HyperGenFL: Hypernetwork-Generated Model Aggregation in Federated Learning

Jerry Chen
University of Alberta
Edmonton, Alberta, Canada
jerry3@ualberta.ca

Qikai Lu
University of Alberta
Edmonton, Alberta, Canada
qikai@ualberta.ca

Ruiqing Tian
University of Alberta
Edmonton, Alberta, Canada
ruiqing2@ualberta.ca

Di Niu
University of Alberta
Edmonton, Alberta, Canada
dniu@ualberta.ca

Baochun Li
University of Toronto
Toronto, Ontario, Canada
bli@ece.toronto.edu

Abstract

Federated learning is a decentralized framework that enables client participation in collaborative learning without centralized data collection. However, the framework is susceptible to suboptimal model convergence induced by heterogeneity among the client datasets. These discrepancies, including label imbalance, dissimilarity in data distributions, and uneven data volumes between clients, may cause disagreements among local client updates, affecting the ability of the global model to converge effectively during aggregation. We suggest that one potential solution to this problem lies in weighting the model aggregation by client importance and client-to-client relationships. Based on this idea, we propose HyperGenFL (*HG-FL*), a hypernetwork that generates aggregation weights from learnable client embeddings without requiring any training or benchmarking data. *HG-FL* utilizes the attention mechanism to capture inter-client relationships based on learnable client-specific embeddings in order to generate model aggregation weights dynamically during federated learning. By guiding the aggregation process with these learnable relationships between local models, *HG-FL* reduces update conflicts and improves global model performance. We assess *HG-FL* under various data-heterogeneous environments based on different benchmark datasets including Fashion-MNIST, CIFAR10, CIFAR100 and Tiny-ImageNet. Experimental results demonstrate that *HG-FL* can achieve superior performance over a range of existing baseline methods under challenging cases with various heterogeneous environments, large models and a large number of clients.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Federated Learning, Data Heterogeneity, Hypernetwork

ACM Reference Format:
Jerry Chen, Qikai Lu, Ruiqing Tian, Di Niu, and Baochun Li. 2025. HyperGenFL: Hypernetwork-Generated Model Aggregation in Federated Learning.

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761368

In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761368

1 Introduction

Federated learning (FL) is a collaborative training framework that facilitates machine learning in a decentralized manner by offloading training tasks to client devices while coordinating the global model aggregation via a central server. This decentralized approach allows model training to use data outside of its local device, as client data remain local and are not exchanged. Moreover, the training FL can potentially scale to large amount of data, by leveraging the respective computational power of each client. Due to these benefits, FL has been adopted in numerous decentralized fields, including but not limited to healthcare [9, 12], finance [24], IoT [25], etc.

Conventional FL methods, e.g., FedAvg [21], rely on the assumption that client data are drawn from similar distributions. However, real-world applications rarely adhere to this assumption as variations exist in client tasks, data sources, and data quality. This heterogeneity hampers convergence, as aggregation with the disagreeing client updates can drive the global model toward suboptimality. In general, data heterogeneity may be categorized into three types: label imbalance [27], variations in client data distributions [36], and uneven data volumes [1, 26]. Label imbalance occurs when the proportions of samples for different labels vary across client datasets. Variations in client distributions occur when clients collect data from distinct domains or tasks. Uneven data volumes refer to discrepancies in the amount of samples available to different clients for training tasks. These forms of data heterogeneity often result in non-IID (not independently and identically distributed) data. Addressing the adverse effects of data heterogeneity is essential to ensuring the quality of FL in real-world scenarios.

Recent research to mitigate the divergence caused by data heterogeneity in FL can be broadly classified into client-side and server-side optimization methods. Client-side methods [13, 14, 28] generally use regularization to prevent the deviation of locally updated client models from the global model. However, these methods generally do not fully adapt to local client data distributions in a heterogeneous environment, a problem exacerbated when facing a large number of clients [29, 34].

On the other hand, server-side methods focus on mitigating the divergence during global model aggregation. FedDF [16] introduces an additional training phase for the newly aggregated global model at the server. FedLAW [15] trains a hypernetwork at the server to predict the model aggregation weights. However, both methods requires a benchmark dataset at the server that well resembles the test set distribution, which is often impractical to obtain. FedLaw [18] (different from the aforementioned FedLAW) aims to derive aggregation weights by computing the pairwise similarities of model parameters between clients. However, their proposed similarity measure tends to diminish for larger models, thereby undermining the resultant model accuracy and effectiveness. These limitations underscore the need of more robust and practical aggregation strategies in FL.

In this paper, we propose *HG-FL*, a robust framework to conduct federated learning in highly heterogeneous environments. *HG-FL* generates aggregation weights through a server-side hypernetwork from learnable client embeddings. Not only does *HG-FL* achieve significant and consistent performance improvements over state-of-the-art FL methods when applied to large models and scenarios with extreme data heterogeneity, but it also eliminates the need for training or benchmarking data at the server. In designing *HG-FL*, we make the following main contributions:

- We employ attention mechanism [30] to dynamically learn inter-client relationships in *HG-FL*. This approach effectively captures complex, nonlinear dependencies between clients, prioritizes the most relevant connections, and adapts to fluctuations in data distributions. As a result, it circumvents the biases and noise typically introduced by hand-crafted similarity measures between client models.
- *HG-FL* also learns client embeddings to provide long-term, yet efficient, memory of the participating clients, allowing the retention of crucial client characteristics over time. This continuity helps eliminate disruptions caused by noise and randomness in client-side local training, leading to more stable and personalized learning. Essentially, embeddings act as a memory foundation, ensuring that the model adapts effectively without losing valuable insights.
- We propose a dataless hypernetwork training method at the server. Instead of computing the loss based on benchmarking data at the server, our loss function aims to reduce the difference between the global model and the aggregated local model parameters. This approach leverages the discrepancy between the central model and the distributed client models as a proxy for optimization, enabling server-side hypernetwork training without any public training or benchmarking data. By eliminating the need for benchmarking or additional training data, *HG-FL* also achieves higher robustness on non-IID data distributions between clients, making the approach highly scalable and suitable for FL under data heterogeneity.

We have performed extensive experiments on several widely used FL evaluation data sets, including Fashion-MNIST [33], CIFAR10, CIFAR100 [11], and Tiny-ImageNet [22], under varying degrees of data heterogeneity and varying numbers of clients. Results demonstrate that *HG-FL* consistently achieves significant performance improvements over state-of-the-art baseline FL methods, particularly in challenging and heterogeneous scenarios, often outperforming them by substantial margins (e.g., beating the best existing method by 0.88% on CIFAR10 and by 3.18% on CIFAR100 in the heterogeneous test case). Additionally, *HG-FL* also delivers competitive results on standard benchmarks without heterogeneity, demonstrating its versatility and generality. These findings reveal *HG-FL*'s superior capability to handle FL under diverse and complex scenarios, including large-scale FL tasks, large models and highly non-IID data distributions.

2 Related Work

The conventional FL framework was first introduce by Google as FedAvg. FedAvg deployed a simple framework of local training at clients and aggregation at the server. This approach provides a solution to the issues mentioned earlier, but it has its problems. The non-IID issue introduced by data heterogeneity destroys the performance of FedAvg. Subsequent literature addressing this issue can be broadly categorized into two major directions: a more sophisticated learning method for clients and a more optimized server aggregation method. On the local training side, FedProx [14] introduces a proximal term. This term adds an additional loss that calculates how much the local model parameters deviate from the given global model, forcing small changes in updating values. SCAFFOLD [10] introduces control variates to correct client drift during each local training by re-computing the update gradients. FedSam [28] ensures that the aggregation results in a model close to a local minimum. It achieves this by adapting sharpness-awareness minimization (SAM) with stochastic averaging. FedDyn [2] also introduced a regularization term α in local updates to constrain the model gradients. MOON [13] adds a contrastive loss at local training converted from the cosine similarity between the the local model and the global model predictions. These methods only focus on the client-to-server relationship and ignore client-to-client relationships.

Alternatively, modifications can be made at the server-side aggregation process. As mentioned earlier, the conventional method generates weights by the client data volumes. Some studies try to calculate the weights with more sophisticated methods. FedNova [31] adds an additional layer of normalization to the local gradients and performs averaging on the gradients instead of the model parameters. FedVARP [31] attempts to reduce variance from partial client participation. FedVARP approximate the updates from inactive clients by using the latest updates from those clients. FedAVGAU [32] tackles the problem from a different aspect by accumulating the updates to the server and adding a fraction of the accumulated update to the global model in regular communication intervals. FedDF [16] extends on FedAvg by taking the extra step of using knowledge distillation from unlabeled calibration data to further update the global model. FedBE [3] takes advantage of Bayesian posterior distribution based on the assumption that the parameter of the model is a probability distribution based on data. FedLAW [15] generates the aggregation weights from a hypernetwork. This hypernetwork treats the aggregation weights as model weights and trains a number of epochs at the server on a partition of the benchmark set. FedLaw [18] computing the difference and cosine similarity between their respective model parameters, and uses

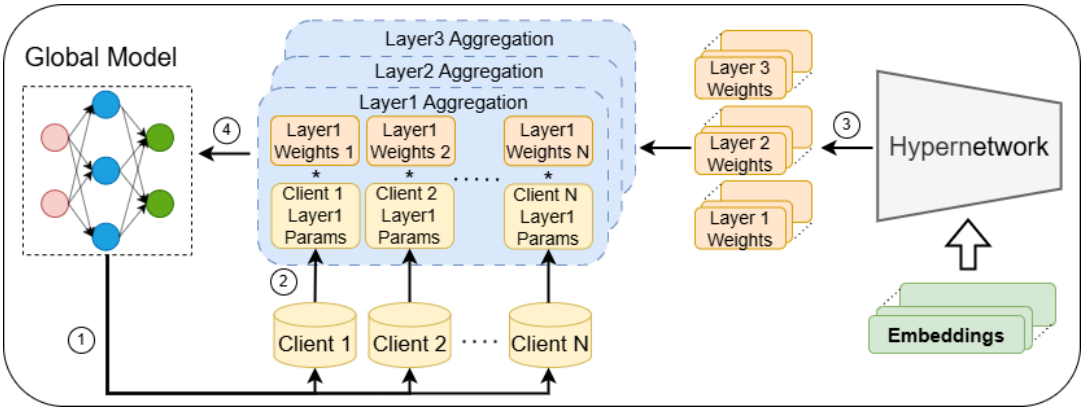


Figure 1: *HG-FL* structure overview, the hypernetwork replaces the original weighted averaging component. It has the following steps similar to FedAvg: 1 The server sends a copy of the global model to the clients. 2 The local models send the updated local models back to the server. 3 The hypernetwork generates the layer-wise aggregation weights for the collected models. 4. Replaces the old global model with the new model.

these distances to compute "fair" aggregation weights. This method measures the client pairwise relationships, but is insufficient for capturing comprehensive information for aggregation.

There are other methods that focus on different aspects of the FL framework. IFCA [7] is the first method that proposes a clustering scheme for server side aggregation. The server stores multiple global models, and each client picks one that best fits to its data distribution for update and upload to the server. FedSEM [17] also has a clustering scheme. But instead of letting the clients pick, the server assigns the clients to their respective clusters. Without data at the server, FedSEM uses the model parameter differences to infer distance between two models. K-FED [4] introduces One-shot clustering. It clusters the clients at the very beginning of the communication rounds and keeps the clustering to reduce communication costs. FedOV [5] introduces an unknown label to the models. In order to train the model for the unknown label, they manually generate outliers by modifying the data samples (such as cropping, random rotations, and patch swapping). This allows the clients to avoid guessing an input that is not in its data distribution during prediction. FLDR [23] takes the approach of selecting the clients' data. It lets each client train a correlation data selector that predicts correlations of its data, then discards the irrelevant data. FedKernel [35] proposes a CV-specific solution to solve the FL challenges. It alters the backbone model by adding three parallel convolutional kernels. These three kernels are trained simultaneously, and during testing are summed on overlapping cells to form one single kernel. FedDNA [6] suggests that the statistical parameters (such as running mean and running variance) should not be aggregated in the same way as other model parameters. Instead, they proposed a variational encoder-decoder to take statistical parameters and generate. Another approach is client selection where the server would evaluate the clients and only select the clients that would yield the most optimal global model for aggregation. F-RCCE [37] adapts a reinforcement learning agent to actively select the clients and rewards it with the performance of the aggregated model. A couple of notable mentions of the weights generation by hypernetwork methods are two personalized FL methods called pFedLA [20] and HyperFLoRA [19]. pFedLA is a generative model that decodes a single embedding into aggregation weights for all active clients. The hypernetwork of HyperFLoRA generates model parameters of personalized adapters instead of aggregation weights. These adapters would attach themselves to the layers of the global model to form their respective client model. However, none of these methods takes client-to-client relationships into consideration.

3 Method

In this section, we first introduce the background, then present the major components of our network, and lastly, formulate the data-less learning method.

Let C be the set of active clients in the collaboration network, where $|C|$ is the total number of clients. For an arbitrary client $c \in C$, D_c refers to its local private data. FL attempts to train a global model, θ_g, that best accommodates all private datasets, which can be expressed by the following objective:

$$\begin{aligned} &\arg\min_{\theta_g} \sum_{c\in C} \frac{|D_c|}{n} \sum_{(x,y)\in D_c} \frac{1}{|D_c|} L(\theta_g; (x,y)) \\ &= \frac{1}{n} \arg\min_{\theta_g} \sum_{c\in C} \sum_{(x,y)\in D_c} L(\theta_g; (x,y)) \\ &= \frac{1}{n} \arg\min_{\theta_g} \sum_{c\in C} L_c(\theta_g), \end{aligned} \tag{1}$$

where the L is the loss function, and $n = \sum_{c=1}^{C} |D_c|$. Generally, training of θ_g is conducted at the server side through the weighted aggregation of a set of collected client-trained models θ_c, with the client models identical in backbone architecture. As defined by FedAvg, aggregation is performed via the following equation:

$$\theta_g = \sum_{c\in C} \frac{|D_c|}{n} \theta_c, \tag{2}$$

Effectively, FedAvg gives clients with more data samples more influence during aggregation. However, this assumption is invalid

in non-IID scenarios. Additionally, different θ_c may enact divergent weight updates, leading to suboptimal θ_c after aggregation.

We introduce HG-FL to resolve suboptimal aggregation pertaining to non-IID scenarios. Principally, we employ a hypernetwork to learn and generate the optimal aggregation weights by leveraging inter-client relationships. Additionally, we facilitate fine-grained aggregation by allowing each model layer to independently acquire different aggregation weights. Mathematically, we define the layer l (up to l_{max}) of the model θ_g and θ_c respectively as θ_g^l and θ_c^l, and α_c^l as corresponding aggregation weight. Thus, HG-FL aggregation can be defined as follows:

$$\theta_g^l = \sum_{c \in C} \alpha_c^l \theta_c^l, \text{ where } \sum_{c \in C} \alpha_c^l = 1. \tag{3}$$

For organizational purposes, for client c, we define $\theta_c = \{\theta_c^l\}_{l=1}^{l_{max}}$ and correspondingly $\alpha_c = \{\alpha_c^l\}_{l=1}^{l_{max}}$. We then group the client-specific variables as $\Theta = \{\theta_c\}_{c \in C}$ and $A = \{\alpha_c\}_{c \in C}$.

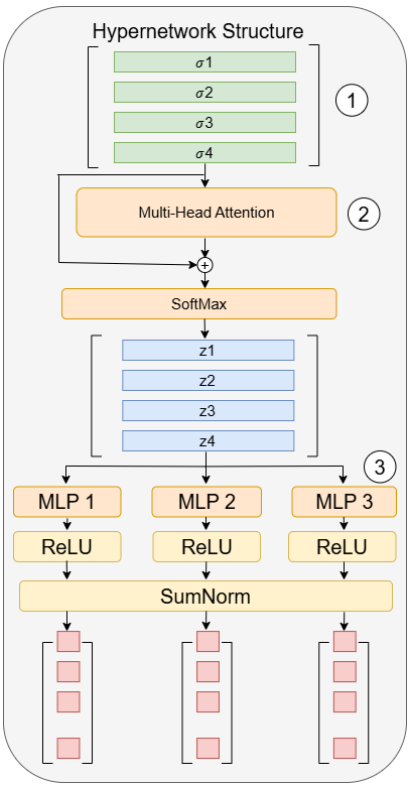


Figure 2: An example of *HG-FL* hypernetwork with 4 clients and 3-layer backbone model. 1 The embeddings (green blocks) are fed to the attention block. 2 The attention block produces relationship-aware latent vectors (blue blocks). 3 Multiple parallel layer-specific MLPs, ReLU activations and the row-wise SumNorm block generates the layer-wise aggregation weights (red blocks).

3.1 Hypernetwork Weights Generation

The proposed hypernetwork is designed to generate aggregation weights for constructing the global model. To achieve this, each client is assigned a learnable embedding vector for encapsulating latent information of its respective data distribution. As shown in Figure 2, the hypernetwork thus consumes the client representations, captures the inter-client relationships using a multi-head attention mechanism (MHA), and finally infers the aggregation weights.

To generate a collection of α_c^l for a set of participating clients during aggregation, the hypernetwork receives a set of client embeddings corresponding to active clients and successively feeds them through its internal MHA and softmax layer. We compose a matrix $\sigma \in |C| \times \mathbb{R}^d$ from the set of active client embeddings $\{\sigma_c\}_{c \in C}$. The MHA contextualizes the embedding with relationships of the underlying distributions between the set of clients. This relationship enables the clients to jointly compute their respective client-aware latent vectors z_c for subsequent client parameter generation. Acquisition of the encoding can therefore be formulated as:

$$\begin{aligned} \{z_c\}_{c \in C} &= Query(\phi_{MHA}; \sigma) \\ &= \text{softmax}(\text{MHA}(\phi_{MHA}; \sigma) + \sigma). \end{aligned} \tag{4}$$

The second part of the hypernetwork focuses on generating the aggregation weights using the output from the attention model. To decode each z_c into a set of layer-wise scores α_c (recall that $\alpha_c = \{\alpha_c^l\}_{l=1}^{l_{max}}$), we use a concurrent 1-layer MLP with ReLU, each corresponding to a specific θ_c^l. This yields the unnormalized scores, as shown below:

$$\begin{aligned} s_c^l &= Decode(\phi_{MLP}^l, \lambda; z_c) \\ &= ReLU(MLP(\phi_{MLP}^l; z_c)) + \lambda, \end{aligned} \tag{5}$$

Note that the λ value is a small positive constant that ensures the sum of s is a positive value. Then we normalized these scores to form the final aggregation scores:

$$\begin{aligned} \alpha_c^l &= SumNorm(\{s_c^l\}_{c \in C}, c) \\ &= \frac{s_c^l}{\sum_{c \in C} s_c^l}. \end{aligned} \tag{6}$$

Once we have the normalized α values, we apply weighted averaging to each layer in the backbone.

Overall, we summarize the hypernetwork as:

$$\begin{aligned} A &= HN(\phi_{hn}, \sigma) \\ &= \{SumNorm(Decode(\phi_{MLP}^l, \lambda; Query(\phi_{MHA}; \sigma)))\}_{l=1}^{l_{max}} \end{aligned} \tag{7}$$

and the aggregation process, henceforth denoted by $[*]$, is defined as:

$$\begin{aligned} \theta_g &= [\Theta * A] \\ &= \{\sum_{c \in C} \theta_c^l \cdot \alpha_c^l\}_{l=1}^{l_{max}}. \end{aligned} \tag{8}$$

This aggregates the models using weights from the set of Eq. (7) to produce the parameters in the θ_g for the next round.

3.2 Hypernetwork Learning

In this section we explain how to train the hypernetwork. Generally, training a model requires data and a loss function. However, in the FL setup, we do not have access to private data, therefore we cannot update our hypernetwork conventionally. To tackle this problem, we have to approach this problem differently.

Given the objective function

$$\frac{1}{n}\arg\min_{\theta_g}\sum_{c\in C}\sum_{(x,y)\in D_c} L(\theta_g;(x,y)), \tag{9}$$

where σ is the active client embeddings, ϕ is the combination of the hypernetwork parameters, C is collection of the active clients, l_c is the loss function of client c, and θ_g is the parameters of the backbone model aggregated at the server. In our case, we take the difference between global model parameters and collected local model parameters, then manually compute the gradients.

$$\begin{aligned}\Delta\sigma &= \frac{1}{|C|}\sum_{c\in C}\frac{\partial\theta_g}{\partial\sigma}\cdot\Delta\theta_c \\ &= \frac{1}{|C|}\sum_{c\in C}[\Theta * \frac{\partial A}{\partial\sigma}]\cdot\Delta\theta_c\end{aligned} \tag{10}$$

and

$$\begin{aligned}\Delta\phi &= \frac{1}{|C|}\sum_{c\in C}\frac{\partial\theta_g}{\partial\phi}\cdot\Delta\theta_c \\ &= \frac{1}{|C|}\sum_{c\in C}[\Theta * \frac{\partial A}{\partial\phi}]\cdot\Delta\theta_c.\end{aligned} \tag{11}$$

To update σ and ϕ, we can find the gradients using (10) and (11). Recall that Θ represents the set of local updated model $\{\theta_1, \theta_2..., \theta_{|C|}\}$ and $\Delta\theta_c$ is the difference in parameters between local model c and the global model θ_g. Once we have the gradients $\Delta\sigma$ and $\Delta\phi$, we multiply them with our learning rate and update both embeddings and the hypernetwork.

In order to obtain equations (10) and (11), we start by applying the chain rule to (9) and we get the gradient of σ as:

$$\frac{\partial L_c}{\partial\sigma} = \frac{\partial\theta_g}{\partial\sigma}\cdot\frac{\partial L_c}{\partial\theta_g}. \tag{12}$$

From (8) we know that only α values are affected by σ while θ remains constant during aggregation, therefore by inserting the first order derivative of (8) into (12) we get the following:

$$\frac{\partial L_c}{\partial\sigma} = [\Theta * \frac{\partial A}{\partial\sigma}]\cdot\frac{\partial L_c}{\partial\theta_g} \tag{13}$$

and similarly the gradient of ϕ as:

$$\begin{aligned}\frac{\partial L_c}{\partial\phi} &= \frac{\partial\theta_g}{\partial\phi}\cdot\frac{\partial L_c}{\partial\theta_g} \\ &= [\Theta * \frac{\partial A}{\partial\phi}]\cdot\frac{\partial l_c}{\partial\theta_g}.\end{aligned} \tag{14}$$

We can find $\frac{\partial L_c}{\partial\theta_g}$ as $\Delta\theta_c$ from client c's local training. We set each local client as the update target for the global model:

$$\Delta\theta_c = \theta_c - \theta_g, c \in C. \tag{15}$$

The rationale is that with the following update rule where η is the learning rate:

$$\theta' = \theta - \eta\frac{\partial L}{\partial\theta}. \tag{16}$$

After some transformations we get:

$$\begin{aligned}\frac{\partial L}{\partial\theta} &= \frac{1}{\eta}(\theta - \theta') \\ &= \frac{1}{\eta}\Delta\theta.\end{aligned} \tag{17}$$

We can safely ignore the learning rate η as we only need to find the update direction, and we will apply our own learning rate. Substitute (14) into (12) and (13), we can compute the updates of σ and ϕ as with (10) and (11). During the communication round after collecting the local updated models, we are able to update the corresponding embeddings σ and hypernetwork parameters ϕ (which is composed of ϕ_{trans} and ϕ_{MLP}).

Algorithm 1 HG-FL

1: **Server executes:**
2: Initialize θ_0 and broadcast to all clients
3: $\sigma \leftarrow 1$
4: **for** each round $t = 0, \ldots, T-1$ **do**
5: $C \leftarrow$ the active clients are randomly selected
6: $\sigma \leftarrow$ the embeddings of active clients C
7: $\Theta \leftarrow$ the local models active of clients C
8: $A \leftarrow HN(\phi, \sigma)$ as (7)
9: $\theta_g \leftarrow [\Theta * A]$ as (8)
10: Server sends θ_g to clients C
11: **for** client parameters $\theta_c \in \Theta$ **in parallel do**
12: $\theta_c \leftarrow$ **Local Training**(θ_c, D_c)
13: **end for**
14: $\Theta \leftarrow \{\theta_c\}_{c\in C}$
15: $\Delta\Theta \leftarrow \{\theta_c - \theta_g\}_{c\in C}$ as (15)
16: $\phi = \phi - \eta\cdot updateParams(\Theta, \Delta\Theta, (\phi, \sigma))$ as (10)
17: $\sigma = \sigma - \eta\cdot updateEmbeddings(\Theta, \Delta\Theta, (\phi, \sigma))$ as (11)
18: **end for**

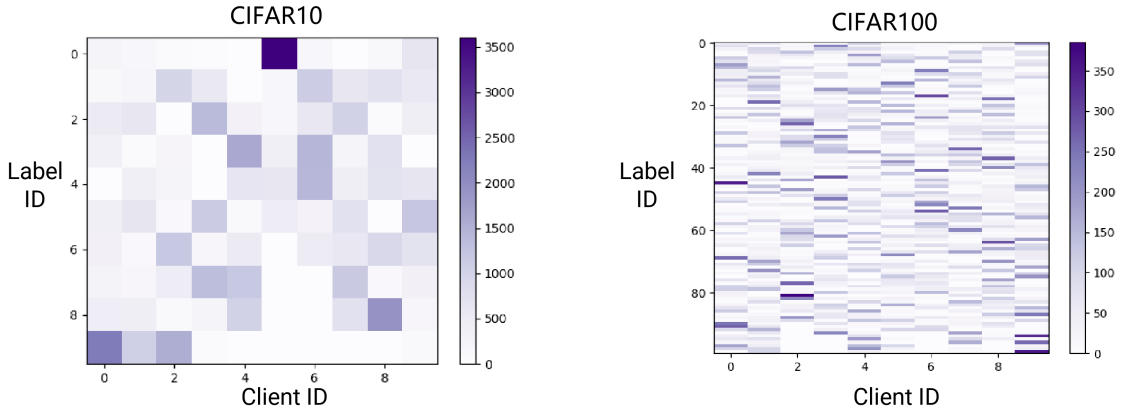
The pseudocode 1 shows the entire FL procedure that includes both server aggregation (lines 5 to 9) and hypernetwork update (lines 14 to 17).

4 Experiments

In this section, we demonstrate our experiment designs and the results obtained.

4.1 Experiment Setup

We conducted the experiments on well-known datasets Fashion-MNIST[33], CIFAR10, CIFAR100 [11] and Tiny-ImageNet [22]. To simulate the data heterogeneity problem, we split each dataset into several partitions by Dirichlet distribution. This method randomly partitions the samples across the clients with a Gamma distribution controlled by hyperparameter β. Figure 3 shows the label distribution for CIFAR10 and CIFAR100 datasets when β set to 0.5. For CIFAR10, we can see that the client 5 partition is predominantly



(a) Label distributions for CIFAR10 standard scenario

(b) Label distributions for CIFAR100 extreme scenario

Figure 3: A single stance set of label distributions for all clients on CIFAR10 and CIFAR100 datasets. Both images reveal patterns of high variations in color patches, signifying imbalance and non-IID distribution of samples across clients and labels.

Table 1: Experiments Configurations

Experiment Name	# total clients	# active clients	β
Standard	10	5	0.5
Advanced	100	20	0.25
Extreme	100	20	0.1

populated with samples belonging to label 0, but is scarce with samples belonging to labels 6,7,8,9. This shows an extremely skewed distribution generated from the Dirichlet distribution. Summarily, as β decreases, the client data distribution becomes increasingly skewed.

For the experiments, we evaluate our methods on three scenarios of increasing difficulty. The first one is a standard test case with 10 clients in total. During each communication round, we randomly sample 5 clients as active clients. These clients perform local training and pass the updated models back to the server. The β value of the Dirichlet distribution is set to 0.5. The test case runs a total of 100 communication rounds. For Fashion-MNIST, CIFAR10, and CIFAR100, each active client has 10 epochs of local training, while for Tiny-ImageNet, the clients only run 1 local epoch. We call this the standard test case as it has very few clients with a high active ratio, as well as a relatively high β value. In this setting, most baselines can have reasonable performances. The second test case is an advanced test case that has 100 clients, but only 20 of them are active per round. On top of that, we set β to 0.25, a much more skewed setup. All of the other settings remain the same as the standard test case. The last test case is an extreme test case that has a β of 0.1 while keeping everything else the same as in the second test case.

The backbone model differs for different datasets. For CIFAR10, we deploy a simple CNN model, with two convolution layers followed by two MLP layers in the encoder, for a small-scale preliminary assessment. For Fashion-MNIST, CIFAR100, and Tiny-ImageNet, we use Resnet50 [8] as the backbone model.

We implemented the *HG-FL* with Pytorch. All of the other baselines are modified accordingly. We set the batch size to 32. We used AdamW as an optimizer with a learning rate of 0.01 and weight decay of 0.00001 for the *HG-FL*. Since the task is image classification, the loss function at the client end is set to cross-entropy loss. We set the learning rate for the hypernetwork to 0.01. We configured the dimension sizes of both the embeddings and the hypernetwork MHA to 128. Based on studies in Section 5.1, we initialize the embeddings as all-ones vectors. We set the special hyperparameter μ to 0.01 for FedProx and MOON, and we set temperature to 0.01 for FedProx and 0.5 for MOON as in their papers. After every round of communication, the server sends the newly aggregated model to all clients and replaces their old models. All experiments are conducted on an Nvidia RTX 3090ti GPU, with a CPU AMD Ryzen 9 5900X 12-Core Processor.

Baselines We examine our performance with other state-of-the-art methods. They are FedProx, MOON, FedSam, FedLaw, FedDF, and FedBE. In addition, we have vanilla FedAvg as the baseline. The final performances are the highest accuracies recorded during the server evaluation on the test set at the end of each communication round.

The followings are the state-of-the-art methods we compare our results with:

- **FedAvg** [21] is the conventional aggregation method that performs a weighted average by clients' data volumes.
- Client-side optimization methods
  - **FedProx** [14] adds a regularization loss at the client that restrains local deviation by finding model parameter differences.

Table 2: Experiment results on Fashion-MNIST as private datasets with three test experiment scenarios.

Algorithm	Standard	Advanced	Extreme
FedAvg	92.47	86.52	77.51
FedProx	91.02	85.26	78.74
MOON	92.84	85.16	84.14
FedSam	92.76	88.16	78.89
FedLAW	91.96	**90.18**	78.30
FedDF	90.35	89.39	80.07
FedBE	90.75	89.33	76.96
HG-FL	**93.09**	88.83	**84.65**

Table 3: Experiment results on CIFAR10 as private datasets with three test experiment scenarios.

Algorithm	Standard	Advanced	Extreme
FedAvg	58.8	49.62	38.78
FedProx	67.96	43.62	35.95
MOON	**71.7**	53.64	43.43
FedSam	69.31	49.43	48.56
FedLAW	70.59	29.71	46.07
FedDF	66.72	52.44	36.36
FedBE	66.72	52.14	39.14
HG-FL	70.04	**58.88**	**49.66**

Table 4: Experiment results on CIFAR100 as private datasets with three test experiment scenarios.

Algorithm	Standard	Advanced	Extreme
FedAvg	62.83	36.16	28.86
FedProx	59.21	17.44	22.11
MOON	**67.38**	41.88	35.38
FedSam	64.33	41.82	30.57
FedLAW	69.27	28.8	28.05
FedDF	60.99	27.26	24.57
FedBE	60.81	27.91	27.22
HG-FL	63.30	**41.97**	**39.03**

Table 5: Experiment results on Tiny-ImageNet as private datasets with two test experiment scenarios.

Algorithm	Standard	Advanced
FedAvg	27.93	4.06
FedProx	17.44	1.59
MOON	33.15	3.40
FedSam	33.4	5.45
FedLAW	28.28	1.03
FedDF	27.61	3.02
FedBE	26.94	5.56
HG-FL	**35.12**	**10.54**

  - **MOON** [13] adds a regularization loss at the client that restrains local deviation by computing the prediction difference between local and global models.
  - **FedSam** [28] seeks a region with low loss values via adding a small perturbation to the model update.
- Server-side optimization methods
  - **FedLAW** [15] adapts a hypernetwork to generate the global model, and trains the hypernetwork with the data on the server side.
  - **FedDF** [16] generates the global model with conventional FedAvg, then trains on the global model directly on benchmark data at the server side.
  - **FedBE** [3] generates the global model parameters by sampling from a Bayesian posterior distribution.

4.2 Results

Tables 2 3, 4 and 5 present the experiment results on datasets Fashion-MNIST, CIFAR10, CIFAR100 and Tiny-ImageNet respectively. Each table includes three scenarios—standard, advanced, and extreme—ordered from the easiest to the most challenging. In table 2, *HG-FL* beats other methods in both the standard and extreme scenarios and achieves comparable results in the advanced difficulty case. However, Fashion-MNIST is a relatively simple dataset where high baseline accuracies may limit the observable improvements of federated learning methods. Tables 3, 4 both demonstrate that *HG-FL* excels under high data heterogeneous scenarios, particularly in the extreme conditions. Note that *HG-FL* beats the second-best method by 3.65% in the extreme scenario, despite that the overall performances are lower than 40%. Lastly, table 5 highlights the performance on the more challenging Tiny-ImageNet dataset, which involves colored images and a larger label space. Even in the standard case, most state-of-the-art methods only obtain accuracies around 20-30%, yet *HG-FL* still outperforms them. Due to uniformly poor performance across methods in the advanced setting, the extreme scenario is omitted. Nevertheless, *HG-FL* maintains the top position even in this more difficult case.

Another aspect is that during communication, the clients and the server are only exchanging model parameters, just like conventional FedAvg. This ensures our methods do not pose any extra communication cost.

4.3 Convergence Analysis

Recall that our global objective is to minimize the average loss across all clients as in Eq. (1). A consistent decrease in loss, followed by a plateau, typically suggests successful convergence of the model, although it may also indicate potential overfitting. Figure 4 illustrates the training losses and test accuracies of four algorithms (*HG-FL*, FedLAW, FedDF, FedBE) across the communication rounds under two experiments: CIFAR100 with extreme data heterogeneity (left) and Tiny-ImageNet with standard heterogeneity (right). In the top-left plot, all methods show steadily declining losses, suggesting convergence. However, in the bottom-left plot, FedLAW exhibits unstable test accuracy despite its smooth training loss curve, implying overfitting and poor generalization. The results on Tiny-ImageNet (right column) show similar or even more pronounced issues. FedLAW again shows severe overfitting, with highly erratic

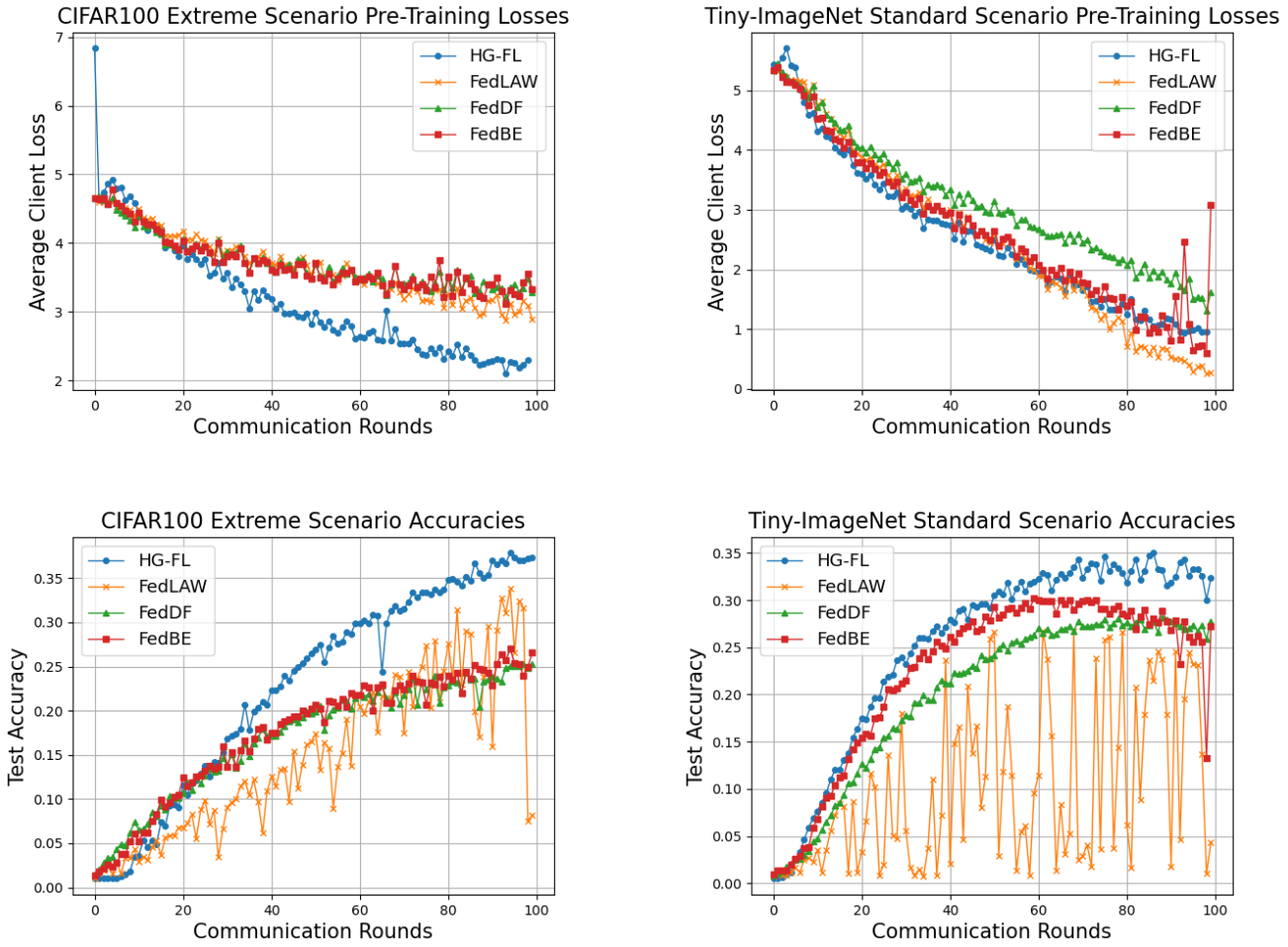


Figure 4: The average pre-training loss and accuracies recorded from the server-optimization based methods (*HG-FL*, FedLAW, FedDF and FedBE). The left two figures are the recorded losses and accuracies from experiments conducted on the dataset CIFAR100 under an extreme scenario. The right two figures are the recorded losses and accuracies from experiments conducted on the dataset Tiny-ImageNet under the standard scenario.

accuracy despite smooth loss decay. Additionally, FedBE exhibits fluctuations in its loss curve toward the later stages of training (top-right), which is mirrored by instability in its accuracy performance (bottom-right). This behavior suggests that FedBE fails to achieve stable convergence under the given experimental configuration. In contrast, *HG-FL* consistently demonstrates smooth and stable convergence across the scenarios, maintaining both decreasing loss and steadily improving test accuracy—highlighting its robustness in handling heterogeneous data distributions.

5 Ablation Study

In this section, we examine two key areas: the impact of embeddings on performance and the alternative loss functions available for updating the hypernetwork on the server.

5.1 Embeddings Initialization

In *HG-FL*, we configure the embedding dimension to 128 and initialize the embedding values as all-ones. To investigate the validity of our approach, we explore two additional methods. The first method employs one-hot encoding, where the embedding dimension is set to the number of clients ($|C|$), with a value of 1 assigned to the position corresponding to the client index. The embeddings in this case are not updated during training at the server. The second method retains the original dimension (128) but initializes the embeddings with random values, and they will be updated just like the original *HG-FL* embeddings.

Table 6 indicates that neither one-hot encoding nor random embeddings achieves results comparable to starting with ones. One-hot encoding fails to capture client distribution information and introduces sparsity, which logically explains its poorer performance. Randomly initialized embeddings lack consistency in representing

Table 6: Experiment results for *HG-FL* with different embeddings initialization methods. Evaluated on CIFAR10 (Simple-CNN), CIFAR100 (Resnet50) and Tiny-ImageNet (Resnet50) as private datasets under the extreme scenario.

Algorithm	CIFAR10	CIFAR100	Tiny-ImageNet
1-init embeddings	**49.66**	**39.03**	**6.91**
one-hot embeddings	48.38	37.17	6.58
random embeddings	46.77	38.23	6.71

Table 7: Experiment results for *HG-FL* with different MHA attention layers and embedding dimensions. Evaluated on the dataset CIFAR10 under the extreme scenario.

MHA Layers	Embedding Size		
	64	128	256
1	48.23	**49.66**	48.93
2	48.81	48.24	47.09
3	48.78	49.33	49.53

Table 8: Experiment results for *HG-FL* with different MHA attention layers and embedding dimensions. Evaluated on the dataset CIFAR100 under the extreme scenario.

MHA Layers	Embedding Size		
	64	128	256
1	**39.36**	39.03	37.42
2	38.72	38.54	37.11
3	38.38	37.90	38.07

client contributions, and the inherent randomness introduces noise, complicating the convergence process.

5.2 MHA Layer Size And Embeddings Dimension Size

The default settings in *HG-FL* have only 1 layer of MHA block, and the embeddings have a dimension of 128. We want to observe how different values affect the model performance.

Both tables 7 and 8 show that the model performs under different values with CIFAR10 and CIFAR100 datasets under extreme scenarios, respectively. MHA with 1 layer always outperforms other numbers of layers. This indicates that 1 layer is sufficient to capture the inter-client relationships. Having more layers introduces more parameters, and complicates the hypernetwork training. As for embedding size, 64 and 128 have the best performances, respectively. This shows that a small embedding dimension enforces a more compact representation that avoids overfitting.

5.3 Different Loss Functions

HG-FL utilizes a loss function similar to MSE loss. To evaluate alternative approaches, we propose comparing them with MSE loss and Log-Cosh loss.

Table 9: Experiment results for *HG-FL* with different loss functions to update the hypernetwork. Evaluated on CIFAR10 (Simple-CNN), CIFAR100 (Resnet50), and Tiny-ImageNet (Resnet50) as private datasets with the extreme scenario configuration.

Algorithm	CIFAR10	CIFAR100	Tiny-ImageNet
HG-FL Default Loss	**49.66**	**39.03**	**6.91**
HG-FL MSE loss	47.2	34.34	6.51
HG-FL Log-Cosh loss	48.58	38.33	5.82

Table 9 shows the results for different loss functions. Our update method beats the methods using MSE loss of Log-cosh loss. Log-cosh can saturate gradients for large errors (due to the log component), and MSE can produce very large gradients, causing instability. Our method explicitly computes the gradient of the hypernetwork parameters with respect to the raw backbone model parameter difference. This direct gradient calculation can provide more precise updates aligned strictly with reducing the immediate error (the parameter difference). The results suggest that our loss function is the most promising approach.

6 Conclusion

We introduced *HG-FL*, a robust federated learning framework designed for highly data heterogeneous environments. The crux of the design lies in a hypernetwork that generates layer-specific aggregation weights, effectively mitigating the adverse effects of divergent updates caused by non-IID client data. Our approach leverages learnable embeddings to maintain reliable and long-term representations of client distributions, and uses an attention mechanism to capture inter-client relationships for aggregation weights generation. Moreover, we devised an accompanying strategy for updating the hypernetwork in a data-less manner. Extensive experiments validate the effectiveness of HG-FL, demonstrating consistent improvements over baseline FL methods, with accuracy gains of up to 3.65% in scenarios with extreme heterogeneity.

7 GenAI Usage Disclosure

We claim that the only generative AI tool to prepare this manuscript was ChatGPT, and it was exclusively used for grammar and typo corrections. No content, ideas or designs were generated by AI in the development of this manuscript.

Acknowledgments

This research received support from the Natural Sciences and Engineering Research Council of Canada (NSERC) Alliance and Alberta Innovates CASBE grant under project number 232404547 and was enabled through collaboration with Wedge Networks.

References

[1] Ahmed M Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, and Marco Canini. 2023. A comprehensive empirical study of heterogeneity in federated learning. *IEEE Internet of Things Journal* 10, 16 (2023), 14071–14083.

[2] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. 2021. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263* (2021).

[3] Hong-You Chen and Wei-Lun Chao. 2020. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974* (2020).

[4] Don Kurian Dennis, Tian Li, and Virginia Smith. 2021. Heterogeneity for the win: One-shot federated clustering. In *International conference on machine learning*. PMLR, 2611–2620.

[5] Yiqun Diao, Qinbin Li, and Bingsheng He. 2023. Towards addressing label skews in one-shot federated learning. In *The Eleventh International Conference on Learning Representations*.

[6] Jian-Hui Duan, Wenzhong Li, and Sanglu Lu. 2021. FedDNA: Federated learning with decoupled normalization-layer aggregation for non-iid data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 722–737.

[7] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in neural information processing systems* 33 (2020), 19586–19597.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. 2020. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence* 2, 6 (2020), 305–311.

[10] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.

[11] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[12] Rajesh Kumar, Abdullah Aman Khan, Jay Kumar, Noorbakhsh Amiri Golilarz, Simin Zhang, Yang Ting, Chengyu Zheng, Wenyong Wang, et al. 2021. Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging. *IEEE Sensors Journal* 21, 14 (2021), 16301–16314.

[13] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10713–10722.

[14] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[15] Zexi Li, Tao Lin, Xinyi Shang, and Chao Wu. 2023. Revisiting weighted aggregation in federated learning with neural networks. In *International Conference on Machine Learning*. PMLR, 19767–19788.

[16] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems* 33 (2020), 2351–2363.

[17] Guodong Long, Ming Xie, Tao Shen, Tianyi Zhou, Xianzhi Wang, and Jing Jiang. 2023. Multi-center federated learning: clients clustering for better personalization. *World Wide Web* 26, 1 (2023), 481–500.

[18] Jianfeng Lu, Hangjian Zhang, Pan Zhou, Xiong Wang, Chen Wang, and Dapeng Oliver Wu. 2024. FedLaw: Value-Aware Federated Learning With Individual Fairness and Coalition Stability. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2024).

[19] Qikai Lu, Di Niu, Mohammadamin Samadi Khoshkho, and Baochun Li. 2024. Hyperflora: Federated learning with instantaneous personalization. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 824–832.

[20] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. 2022. Layer-wised model aggregation for personalized federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10092–10101.

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

[22] mnmoustafa and Mohammed Ali. 2017. Tiny ImageNet. https://kaggle.com/competitions/tiny-imagenet. Kaggle.

[23] Lokesh Nagalapatti, Ruhi Sharma Mittal, and Ramasuri Narayanam. 2022. Is your data relevant?: Dynamic selection of relevant data for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7859–7867.

[24] Theodora Nevrataki, Anastasia Iliadou, George Ntolkeras, Ioannis Sfakianakis, Lazaros Lazaridis, George Maraslidis, Nikolaos Asimopoulos, and George F Fragulis. 2023. A survey on federated learning applications in healthcare, finance, and data privacy/data security. In *AIP Conference Proceedings*, Vol. 2909. AIP Publishing.

[25] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1622–1658.

[26] Jiaming Pei, Wenxuan Liu, Jinhai Li, Lukun Wang, and Chao Liu. 2024. A review of federated learning methods in heterogeneous scenarios. *IEEE Transactions on Consumer Electronics* (2024).

[27] Hongyan Peng, Tongtong Wu, Zhenkui Shi, and Xianxian Li. 2023. Fedef: Federated learning for heterogeneous and class imbalance data. In *2023 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 619–624.

[28] Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. 2022. Generalized federated learning via sharpness aware minimization. In *International conference on machine learning*. PMLR, 18250–18280.

[29] Pushpa Singh, Murari Kumar Singh, Rajnesh Singh, and Narendra Singh. 2022. Federated learning: Challenges, methods, and future directions. In *Federated Learning for IoT Applications*. Springer, 199–214.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[31] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems* 33 (2020), 7611–7623.

[32] Shiqiang Wang and Mingyue Ji. 2022. A unified analysis of federated learning with arbitrary client participation. *Advances in Neural Information Processing Systems* 35 (2022), 19124–19137.

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[34] Xiaotong Yuan and Ping Li. 2022. On convergence of fedprox: Local dissimilarity invariant bounds, non-smoothness and beyond. *Advances in Neural Information Processing Systems* 35 (2022), 10752–10765.

[35] Feilong Zhang, Yinchuan Li, Shiyi Lin, Junjun Jiang, Xianming Liu, et al. 2023. Large sparse kernels for federated learning. (2023).

[36] Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. 2022. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning*. PMLR, 26311–26329.

[37] Jie Zhao, Xinghua Zhu, Jianzong Wang, and Jing Xiao. 2021. Efficient client contribution evaluation for horizontal federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3060–3064.