, performance) in a learning environment with diverse characteristics. Layer freezing technical is a non-intrusive approach that skips the computation and communication of the frozen layer from training, so it can be easily combined with other training techniques for addressing multiple challenges.

To validate our approach, we conduct extensive experiments on two federated datasets with a heterogeneous setting to demonstrate the effectiveness of our frameworks. We empirically show that FreezeFL shortens round length up to 46%, improves test accuracy up to 2%, and speed up convergence rate up to $2\times$. Furthermore, we show that FreezeFL is compatible with other existing methods aiming to solve system heterogeneity and achieve better test accuracy and shorter round length.

II. Related work

In FL, edge devices are enabled to train a model collaboratively while keeping all private data on local devices and ensuring privacy [36], [37], [43]. Although FL has been proven feasible [18], [19], [26], [40], the characteristics of the FL environment bring many challenges, such as heterogeneous data and devices, privacy, and massively distributed devices.

Layer freezing Layer freezing is a representative technique for accelerating fine-tuning neuron networks in centralized learning [4], [23]. The methods exclude the parameters of layers from training in an input-output direction sequentially. Since most model architectures converge from the input side sequentially, layer freezing can accelerate training without degrading accuracy performance. However, the existing methods are not applicable to FL since FL is a very different environment from a centralized one. The freezing mechanism in FL should consider clients' limited capabilities and the trade-off between training efficiency and performance. Moreover, the information assumed known in centralized learning is usually unavailable in FL. In this work, we address the challenges mentioned above and introduce layer freezing into FL to solve system heterogeneity.

Partial model training Training a subset of a model is a common strategy in deep learning. Several works [3], [8], [31], [34], [39], [41] achieve communication-efficient FL by performing freezing/dropping in units of neurons and only uploading trainable parameters. By contrast, our work co-optimizes both communication and computation by performing layer freezing [4], [23], which excludes parameters from training in units of layers instead of neurons. In the past, layer freezing was considered inapplicable for FL due to its coarse operation granularity and the lack of freezing strategy [8]. In this work, we explore whether freezing in units of layers is applicable for FL and give a positive answer.

Heterogeneous local models The works about heterogeneous local models [6], [10], [13], [16], [27], [38] combine the ideas of partial model training and model pruning. In other words, they allow local models to have different architectures, usually a subset of the global model and still produce a single global model. It is natural to distribute subnetworks of various sizes according to clients' capabilities to solve system heterogeneity. However, such solutions depend highly on the model architecture and have high algorithm complexity. By contrast, FreezeFL does not need network-specific knowledge and has lower algorithm complexity since we do not modify the model architecture and simply decide the number of frozen layers. In addition, they remain orthogonal to our work and can be complementarily combined to reduce costs further.

Client selection Limiting the participation of stragglers effectively improves training efficiency. [29] proposed FedCS, which selects powerful clients greedily. [7] proposed TiFL, which selects clients with similar performance in each round to counterbalance the slowdown introduced by stragglers. However, they do not consider that an interplay exists between system and statistical heterogeneity and select clients too biased so that the accuracy performance may be degraded. Compared with them, our work provides a fairness guarantee in client selection to tackle the trade-off between training efficiency and model performance.

III. Preliminaries

Layer freezing. Layer freezing in deep learning means that the frozen layers' weights do not be modified by backward propagation. It is widely used in transfer learning [14], [24] for transferring domain knowledge to tasks with fewer data without overfitting. In this work, we explore its benefit on accelerating training. Layer freezing can save communication costs because only the weight of the unfrozen layers needs to be uploaded from clients to the server. On the other hand, in order to save computation cost, the layers must be frozen one by one from the first layer to the last layer in consecutive order so that the frozen

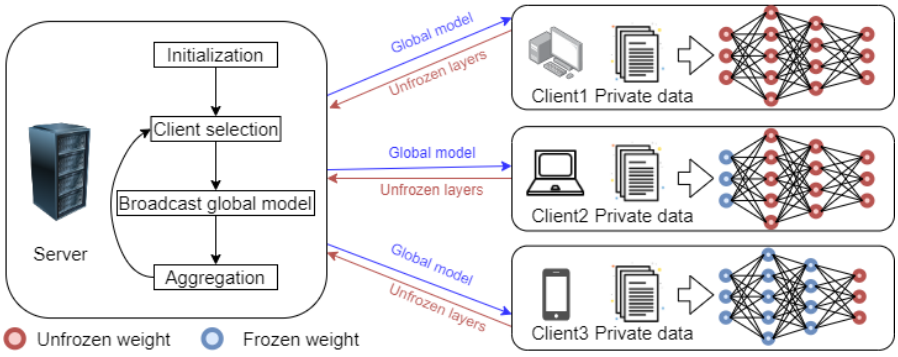
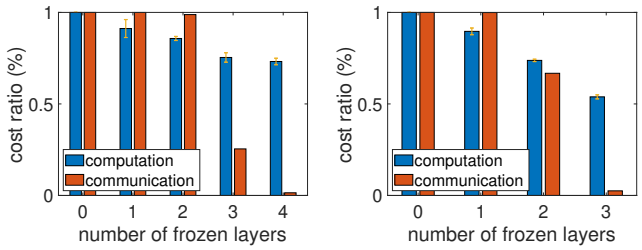


Fig. 1. The training procedure of FL with layer freezing. Clients with poorer capabilities will be frozen for more layers. In addition, we only upload unfrozen layers to reduce communication costs.



(a) CNN on FEMNIST (b) RNN on Shakespeare

Fig. 2. Cost ratio under a different number of frozen layers. The computation cost reduction is proportional to the computation complexity of frozen layers, and the communication cost reduction is proportional to the frozen layers' number of neurons.

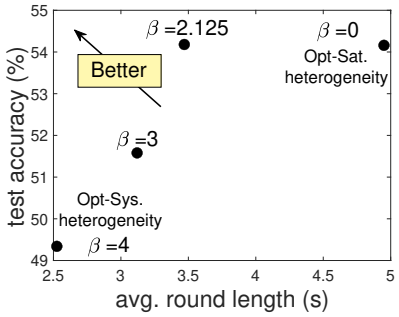


Fig. 3. FreezeFL explores the β to find the sweet spot between two heterogeneity. The numerical results are from the RNN on Shakespeare(experimental setting is detailed in Section. V-A)

layers can be excluded from the backward passes. As shown in Fig. 2, we evaluate the cost-saving of layer freezing on CNN and RNN and show that layer freezing can bring considerable cost reduction under different network architectures (The detail of the setup is summarized in Section. V-A).

FL with layer freezing. In this work, we propose to apply layer freezing in the FL training procedure as shown in Fig. 1. In FL, all clients train a model collaboratively. First, the server selects a subset of clients at each round and broadcasts the global model to them. Next, the selected clients perform training for some local epochs over local private datasets. Finally, the server collects and aggregates the updates to form an up-to-date global model. In the training procedure, the time cost of a round depends on the maximum of selected clients' model exchange time. We refer to the model exchange time as the time spent on a local training task, including model broadcasting, model training, and model uploading. In this work, we apply layer freezing to local training to shorten the stragglers' model exchange time. Each selected client will independently determine the number of frozen layers. Furthermore, we optimize the aggregation rule to account for the different degrees of freezing. Finally, we use the number of frozen layers reported by the clients as the reference for client selection to achieve the goal of client-oblivious.

Strength. Compared with the previous works that attempt to prune [16], [27] or freeze [8] of individual neural, layer freezing offers several advantages. (1) It does not alter the network architecture. Therefore, we do not deal with the problem of aggregating heterogeneous network models among clients. (2) The problem complexity is significantly reduced and oblivious to the model complexity because we simply decide the number of freezing layers of each client according to their computing capability. (3) The early layers tend to converge faster. Hence, freezing converged layers can potentially reduce computing cost without sacrificing training results.

Challenges. The key challenge in our approach is to decide the number of frozen layers of each client during the learning process. Since statistical heterogeneity will deteriorate and degrade model performance if we freeze too many layers rashly. On the other hand, system heterogeneity will not be improved if we freeze too few layers. In other words, the decision on the number of frozen layers is the trade-off between statistical and system heterogeneity. Therefore, the goal of our work is to propose a layer freezing strategy that can minimize additional statistical heterogeneity while solving the system heterogeneity problem.

IV. Algorithm

In this section, we first present the overview of FreezeFL, which addresses system heterogeneity while minimizing additional statistical heterogeneity in FL. Next, in Section IV-B, we detail how we adaptively decide the number of frozen layers for clients and the layerwise aggregation rule. Finally, Section IV-C

Algorithm 1 FreezeFL

Input: K, number of selected clients in each round. E, number of local epochs. η, learning rate. R, number of rounds, B. local minibatch size. Φ, the warm-restart interval.

Output: A trained global model.

1: **procedure** SERVEREXECUTE
2: Initialize $w_0, T_0, u_0^k, k \in 1, ..., N$
3: **for** each round $r = 1, 2, ..., R$ **do**
4: Sample a set S_r from N clients according to clients' utility $u_r^k, k \in 1, ..., N$
5: **for** each client $k \in S_r$ in parallel **do**
6: $(w_{r+1}^k, n^k) \leftarrow$ ClientUpdate(k, w_r, T_r)
7: Update soft deadline T_{r+1} as in Eq. 1
8: Update clients' utilities $u_{r+1}^k, k \in S_r$ as in Eq. 7
9: Update the global model w_{r+1} as in Eq. 4
10: **if** r mod Φ == 0 **then**
11: Update all clients' utilities $u_r^k, k \in 1, ..., N$ as in Eq. 8
12: **procedure** CLIENTUPDATE(k, w, T_r)
13: $w^k \leftarrow w$
14: $B_i \leftarrow$ Split local dataset X_i into batches of size B
15: **for** each local epoch $e = 1, ..., E$ **do**
16: **for** batch $b \in B_i$ **do**
17: $w^k \leftarrow w^k - \eta \nabla \ell(w; b)$
18: **if** $e == 1$ **then**
19: Calculate the importance of layers P as in Eq. 2
20: Freeze first n layers as in Eq. 3 and rollback their weights
21: return $\{w^{k,l}\}_{l \in n+1,...,L}, n$ to server

presents our novel reputation-based client selection scheme, which aims to address both system and statistical heterogeneity while only needing the client's layer freezing information.

A. Overview

The proposed framework, FreezeFL, is summarized at Algorithm. 1. Training proceeds as follows: for each round, the server performs client selection based on all clients' utility, namely, their training efficiency and quality of local datasets (line 4). Next, the server broadcasts global model weights and the soft deadline T_r to the selected clients (line 6), where the soft deadline T_r means the estimated average model exchange time at the rth round. On the client side, selected clients first perform training for a single local epoch without layer freezing to collect needed knowledge such as model weight update and further quantify the importance of each layer based on it (line 19, Eq 2). Afterward, each client determines the number of frozen layers according to the importance of each layer and comparison between its model exchange time and soft deadline (Eq 3). Subsequently, each client performs layer freezing, rollbacks the frozen layers' weight to that of the received global model (line 20), and finishes the remaining local epochs. Finally, after receiving all clients' responses, the server updates the soft deadline, clients' utility, and global model weights (line 7~11, Eq. 1, 4, 7, 8). We describe critical steps in further detail in the following subsections.

B. Layer freezing

Adaptive layer freezing. As mentioned in Section. III, determining the number of frozen layers is the tradeoff between two heterogeneity. Our goal is to solve system heterogeneity while not exacerbating statistical heterogeneity. Therefore, we try to quantify the effect of layer freezing on the above two heterogeneity and find the sweet spot between the tradeoff. For system heterogeneity, we maintain a criterion T_r, called soft deadline, to let clients determine if they are stragglers. In addition, Since the acceleration brought by layer freezing can be predicted, we can further estimate how much improvement each layer can bring to the system heterogeneity. The key insight we have for statistical heterogeneity is that discarding the model updates of clients is the source of additional statistical heterogeneity. Therefore, we quantify the importance of each layer in terms of weight update to measure the deterioration caused by layer freezing for statistical heterogeneity. More specific descriptions and formulas are as follows.

During the training process, the only information provided by the server to the clients is soft deadline T_r, more specifically, the estimated average model exchange time at the rth round. In our algorithm, T_r is used as a criterion to judge whether clients are stragglers. T_0 is customized by the developer. After each round, the server updates T_r with the average of selected clients' model exchange time by Exponential Moving Average (EMA). In the EMA equation below, the first parameter is the estimated value, and the second parameter is the current observation:

$$T_{r+1} = EMA(T_r, \frac{\sum_{k=1}^{K} t^k}{K}) \tag{1}$$

where r is the round index, K is the number of selected clients, t^k is the model exchange time of client k. As the number of rounds increases, the server accumulates more samples of model exchange time reported by clients. Therefore, the T_r will gradually approach the actual value.

Next, we detail the method of quantifying the importance of layers, which is motivated by Deep Gradient Compression (DGC) [22], [33]. In DGC, gradient magnitude is used as a heuristics for importance. Obliviously, since the larger gradient magnitude brings larger model weight updates, it is reasonable to use weight updates magnitude as another heuristics for importance. Based on the above insight, the metric of layer importance is formulated as follows:

$$P_l = \frac{||w^l - w^{k,l}||_1}{dim(w^l)}, \quad l \in \{1, ..., L\} \tag{2}$$

Where l is the layer index, L is the number of total layers, w is the received global model weights, w^k is client k's local model weights that have been trained for one local epoch, and $dim(.)$ is the number of neurons. This metric quantifies how much each layer contributes to the model training by calculating the average of absolute weight change, i.e., its ℓ_1-norm. For layers with more significant weight changes, we give them

higher importance scores and less chance of being frozen since freezing such layers is harmful to statistical heterogeneity.

Finally, we decide on the number of frozen layers based on the importance of layers and the model exchange time. To shorten the straggler's model exchange time while preserving their significant updates as much as possible, we should maximize the sum of unfrozen layers' importance scores with a time penalty. Each client obtains the specific number of frozen layers by the following formula:

$$\max_{n} \underbrace{\Sigma_{l=n+1}^{L} P_l}_{statistical} \times \underbrace{(\frac{T_r}{\tau_n})^{I(T_r<\tau_n)\times\beta}}_{system}, \quad n \in \{0, ..., L-1\} \quad (3)$$

where n means the number of frozen layers, τ_n means the model exchange time while freezing the first n layers, $I(T_r < \tau_n)$ is an indicator function where the output is 1 when the content is true, and 0 otherwise, and β is a hyperparameter handling the tradeoff between statistical heterogeneity and system heterogeneity. τ_n is calculated based on the performance profiling information collected from the previous training round at the client. In the formula, the clients whose model exchange time is longer than T_r receive the time penalty and thus increase the number of frozen layers to reduce their model exchange time. However, we still allow clients' model exchange time longer than T_r if the importance scores of the layers are large enough to offset the time penalty imposed by unfreezing the layers. Moreover, all clients unfreeze at least one layer, namely, the number of frozen layers n is up to $L-1$, to avoid entirely discarding the stragglers' model updates.

β is a purposely designed tuning knob in our algorithm to tackle the tradeoff between the two heterogeneities. In extreme settings, our algorithm does not freeze any layers ($\beta = 0$, as same as FedAvg [26]) to avoid introducing additional statistical heterogeneity and suffers from poor training efficiency. On the other hand, we can also freeze most of the straggler models ($\beta = \infty$) to address system heterogeneity but suffer from significant accuracy degradation. As shown in Fig. 3, the best results in accuracy and round length can be achieved under two extreme settings, respectively. Furthermore, the sweet spot that tackles the tradeoff does exist, and we can search for it through parameter tuning or multi-objective optimization algorithms. In summary, compared with other works, we perform better by quantifying two heterogeneity and exploring a 1-dimension tuning space.

Layerwise Aggregation. The heterogeneity of clients leads to heterogeneity in the degree of freezing. Therefore, involving the weights of the frozen layers in aggregation may make the updates too small and hurt the global model convergence. To this end, we perform aggregation in the following way:

$$w_{r+1}^{l} = \sum_{k} \frac{|D_k|}{\sum_k |D_k|} w_r^{k,l}, \quad k \in S_r^l \quad (4)$$

where $|D_k|$ means the size of client k's local dataset, and S_r^l means the set of clients which do not freeze layer l. For each layer, We only let clients which unfroze it participate in aggregation. Otherwise, the out-of-date weights of the previous round will slow down the model convergence and even reduce the test accuracy. We will show the advantage of layerwise aggregation in Section. V-B.

C. Fair reputation-based client selection

Given the highly variable FL environment, we address the following two concerns to select more valuable participants in terms of increasing test accuracy and shortening model exchange time. 1) *client-oblivious.* A client's utility can only be determined by its historical performance since reconfirming all clients' states before each round is not feasible. 2) *Fairness.* To avoid increasing statistical heterogeneity and account for the change in clients' utility; we should guarantee fairness in participation. To tackle the challenges, we develop a client selection algorithm that keeps client-oblivious through freezing-based reputation and guarantees fairness through the warm-restart mechanism.

Freezing-based reputation. In the client selection algorithm, our target is to determine clients' utility without client-specific information. A client's utility means its training efficiency and quality of the local dataset. The main idea is to obtain such information from a client's number of unfrozen layers and model weight updates reported by clients. For each client, the number of unfrozen layers is positively correlated with its computation and communication capability. Therefore, we cleverly use this information to estimate clients' training efficiency. In addition, we use the inner product between local model update and global model update to evaluate whether the client's local dataset is valuable. As a result, u_r^k, the utility of client k in round r is formulated as follows:

$$U_{sys}^k = (L - n^k) \quad (5)$$

$$U_{data}^k = max(0, \Sigma_{l=n^k+1}^{L} \frac{\langle w_{r+1}^{k,l} - w_r^l, w_{r+1}^l - w_r^l \rangle}{dim(w^l)}) \quad (6)$$

$$u_{r+1}^k = EMA(u_r^k, U_{sys}^k \times U_{data}^k) \quad (7)$$

where n^k is the number of frozen layers of client k, w_{r+1}^k is client k's uploaded model weights, w_{r+1} is global model weights, $\langle \cdot, \cdot \rangle$ is the inner product function, U_{sys}^k means the training efficiency, and U_{data}^k means the quality of the local dataset. Intuitively, a client with more unfrozen layers usually has a shorter model exchange time, which helps the training efficiency. Hence the number of unfrozen layers is proportional to the utility. On the other hand, the inner product considers both the direction and step size of the weight update. When the client's weight update direction is more similar to most others, it is more helpful to the training. A similar idea is also theoretically proven by Nguyen *et al.* [28], where a client's contribution in reducing the global loss is bound by the similarity of local gradient and global gradient. In FreezeFL, we extend this insight to the layer level and use the weight updates as an alternate. In addition, such an algorithm can effectively identify malicious clients uploading random weight updates. Through the inner product, when the update direction of the client does not match most people, we can significantly reduce the client's utility. Finally, the utility of a client is

updated by EMA with the product of the two factors above. In the client selection stage, FreezeFL samples clients with probability proportional to their utility.

Warm-restart. In our algorithm, a client's selection probability is proportional to its utility. As the training progresses, the utilities and selection probabilities of stragglers will be gradually decreased due to their low training efficiency. Although reducing stragglers' number of participation helps address system heterogeneity, excessively reducing it may introduce additional statistical heterogeneity. To tackle the tradeoff between two heterogeneity in client selection, we apply warm-restart to adjust client selection probability and further improve the fairness of client participation. The warm-restart mechanism means to rollback a value or state to an earlier one but not from scratch. Specifically, warm-restart adjusts the current selection probability distribution to be close to the uniform one every Φ rounds while satisfying the following two goals. First, we give the clients with lower utility larger utility increments to help them participate more in training and better improve fairness. Second, we give less change to the clients whose utility is already well certain because re-exploring clients' utility through Eq. 7 takes several rounds, and the improvement of training efficiency brought by client selection may decrease. The adjustment of the utility of client k is formulated as follows:

$$u^k = \begin{cases} min(\bar{u}, u^k + \sqrt{2\frac{\log \Phi}{\phi_k}}), & \text{if } u^k < \bar{u} \\ max(\bar{u}, u^k - \sqrt{2\frac{\log \Phi}{\phi_k}}), & \text{otherwise} \end{cases} \tag{8}$$

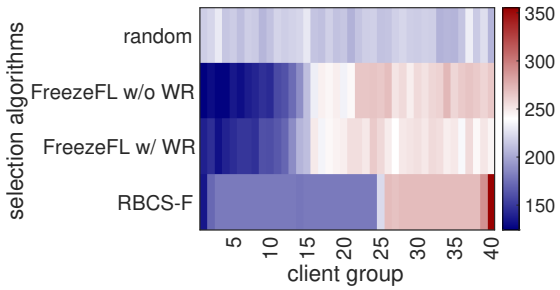
where $\bar{u}$ is the average of all clients' utility, and ϕ_k is client k's number of times of participation in the last Φ rounds. As observed from the equation, if the client's utility (u_k) is less than the average utility ($\bar{u}$), its value is increased; otherwise, it is decreased. The reputation-based utility estimation method described in Eq. 7 is similar to a multi-armed bandit problem [17]. Hence, our adjustment is based on the confidence bound of utility ($\sqrt{2\frac{\log \Phi}{\phi_k}}$) to ensure the two design goals mentioned above are satisfied. For the clients with more participation (i.e., larger ϕ_k) and higher utility, its confidence bound will be lower and hence receives a smaller decrement. Vice versa, for the client with less participation and lower utility, its confidence bound will be higher and hence receives a larger increment from our proposed equation.

V. Evaluation

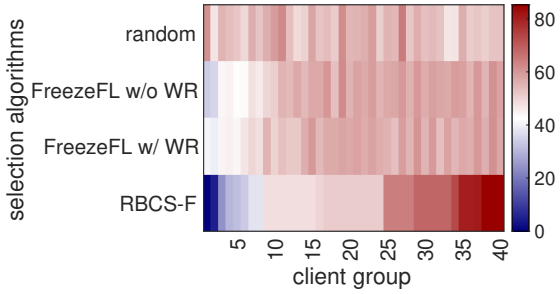
In this section, we provide a thorough evaluation of FreezeFL. We show that our framework solves the system heterogeneity, relieves the trade-off between system and statistical heterogeneity, and achieves superior performance. Also, we explore additional beneficial properties of FreezeFL.

A. Setup

Testing platform setting. We implement FreezeFL based on tensorflow [1] and LEAF [5], a benchmark for federated settings. Given that LEAF does not implement system heterogeneity, we add the feature ourselves. We randomly assign



(a) On FEMNIST



(b) On Shakespeare

Fig. 4. Number of participation of clients under different selection algorithms

capabilities to each client. Specifically, the distribution of capabilities follows a uniform distribution, and the ability gap is up to six times among all clients. Each client's computation and communication capabilities are proportional. Our experiments were conducted on an in-house cluster with two physical nodes. Each node has 2 Intel Xeon Silver 4110 CPU 2.10GHz, 4 V100 GPUs, and 128GB memory.

Datasets and models. The detail of datasets and hyperparameters in our framework are summarized in Table I. We conduct experiments on two different LEAF datasets: 1) FEMNIST dataset [9] for 62-class image classification. 2) Shakespeare dataset for next-character prediction. Both datasets are divided based on the writer or role to form non-IID partitions. For FEMNIST, we use a CNN with two convolutional layers followed by two fully connected layers and a softmax layer. For Shakespeare, we use a RNN with an embedding layer followed by two LSTM layer [12] and a softmax layer.

Baseline. We compare FreezeFL with FedAvg [26] and the following state-of-the-art solutions: 1) Fjord [13], a framework that addresses system heterogeneity by dynamically adapting model size through ordered dropout. 2) RBCS-F [15], a client selection algorithm that strikes a balance between training efficiency and fairness. It is worth noting that Fjord assumes clients' status is known, and RBCS-F predicts it by a Contextual Combinatorial Multi-Arm Bandit Model. Here, we make both solutions have clients' status set to known, and it is clear that the setting provides tougher baselines.

B. Experimental results

Sensitivity analysis of hyper-parameters First of all, we present the analysis of two important hyper-parameters, β and the number of local epochs E, in Table. II. As mentioned in

TABLE I
DATASETS AND MODELS

Dataset	Model	Total clients	Total samples	Samples per device		E	T_0	β	Φ
				mean	stdev				
FEMNIST	CNN	3550	805,263	227.4	88.9	3	1	4	60
Shakespeare	RNN	660	4,035,372	6114.2	7,184.1	3	4	2.125	30

TABLE II
SENSITIVITY ANALYSIS OF HYPER-PARAMETERS, AVERAGE AND STAND DEVIATION

		FEMNIST		Shakespeare	
Methods	Hyper-parameter	Best accuracy	Round length	Best accuracy	Round length
FreezeFL	$\beta = 4/2.25$	79.40 (0.24)	1.04 (0.03)	53.26 (0.34)	2.46 (0.13)
	$\beta = 6/2.5$	78.09 (0.16)	0.99 (0.01)	53.03 (0.13)	2.28 (0.12)
	$\beta = 8/2.75$	77.68 (0.26)	0.92 (0.01)	52.28 (0.20)	2.19 (0.10)
	$\beta = 10/3$	75.50 (0.17)	0.90 (0.02)	51.80 (0.25)	2.09 (0.06)
	$E = 2$	75.40 (0.15)	0.85 (0.01)	53.74 (0.43)	2.70 (0.41)
	$E = 3$	79.72 (0.27)	1.03 (0.02)	53.89 (0.36)	3.14 (0.15)
	$E = 4$	79.98 (0.17)	1.15 (0.02)	54.49 (0.20)	3.73 (0.33)
	$E = 5$	79.21 (0.25)	1.27 (0.09)	52.35 (0.28)	4.08 (0.20)
FedAvg	$E = 2$	76.60 (0.13)	1.22 (0.01)	52.51 (0.43)	3.50 (0.41)
	$E = 3$	77.92 (0.19)	1.52 (0.02)	54.16 (0.68)	4.95 (0.79)
	$E = 4$	77.65 (0.12)	1.80 (0.01)	54.00 (0.29)	6.39 (0.51)
	$E = 5$	77.7 (0.21)	2.10 (0.02)	52.85 (0.37)	7.91 (0.44)

TABLE III
AVERAGE (STANDARD DEVIATION) TEST ACCURACY AND CONVERGENCE FOR DIFFERENT METHODS

	FEMNIST				Shakespeare			
	Stop @$R_{max} = 80$		Stop @$Acc = 0.75$		Stop @$R_{max} = 40$		Stop @$Acc = 0.51$	
Methods	Best accuracy	Round length	Round needed	Total time	Best accuracy	Round length	Round needed	Total time
FedAvg	77.92 (0.19)	1.52 (0.02)	46.13 (2.98)	70.56 (5.06)	54.16 (0.68)	4.95 (0.79)	22.63 (2.63)	109.22 (16.40)
Freezing	79.87 (0.18)	1.07 (0.02)	**42.18 (1.15)**	44.81 (1.89)	**54.18 (0.43)**	3.47 (0.47)	20.08 (2.45)	68.80 (6.23)
Freezing w/ CS	79.49 (0.33)	1.01 (0.01)	42.44 (1.52)	**42.96 (1.35)**	53.91 (0.80)	2.66 (0.375)	**19.53 (2.65)**	**51.09 (4.96)**
Freezing w/ CS,WR	**80.02 (0.39)**	1.03 (0.02)	43.52 (2.52)	44.78 (3.12)	54.15 (0.51)	2.94 (0.41)	19.71(2.36)	54.48(2.80)
RBCS-F	77.67 (0.33)	**0.892 (0.02)**	52.75 (2.75)	47.01 (1.82)	51.99 (0.79)	**1.94 (0.09)**	35.68 (3.77)	69.34 (8.98)
Fjord	75.79 (0.16)	0.95 (0.03)	68.74 (4.83)	64.83 (6.22)	51.56 (0.43)	3.70 (0.18)	34.41 (3.78)	127.75 (13.57)

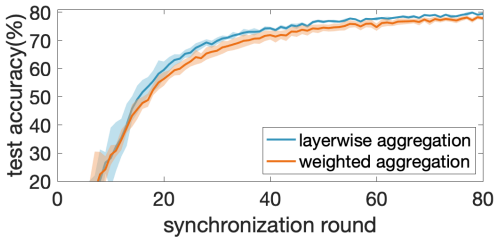
TABLE IV
AUGMENTING FJORD WITH CLIENT SELECTION

	FEMNIST		Shakespeare	
Methods	Best accuracy	Round length	Best accuracy	Round length
FJORD	75.86 (0.14)	0.95 (0.03)	51.56 (0.43)	3.71 (0.19)
FJORD W/ CS	76.17 (0.36)	0.93 (0.02)	52.02 (0.61)	3.34 (0.04)

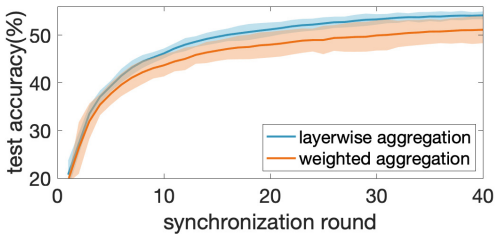
Section. IV, β is a key design in our approach for addressing the challenges of statistical heterogeneity (i.e., measured by "best accuracy") and the system heterogeneity (i.e., measured by "round length"). When $\beta = 0$, FreezeFL doesn't freeze any layer and acts the same as FedAvg. As β increases, FreezeFL tends to freeze more layers of the straggler models to mitigate system heterogeneity; hence the round length and training efficiency can be improved; but at the same time, the frozen layers can cause accuracy degradation. Therefore, as shown by the results in Table. II, both the training accuracy and round length strictly decrease as β increases. Take the result of FEMNIST as an example, its accuracy decreases from 79.4 to 75.5, and round length decreases from 1.04 to 0.90 as β becomes larger. This strong correlation property gives users the explicit control of optimizing their training for accuracy or efficiency, and any setting of β results in an optimal solution in the Pareto set from our multi-objective optimization problem.

On the other hand, the impact of E lies in the improvement of training efficiency, measured by round length. As observed in the results of Shakespeare, compared with FedAvg, the improvement of round length increases from 23% (=(3.5-2.7)/3.5) to 48% (=(7.91-4.08)/7.91) as E increases from 2 to 5. On the other hand, the impact of local epochs on training accuracy is less certain. But we can find that FreezeFL often has higher model accuracy results than FedAvg under the same setting of E in Table. II. Hence, overall FreezeFL provides an efficient solution to address the tradeoff problem between statistic heterogeneity and system heterogeneity and achieves comparable model accuracy to FedAvg with much higher training efficiency (i.e., fewer round length).

Solution Comparison As shown in Table. III, we evaluate the solutions in two scenarios: 1) stop the training at the preset round and compare the best accuracy and average round length. 2) stop the training when a preset test accuracy is



(a) On FEMNIST



(b) On Shakespeare

Fig. 5. Test accuracy under different aggregation methods

achieved and compare the total time and number of rounds needed for convergence. We can see that, compared with FedAvg, layer freezing shortens the round length by 30% and 33%, respectively, while not hurting the accuracy and model convergence. After applying client selection, the round length can be further reduced by 4% and 16%, respectively, but accuracy may be reduced due to additional statistical heterogeneity introduced by biased client selection. Tackling such accuracy reduction is the main target of warm-restart. We can observe that warm-restart increases accuracy at the cost of slightly increasing round length but still has a shorter round length than only applying layer freezing. Compared with the other two baselines, we got a big win in accuracy and convergence while being competitive in round length. This is because we consider the interaction between two heterogeneity and tackle the trade-off properly when deciding the number of frozen layers and client selection.

Furthermore, the variance in accuracy and convergence could be alternatively explained by Fig. 4. In this experiment, we divide the clients into 40 groups based on their capabilities. The clients in group 1 have the least capability, namely, the stragglers. In contrast, the clients in group 40 have the most powerful capability. Fig. 4 records the number of participation of the groups. We noticed that warm-restart does improve fairness. It less chooses the clients with powerful capability and guarantees the number of participation of the stragglers. It explains why warm-restart improves the accuracy but has a slightly larger round length. Compared with RBCS-F, we better guarantee fairness in the extreme case where the ability gap is up to six times. As described in Section. IV-B, the clients whose model exchange time is below the average do not need to perform layer freezing, and the client's utility is proportional to the number of unfrozen layers instead of its capability. Therefore, we do not excessively select the most powerful clients. Moreover, we apply the warm-restart in client selection to help stragglers more participate in training. To sum up, freezing-based reputation precisely captures the presence of stragglers and reduces their involvement, and warm-restart is used as a tuning knob, which can be adjusted according to different requirements for fairness.

Model aggregation analysis Next, we compare the layerwise aggregation with the normal weighted aggregation used in FedAvg. As shown in Fig. 5, layerwise aggregation stably achieves higher accuracy because it considers the heterogeneity in the degree of freezing and kicks out the weights of frozen layers during aggregation to prevent it slow down the convergence. With layerwise aggregation, layer freezing could bring more significant improvement to FL.

Algorithm adaptation Finally, our proposed layer freezing and client selection methods can be independently combined with other approaches to obtain better results. We demonstrate it by combing our client selection method with Fjord. To achieve it, we use the dropout rate instead of the number of unfrozen layers in Eq. 5 to estimate clients' capability. As shown in Table. IV, Fjord with client selection achieves better accuracy and shorter round length on both datasets. Furthermore, FreezeFL remains client-oblivious when combined with other methods.

VI. Conclusion

We proposed FreezeFL, which is the first work introducing layer freezing to solve system heterogeneity to the best of our knowledge. We first justify the challenge of introducing layer freezing to FL and then address the challenge by quantifying the two heterogeneity and exploring a one-dimension tuning space. We further design a client selection algorithm that cleverly uses the number of unfrozen layers to identify clients' capability to achieve client-oblivious. Furthermore, we apply warm-restart to provide a fairness guarantee and improve model performance. Our empirical evaluation across two federated datasets has demonstrated that FreezeFL can significantly improve training efficiency while not hurting model performance under the heterogeneous FL. Several remaining problems are worth to further study in the future, including providing theoretical analysis on the convergence of our technique, improving client selection and layer freezing decisions based on more experimental analyses, comparing our results with other types of system heterogeneous solutions, such as network pruning, and evaluating the impact from other FL challenges, such as the data privacy issue and availability problem in the cross-device environment.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[3] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2021.

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers. *arXiv preprint arXiv:1706.04983*, 2017.

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[6] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[7] 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 29th International Symposium on High-Performance Parallel and Distributed Computing*, pages 125–136, 2020.

[8] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. Communication-efficient federated learning with adaptive parameter freezing. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 1–11. IEEE, 2021.

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[10] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[11] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[15] Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1552–1564, 2020.

[16] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[17] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[18] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[21] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[22] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[23] Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*, 2021.

[24] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[25] Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. *arXiv preprint arXiv:2112.11256*, 2021.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[27] Muhammad Tahir Munir, Muhammad Mustansar Saeed, Mahad Ali, Zafar Ayyub Qazi, and Ihsan Ayyub Qazi. Fedprune: Towards inclusive federated learning. *arXiv preprint arXiv:2110.14205*, 2021.

[28] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications*, 39(1):201–218, 2020.

[29] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[30] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *arXiv preprint arXiv:2012.14453*, 2020.

[31] Jae Hun Ro, Theresa Breiner, Lara McConnaughey, Mingqing Chen, Ananda Theertha Suresh, Shankar Kumar, and Rajiv Mathews. Scaling language model size in cross-device federated learning. *arXiv preprint arXiv:2204.09715*, 2022.

[32] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[33] Shaohuai Shi, Xiaowen Chu, Ka Chun Cheung, and Simon See. Understanding top-k sparsification in distributed deep learning. *arXiv preprint arXiv:1911.08772*, 2019.

[34] Hakim Sidahmed, Zheng Xu, Ankush Garg, Yuan Cao, and Mingqing Chen. Efficient and private federated learning with partially trainable networks. *arXiv preprint arXiv:2110.03450*, 2021.

[35] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Agüera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas N. Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horváth, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konečný, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtárik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake E. Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. A field guide to federated optimization. *CoRR*, abs/2107.06917, 2021.

[36] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[37] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[38] Dingzhu Wen, Ki-Jun Jeon, and Kaibin Huang. Federated dropout—a simple approach for enabling federated learning on resource constrained devices. *IEEE Wireless Communications Letters*, 11(5):923–927, 2022.

[39] Zirui Xu, Zhao Yang, Jinjun Xiong, Janlei Yang, and Xiang Chen. Elfish: Resource-aware federated learning on heterogeneous edge devices. *Ratio*, 2(r1):r2, 2019.

[40] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[41] Tien-Ju Yang, Dhruv Guliani, Françoise Beaufays, and Giovanni Motta. Partial variable training for efficient on-device federated learning. In

ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4348–4352. IEEE, 2022.

[42] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[43] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[44] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.