HyperFedNet: Communication-Efficient Personalized Federated Learning Via Hypernetwork

Xingyun Chen
Qingdao University
2021020697@qdu.edu.cn

Yan Huang
Kennesaw State University
yhuang24@kennesaw.edu

Zhenzhen Xie
Shandong University
xiezz21@sdu.edu.cn

Junjie Pang
Qingdao University
pangjj@qdu.edu.cn

ABSTRACT

In response to the challenges posed by communication cost and non-independent and identically distributed (non-IID) data in Federated Learning, we introduce HyperFedNet (HFN), a novel architecture that incorporates hypernetwork to revolutionize parameter aggregation and transmission in FL. Traditional FL approaches, characterized by the transmission of extensive parameters, not only incur significant communication overhead but also present vulnerabilities to privacy breaches through gradient analysis. HFN addresses these issues by transmitting a concise set of hypernetwork parameters, thereby reducing communication costs and enhancing privacy protection. Upon deployment, the HFN algorithm enables the dynamic generation of parameters for the basic layer of the FL main network, utilizing local database features quantified by embedding vectors as input. Through extensive experimentation, HFN demonstrates superior performance in reducing communication overhead and improving model accuracy compared to conventional FL methods. By integrating the HFN algorithm into the FL framework, HFN offers a solution to the challenges of communication cost and non-IID data.

VLDB Workshop Reference Format:
Xingyun Chen, Yan Huang, Zhenzhen Xie, and Junjie Pang. HyperFedNet: Communication-Efficient Personalized Federated Learning Via Hypernetwork. VLDB 2024 Workshop: 3rd International Workshop on Large-Scale Graph Data Analytics (LSGDA 2024).

1 INTRODUCTION

In recent years, there has been a growing emphasis on data security due to increased awareness of privacy protection and the evolution of laws. However, in the era of big data, the continuous generation of diverse data by electronic devices poses challenges. This data contains personal privacy information and is unsuitable for transmission to central servers for data mining or machine learning purposes. For example, input prediction technology has significantly enhanced the accuracy of word suggestions, making typing more convenient and enjoyable. While training neural networks requires substantial data support, many owners of private data are reluctant to share it. Conversely, the number of personal devices capable of processing data is increasing. In this context, Federated Learning (FL) has emerged as a solution to address this dilemma. FL is a novel distributed technique that enables multiple parties to collaborate in training neural network models by transmitting only the parameters of the training network without sharing the dataset. This collaborative approach allows users to achieve common goals and benefit from the learned models while safeguarding their privacy. However, the development of FL faces challenges related to communication cost and data heterogeneity. As device computing power increases, the complexity of model structures grows, necessitating the transmission of hundreds of thousands or even millions of parameters between users' devices and the server per round. Consequently, the communication traffic overhead for users becomes substantial. This issue often leads to devices participating only when connected to WiFi, limiting the number of available devices and resulting in suboptimal model learning for FL tasks. Additionally, data heterogeneity arises from variations in user behavior, domain differences, and the disparity in data volume between individual users and institutional users. Data heterogeneity poses a significant obstacle that hinders the improvement of model accuracy and can even impede model convergence. Effectively conserving communication resources and achieving personalized federated learning (pFL) are crucial yet highly challenging aspects in the field of FL. Existing FL methods have not yet addressed these two challenges simultaneously.

Our work introduces a novel algorithm called HyperFedNet (HFN) to achieve communication efficiency and pFL. First, different embedding vectors are used to represent the convolutional kernel features in the convolutional neural networks of different users, indirectly indicating the data characteristics possessed by the users. Second, a hypernetwork, essentially a small neural network, is used to parameterize the main network of FL. This allows for communication using the more compact parameters of the hypernetwork instead of the main network's parameters. The hypernetwork acts as a coordinate map of a low-dimensional manifold in the embedding vector space, where each unique user model structural parameter is constrained and parameterized by the embedding vector[15]. Finally, the personalized layer of the main network is fine-tuned to ensure that the basic layer parameters generated by the hypernetwork are better aligned with the personalized layer parameters.

Our contribution are listed as follows:

Proceedings of the VLDB Endowment. ISSN 2150-8097.
Junjie Pang is the corresponding author.

(1) We propose a pFL method that reduces communication overhead and improves accuracy compared to existing algorithms, offering a tailored balance between model accuracy and communication needs.
(2) We introduce an innovative parameter transmission strategy in FL, utilizing aggregation of small model parameters, which reduces transmission costs and improves data security.
(3) Our extensive experiments demonstrate the proposed HFN method's superiority over current FL algorithms in communication efficiency, convergence speed, and personalization accuracy.
(4) We highlight the integration compatibility of the HFN algorithm with existing algorithms, showing improved performance and potential to enhance current FL approaches.

In summary, our work contributes by proposing a pFL method that reduces communication volume and improves security through a novel parameter transmission idea. Through extensive experiments, the paper demonstrates the effectiveness of the proposed method and its potential for integration with other algorithms.

The paper is organized as follows: Section II presents some work done by other researchers in related areas. Section III adds some necessary background knowledge and problem definition. In Section IV, the HFN method is formulated. The experiments in Section V show the advancement of HFN and the effect of combining it with other algorithms. In Section VI we summarize our work.

2 RELATED WORK

2.1 Personalized Federated Learning

Researchers have proposed many pFL methods. Data augmentation is a technique that transforms or expands the original data to generate new training samples to balance data heterogeneity. In order to balance the differences between clients, some researchers have proposed the use of data augmentation[3]. In [1], only the base layer is aggregated while retaining the personalized layer. However, this approach requires the participation of all users in every round, which is not feasible in a federated learning environment. [4, 9] finds an initial shared model based on migration learning, knowledge distillation, and meta-learning, respectively, and based on the initial model, the users fine-tune the model locally to obtain their own personalized model. In addition, [12] fixes the personalization layer during training, and after convergence, the local data is used to fine-tune the model to obtain a localized model.

The use of data augmentation, meta-learning, and other techniques in FL introduces the risk of data leakage, as it involves some form of data sharing, which contradicts the original purpose of FL. In contrast, the HFN adheres to the fundamental principles of FL without exchanging any data. A single global model is not well-suited for highly non-independent and homogeneously distributed data scenarios. Training the network using the model decoupling approach still subjectively transmits all feature knowledge to the user without targeting personalized knowledge for the main network. As a result, users are unable to obtain their own unique knowledge, leading to slow convergence, high communication overhead, and poor modeling results. The HFN algorithm, on the other hand, generates different knowledge for different users, enhancing convergence speed, and improving the model's effectiveness.

2.2 Efficient Communication in Federated Learning

FL is a distributed machine learning approach in which model parameters and updates need to be exchanged frequently between users for collaborative model training. The high communication cost not only has an impact on the overall performance, but also poses a challenge to devices with limited bandwidth and energy, thus constraining the development of FL[10]. A common approach is to compress and encode model parameters using compression techniques to reduce the bandwidth required for transmission[2, 14]. On the other hand, some work focuses on reducing the frequency of communication. For example, researchers have proposed strategies to increase the number of local computation epochs[11]. However, determining the optimal number of local computation epochs is a key issue, and to address this issue. The study by Reisizadeh et al.[13] introduces the periodic averaging method and quantizes its updates based on the quantizer's accuracy before uploading to the parameter server, thereby lowering communication overhead. Some academics combine methods from other fields, He et al.[6] proposed to utilize knowledge migration instead of traditional transmission transfer.

Compressing and encoding model parameters can introduce noise and lead to information distortion, potentially resulting in reduced model performance. Additionally, the compression and decompression processes between users and the server incur computational overhead. This overhead becomes particularly significant for large models, which may outweigh the benefits gained from compression. Moreover, when employing optimization methods that increase local computing rounds, the heterogeneity among users, including differences in computing capabilities and data distribution, can pose challenges. Adding more local computational epochs may further amplify this variability, leading to poorer training performance for certain users. Quantization operations, similar to model compression, can result in information loss. Representing the model with low-precision values may not accurately capture the intricacies and complexity of the model, necessitating more training iterations to compensate for the loss of detail. Moreover, when areas such as knowledge transfer are combined with FL, transmitting only feature maps and related logs without transmitting network parameters can lead to information loss. Consequently, the server can only use local information for training and may not be able to acquire a complete understanding of the model.

3 PRELIMINARIES

3.1 Federated Learning

Assume that in a FL scenario, there is a FL server and K users, where user $k (k \in \{1, 2, \ldots, K\})$ has a personal data set $\mathcal{D}_k$. Before training, the FL server will initialize a global model to solve the objective function. In each iteration cycle, the server randomly selects some users to participate in training and delivers the global model parameters. After the selected user receives the global model, it is set as a local model and trained using its local data. After training, the locally updated model parameters are transmitted

back to the server. The server uses an aggregation algorithm to aggregate the parameters uploaded by the user, generates a new global model, and performs the next round of iterative training until the preset accuracy is reached or the model converges. During the FL training process, a set of parameters w is found for the global model to minimize the overall loss. The objective function is defined as:

$$\min_{w} \left[\frac{1}{K} \sum_{k=1}^{K} f_k(w) \right] \tag{1}$$

Among them, f_k represents the loss function of user k, which is defined as:

$$f_k(w) = \mathbb{E}_{(x,y)\sim p_k} \mathcal{L}(w; (x, y)) \tag{2}$$

where p_k represents the probability distribution of data set $\mathcal{D}_k$ generated by user k, and $\mathcal{L}(w; (x, y))$ represents the prediction loss function.

3.2 HyperNetwork

Hypernetwork is a special type of neural network structure, usually containing one or more hidden and output layers, which has dynamically adjustable parameters. Hypernetwork can be used to generate parameters of main networks, weight, or architectures for other neural networks[5]. If a hypernetwork is used, the parameters of the main network will no longer be learned by the main network itself, but the parameters will be set by receiving the output of the hypernetwork. The hypernetwork transforms this input into parameters for other neural networks through a complex internal mechanism. These generated parameters can be seen as a kind of search space for exploring the possibilities of different network architectures or weights. The generated parameters are used to construct a specific neural network architecture. This process can be translated into an actual network architecture by some predefined rules or algorithms, e.g., using generative adversarial networks or by hypernetwork tuning. The generated network architecture is used for forward and backpropagation of the training data.

3.3 Problem Definition

The pFL allows each user to build his or her own personalized model to better adapt to the characteristics of local data. pFL can better improve the performance and adaptability of the model.

The personalization algorithm HFN we designed can generate K sets of parameters based on its own characteristic input, and splice it with a locally retained personalization layer, so that each user can learn parameters w_k that adapt to the local data distribution p_k to achieve better forecasting. That is, the goal of HFN can be expressed as:

$$w^* = \arg\min_{w} \left[\frac{1}{K} \sum_{k=1}^{K} f_k(w_k) \right] \tag{3}$$

Where $w = \{w_1, ..., w_K\}$ is the personalized parameter set of all users. w^* is the current optimal solution obtained through continuous iterative learning as FL progresses.

In order to solve w^* for personalization while achieving privacy enhancement and communication efficiency, our work expects to explore a new FL framework in which each user is able to complete the training by simply uploading a summary of the users' local model. Therefore, the pFL problem of our work turns to how to establish a way to automatically generate model summaries that adaptively complete the training of FL under external conditions.

4 METHDOLOGY

4.1 Framework Overview

We propose the HFN algorithm with the architecture shown in Fig.1. The model running on the user consists of two parts, one for the hypernetwork and the other for the FL main network. The hypernetwork takes the user's local embedding vector as input and outputs parameters to the basic layer of the main network based on the characteristics of representation of embedding vectors. In the main network, it accepts the parameters and will perform the prediction training task locally. As input embedding vectors v to the hypernetwork, they correspond to different filters, which are used to characterize some potential features of the filters, essentially reacting to the characteristics of different user databases, in order to help the hypernetwork to accomplish the output localization, as well as to better cope with non-iid data. These embedding vectors v are fed as inputs into the hypernetwork HN to generate the basic layer parameters w_θ of the main network. In the setting, each user's classification layer is retained as a personalization layer in the respective main network model, and the retention of these layers contributes to personalization. The basic layer parameter w_θ is combined with the personalization layer parameter w_β to consist of the complete main network parameter w. When performing forward propagation, the corresponding prediction y_i is made based on the input x_i. If it is in the training phase, a local update is also required, and the update of the hypernetwork in the backpropagation is based on the gradient of the main network. In the parameter exchange phase, the HFN architecture communicates by transmitting the hypernetwork parameters φ. This small and sophisticated network, through aggregation, can communicate its own learned knowledge and ability to generate parameters with other users, and continuously improve the ability to refine the output main network parameters.

4.2 pFL Based on Hypernetwork

Directly transmitting the parameters of each filter in a CNN leads to heavy communication overhead. We attempt to adopt an approach that takes advantage of the powerful generalization and mapping capabilities of the hypernetwork to generate a large number of parameters in the convolutional layer of the main network by mapping them through the hypernetwork. First, the main network parameter w is decomposed into the basic layer parameter, i.e., the convolutional layer parameter w_θ, and the personalized layer parameter w_β. Then the process of communicating the aggregate of the basic layer parameter w_θ is transformed into a learning problem by introducing a hypernetwork, which allows the hypernetwork parameter φ with fewer parameters to be aggregated instead of the main network, thus reducing the communication overhead in FL. At the same time, we keep the personalized layer parameter w_β locally, and each user can personalize the training and adjustment according to his/her own data characteristics. This helps to improve the adaptability and performance of the model to better fit the characteristics of local data.

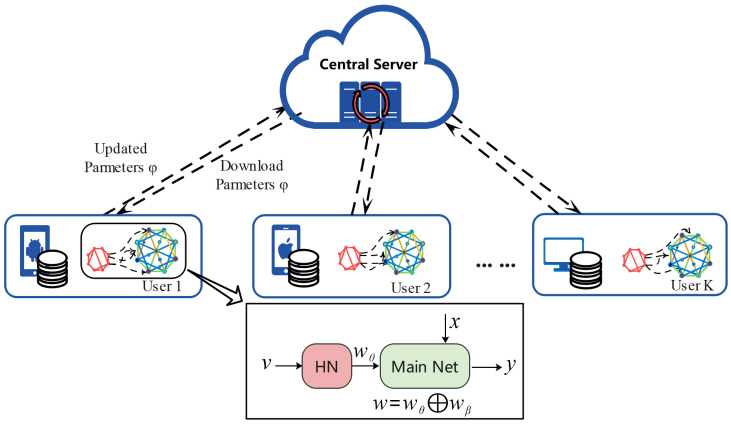

Figure 1: HFN Framework: Small red networks are hypernetwork and large colored ones are main networks. During parameter aggregation, only the hypernetwork parameters are aggregated.

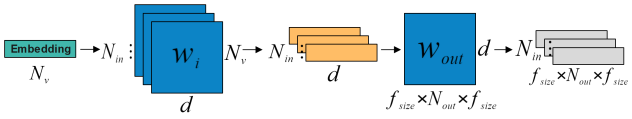

Figure 2: Hypernetwork Model Structure.

The convolutional layer in a convolutional neural network is composed of many filters. Every filter extracts a specific texture[16]. Each filter for each user needs to have a embedding vector v_k^j to represent the potential features of the filter. The purpose of quantizing the convolutional layer filters by embedding vector is to extract potential features. The embedding vector will help the hypernetwork to specify what kind of texture needs to be detected by the filter parameters that will be generated. The learnable embedding vector is denoted by v_k^j. Where k denotes the user serial number and j denotes the serial number of filters in the convolutional layer. The size of the embedding vector can be flexibly adjusted, A larger embedding vector can capture more feature information, and provide richer input information to the hypernetwork.

For convenience, we use $h(v_k^j;\varphi)$ to denote the hypernetwork, and φ to represent the hypernetwork parameters, with different v_k^j inputs, the hypernetwork $h(v_k^j;\varphi)$ can output the corresponding parameters. Each filter in the main network includes $N_{in} \times N_{out}$ kernels, and the size of each kernel is $f_{size} \times f_{size}$, which are mainly responsible for extracting various tiny features of the image for feature learning. The parameters of the filters use the $F_j \in \mathbb{R}^{N_{\text{in}}\, f_{\text{size}} \times N_{\text{out}}\, f_{\text{size}}}$ to indicate that, for each F_j, a hypernetwork is used to receive an embedding $v^j \in \mathbb{R}^{N_v}$ and to predict the outputs F_j, where N_v is the size of the embedding vector. The hypernetwork we use is a two-layer linear network. The hypernetwork model flow is shown schematically in Fig.2, where the embedding vector v_k^j is linearly mapped as input to N_{in} distinct matrices $W_i \in \mathbb{R}^{d \times N_v}$, $i = 1, ..., N_{in}$ and the bias vector $B_i \in \mathbb{R}^d$, $i = 1, ..., N_{in}$, where d is the size of the hidden layer in the hypernetwork. The input size of the second layer is d, using a matrix $W_{out} \in \mathbb{R}^{f_{size} \times N_{out} f_{size} \times d}$ and a bias matrix $B_{out} \in \mathbb{R}^{f_{size} \times N_{out} f_{size}}$ processed. Finally, the output is spliced into the parameters of a basic filter in a certain order.

An embedding vector v_k^j represents a basic filter F_j in the main network. The dimensions of the filters used for convolution operations in different convolution groups may be different, but are generally integer multiples of the basic filter. For example, use an embedding vector v_k^j to represent the basic filter F_j of 16×16. If we want to represent $F_{32\times64}$, we can concatenate the hypernetwork outputs corresponding to multiple embedding vectors as parameters of a single filter. The following shows a 32×64 dimensional filter stitched together from 8 basic dimensions 16×16.

$$F_{32\times64} = \begin{pmatrix} F_1 & F_2 & F_3 & F_4 \\ F_5 & F_6 & F_7 & F_8 \end{pmatrix}$$

Generating all the parameters of the convolutional layer needed for a user k requires all the embedding vector $\{v_k^j | j = 1, 2, ...\}$, and for convenience, v_k is used to represent all the feature embeddings of user k. Based on the above settings, the pFL objective of the HFN is adjusted to:

$$v^*, \varphi^*, w_\beta^* = \arg \min_{v,\varphi,w_\beta} \left[\frac{1}{\mathrm{K}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{f_k}(\mathrm{h}(\mathrm{v_k};\varphi);\mathrm{w}_{\beta_\mathrm{k}}) \right] \tag{4}$$

where $v = \{v_1, \ldots, v_K,\}$ is the set of embedding vectors that can be learned by all users, and $w_\beta = \{w_{\beta_1}, \ldots, w_{\beta_K}\}$ is the set of parameters of the personalization layer for all users. Instead of the traditional main network parameter w, what is aggregated in FL's server is the hypernetwork parameter φ, so the gradient ∇w of the main network needs to be further back-propagated to the hypernetwork, and the hypernetwork φ is updated based on ∇w with $\varphi_{t+1} = \varphi_t - \lambda \nabla \varphi_t$ (λ is the learning rate), thus the hypernetwork can perform end-to-end learning.

5 EXPERIMENTS

5.1 Experiment Setup

Datasets and model: We used four common datasets: MNIST[8], FMNIST[17], CIFAR-10 and CIFAR-100[7]. We divided each dataset into 80% training set and 20% test set and divided them into 100

Table 1: Convolutional Neural Network Settings

Group Name	MNIST	FMNIST	CIFAR10	CIFAR100
conv1	$\begin{bmatrix} 3\times3, 16 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 16 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 16 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 32 \end{bmatrix}$
conv2	$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times6$
conv3	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times6$
conv4	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times6$

users. The test set and the training set on each user have the same data distribution, and the training set does not overlap with the test set data. The dirichlet distribution $Dir(0.5)$ is used to construct the non-iid case.

We used ResNet as the main network for experiments to validate the effect of the parameters generated by the hypernetwork. The architectures of the CNNs dealing with different datasets are shown in Table 1. Convolution, batch normalization and ReLU activation are performed in the residual block in this order. The dimensions of the kernel are all 3×3. After convolution, an appropriate average pooling layer is used for dimensionality reduction. Finally, a fully connected classification layer is used. In the privacy experiment, we use the LeNet model.

Baselines We compared HFN with the state-of-the-art FL algorithm. The unique hyperparameters of each algorithm in the experiment were tested by referring to the default configuration of the original paper. We implemented the following benchmark algorithms in the experiment: 1) Centre: Centralize all data for training, imitating the traditional way of uploading data to the data center. 2) FedAvg: This is the most important algorithm for FL, and it is effective in various scenarios. 3) Local: There is no parameter exchange, but each user only uses his or her own data for training locally. 4) FedBabu: The algorithm for updating and aggregating the model body needs fine-tuning after convergence. 5) FedProx: by adding proximal terms so that the user model does not deviate from the global model in order to deal with non-iid scenarios, we enumerated its unique hyperparameters $\mu = \{1, 0.1, 0.01\}$ and chose the optimal results for each experiment. 6) FedGen: Improve FL accuracy by training a feature generator. 7) FedDyn: Propose a dynamic regularizer for each user in each round to promote consistent solutions for local and global users. 8) pFedSim: Personalized algorithm based on model similarity. 9) FedPer: A method of retaining the personalization layer locally, which can combat the adverse effects of statistical heterogeneity. 10) FedBN: Solve the problem of FL data heterogeneity by adding a batch normalization layer to the local model. 11) FedRep: Train the classifier and feature extractor in sequence, and only aggregate the feature extractor. 12) pFedla: Deploys a hypernetwork on the server side for each user to give the user hierarchical aggregation weights. 13) FedFomo: The optimal weighting of a customer is given to aggregate the model by calculating how much the customer can benefit from other customers' models.

Since HFN and some of the algorithms require a fine-tuning step, to be fair, we add 4 local epochs of fine-tuning to all algorithms to ensure fairness and to evaluate their personalized accuracy.

Settings The embedding vector size of HFN can be adjusted. In this general experiment, it is fixed, 64 when processing MNIST, and 128 for the rest. The experiment uses SGD as the optimizer, uses Nesterov Momentum, momentum is 0.9, weight decay is 5e-4, local epoch and fine-tuning epoch are 4, the batch size is set to 128, the total number of users is 100, the joining rate is 0.25, MNIST and FMNIST global communication round is 90, CIFAR10 and CIFAR100 are 150. We run multiple experiments on the learning rates of all the algorithms and select the best results in each experiment, including multi-step learning rates decaying from 0.1 and fixed learning rates of 0.1, 0.01, and 0.001.

5.2 Hypernetwork for Parameter Generation

In this experiment, there are a total of 20 users, each group of 4 users, with user numbers 0-3 as the first group, and so on, a total of 5 groups, and the users in the group have similar data distribution. Each user has 2 classes (MNIST, FMNIST, CIFAR10 dataset) or 10 classes (CIFAR100 dataset). We then evaluate the similarity of the generated parameters through cosine similarity. As shown in Fig.3, the images show the similarity of the hypernetwork generation parameters under the MNIST, FMNIST, CIFAR10 and CIFAR100 data sets. The horizontal and vertical coordinates in the figure are user numbers. The lighter the color of the grid, the higher the similarity of the user models corresponding to the x-axis and y-axis, and a similarity value of 1 means that the user models are identical. We use the blue border of 4×4 to highlight the similarity corresponding to the user models of the same group. It can be clearly seen from the figure that the color corresponding to the model within the group is significantly lighter, i.e., the similarity is higher, indicating that the user basic layer parameters within the group with similar data distribution have a higher similarity than the parameters in other groups.

Users based on similar data distribution should have similarity in their embedding vector matrices. The parameters generated by the hypernetwork will be subject to similar conditions. This means that the hypernetwork will tend to generate parameters that adapt to similar data distributions, thus making the main network parameters generated by different users similar. However, it should be noted that even if users within a group have similar data distribution, the data content of each user is not completely consistent. The parameters generated by the hypernetwork for different users may vary to some extent. However, these differences should be within a certain range and maintain overall similarity.

This experiment shows that the hypernetwork we use can generate the basic layer parameters suitable for users with different data

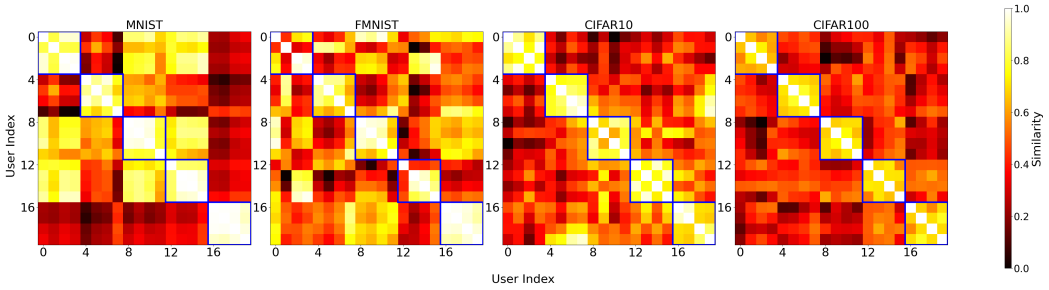

Figure 3: Similarity of main network parameters generated by hypernetwork for different users.

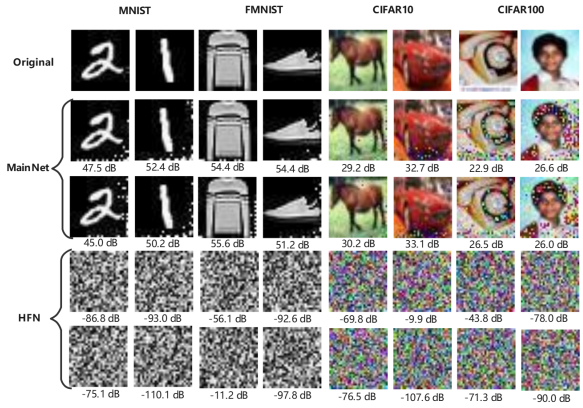

Figure 4: Privacy attack results: The second and fourth rows are the results of the DLG attack. The third and fifth rows are the results of the iDLG attack.

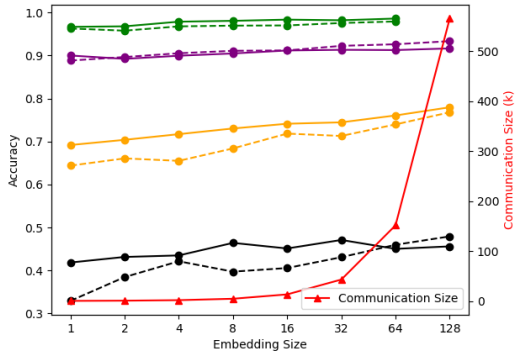

Figure 5: The impact of different embedding sizes in HFN on communication size and accuracy.

distributions by inputting the corresponding embedding vector to extract the unique features.

5.3 Privacy Evaluation

In order to evaluate the security of FL algorithms, we have done on four different datasets attack experiments.

Peak signal-to-noise ratio (PSNR) is a measure of the reconstruction quality of an image compression signal. PSNR is reported under each recovered image. We calculate PSNR to represent the similarity between the original image and the reconstructed image. It is calculated as $PSNR = 20 \cdot \log_{10}\left(\frac{255}{\sqrt{MSE}}\right)$, where MSE is the Mean Squared Error. The larger value of PSNR indicates that the attacked reconstructed image is more similar to the original image. According to Fig.4 shown, we can intuitively observe that the traditional FL method transmits the main network information with the risk of leaking data, which seriously threatens the basic principle of FL. On the contrary, the information transmitted by HFN algorithm is completely resistant to the attacks of DLG[19] and iDLG[18].

5.4 The Affect of Embedding Vector Sizes on HFN

HFN introduces a latent hyperparameter, the size of the Embedding Vector, which is related to the performance and communication overhead of FL. To investigate the relationship between them, we conducted experiments to explore the impact of different embedding sizes on accuracy and communication volume. As shown in Fig.5, the red line represents the Cost per Round (CPR) for individual users, which corresponds to the right y-axis. CPR is composed of the parameter quantity transmitted by each user in one round, including both upload and download. It is important to note that the values represent the number of parameters transmitted in the network, not the actual network traffic. The other colors represent the accuracy achieved after convergence in FL, which corresponds to the left y-axis. The green line corresponds to MNIST, the purple line corresponds to FMNIST, the orange line corresponds to CIFAR-10, and the black line corresponds to CIFAR100. Comparing the iid and non-iid (solid and dashed lines), it can be observed that in the non-iid setting, as the embedding vector size increases, the improvement in accuracy is more significant compared to the iid case. This is because in the non-iid scenario, the variations in data distributions among different users require the convolutional layers of different users to extract more diverse feature information. This necessitates a larger embedding vector for assistance, resulting in a notable increase in accuracy as the embedding vector size increases.

Regarding communication overhead, the communication cost of HFN increases with the increase in embedding size. The communication overhead of most FL algorithms is similar to FedAvg. We calculated the CPR of FedAvg for individual users for the MNIST, FMNIST, CIFAR10, and CIFAR100 datasets, which are 155K, 1127K, 1128K, and 4519K, respectively. Comparing the different embedding

Table 2: The accuracy of the algorithms under four databases

	MNIST		FMNIST		CIFAR10		CIFAR100	
Aggregation method	non-iid	iid	non-iid	iid	non-iid	iid	non-iid	iid
Centre	99.49%		93.60%		84.78%		54.78%	
FedAvg	98.67%	98.96%	92.40%	90.12%	69.95%	62.12%	33.85%	33.67%
Local	94.82%	93.35%	86.94%	76.74%	57.05%	30.45%	20.15%	6.98%
FedBabu	98.94%	98.81%	**94.12%**	91.04%	73.34%	64.39%	43.79%	36.67%
FedProx	98.85%	**99.04%**	93.10%	90.39%	69.89%	60.88%	31.14%	34.17%
FedGen	97.63%	98.18%	91.78%	89.36%	67.06%	62.75%	26.32%	16.06%
FedDyn	98.73%	99.01%	93.20%	90.64%	67.76%	61.60%	29.23%	34.37%
pFedSim	98.92%	99.01%	92.41%	90.47%	70.45%	65.66%	43.74%	37.30%
FedPer	99.05%	98.84%	92.36%	89.59%	72.67%	62.63%	31.15%	17.10%
FedBN	**99.14%**	98.94%	91.64%	89.74%	69.16%	33.77%	35.39%	32.08%
FedRep	98.10%	98.01%	92.36%	87.18%	75.44%	65.59%	28.77%	12.38%
pFedla	98.40%	98.49%	92.78%	88.64%	66.18%	56.98%	29.54%	26.17%
FedFomo	93.95%	92.02%	86.09%	76.06%	56.29%	24.85%	17.71%	12.02%
HFN(Ours)	97.91%	98.59%	93.33%	**91.64%**	**76.78%**	**77.95%**	**47.92%**	**45.60%**

Table 3: Communication cost for a single user at each level of the different algorithms when the dataset is CIFAR10-iid

	CPR	10%	20%	30%	40%	50%	60%	70%
HFN_128	0.57M	0.00M (0×)	1.13M (2×)	7.92M (14×)	10.75M (19×)	15.84M (28×)	23.19M (41×)	35.63M (63×)
FedAvg	1.13M	1.13M (1×)	5.64M (5×)	15.79M (14×)	46.24M (41×)	46.24M (41×)	91.36M (81×)	\
FedBabu	1.13M	1.13M (1×)	2.25M (2×)	7.89M (7×)	25.91M (23×)	46.19M (41×)	46.19M (41×)	\
FedProx	1.13M	1.13M (1×)	14.66M (13×)	46.24M (41×)	46.24M (41×)	72.18M (64×)	92.48M (82×)	\
FedGen	1.13M	1.13M (1×)	28.29M (25×)	39.60M (35×)	46.39M (41×)	89.38M (79×)	91.65M (81×)	\
FedDyn	1.13M	1.13M (1×)	3.38M (3×)	14.66M (13×)	34.96M (31×)	46.24M (41×)	91.36M (81×)	\
pFedSim	1.13M	1.13M (1×)	5.64M (5×)	15.79M (14×)	46.24M (41×)	46.24M (41×)	109.40M (97×)	\
FedPer	1.13M	1.13M (1×)	6.76M (6×)	16.90M (15×)	28.16M (25×)	46.19M (41×)	91.25M (81×)	\
FedBN	1.13M	1.13M (1×)	4.51M (4×)	15.79M (14×)	46.24M (41×)	46.24M (41×)	91.36M (81×)	\
FedRep	1.13M	0.00M (0×)	9.01M (8×)	15.77M (14×)	40.56M (36×)	78.86M (70×)	\	\
pFedla	1.13M	0.00M (0×)	16.92M (15×)	27.07M (24×)	46.24M (41×)	91.36M (81×)	\	\
FedFomo	3.95M	0.00M (0×)	15.79M (4×)	35.53M (9×)	221.06M (56×)	307.91M (78×)	\	\

Table 4: The effect of HFN combined with other algorithms

	MNIST				FMNIST				CIFAR10				CIFAR100			
	non-iid		iid		non-iid		iid		non-iid		iid		non-iid		iid	
	Acc	Comm	Acc	Comm	Acc	Comm	Acc	Comm	Acc	Comm	Acc	Comm	Acc	Comm	Acc	Comm
HFN+FedAvg	-1.02%	15.15%	-0.75%	13.46%	0.98%	50.14%	1.41%	61.29%	13.38%	51.95%	15.33%	74.60%	13.13%	78.80%	13.63%	48.98%
HFN+FedProx	-0.90%	18.27%	-0.55%	19.41%	0.25%	54.32%	1.02%	71.31%	8.14%	78.88%	16.98%	50.55%	16.19%	164.75%	12.51%	48.98%
HFN+FedDyn	-0.82%	22.08%	-0.55%	18.55%	-0.34%	60.64%	1.22%	50.14%	9.30%	140.62%	16.38%	70.93%	18.18%	172.58%	11.67%	51.34%
HFN+FedBN	-0.86%	18.40%	-1.29%	12.84%	-0.89%	71.45%	1.23%	50.14%	0.16%	51.07%	36.67%	94.83%	4.58%	158.78%	5.29%	49.55%
HFN+pFedla	-0.38%	12.00%	-0.21%	15.86%	0.22%	54.32%	3.13%	71.31%	10.92%	119.78%	20.17%	50.14%	16.52%	152.66%	19.51%	34.52%
HFN+FedFomo	3.95%	24.43%	6.47%	32.10%	6.49%	21.88%	15.53%	35.00%	17.19%	71.62%	50.23%	33.12%	20.87%	34.04%	28.54%	11.40%

sizes of HFN in Fig.5, the communication overhead is significantly higher. However, when employing an embedding vector size of 1, the CPR of HFN on the four datasets is only 640. Therefore, the HFN algorithm has an advantage in terms of communication overhead. This graph allows us to flexibly balance accuracy and communication overhead based on practical requirements, enabling the selection of an appropriate embedding vector size.

5.5 Performance Evaluation

The personalization accuracies under the same communication round limit are shown in Table 2. In most cases, the HFN algorithm demonstrates good performance, especially when the dataset is complex. When HFN comes to process the simpler MNIST dataset, average performance may be observed. This can be attributed to the relative simplicity of the MNIST dataset and the simpler model structure used. In the MNIST dataset, the image has a lower image

resolution and the digit patterns are relatively simple, so a simple model can be used to extract enough features for classification. In this case, the introduction of a hypernetwork may increase the complexity of the model and make the task relatively more complex, resulting in less remarkable accuracy. It is worth noting that hypernetwork were originally designed to handle more complex tasks and models. It is when applied to more complex datasets (CIFAR series) and models that the benefits of HFN become more significant. This is because the choice of model parameters can be more critical in complex tasks, and the hypernetwork can provide more appropriate parameters by adaptively generating model parameters that produce higher accuracy.

Communication cost has always been a problem in FL, but the HFN algorithm can solve this problem effectively. In Table 3, we show the communication cost of different algorithms during the training process when users are trained with the CIFAR-10 dataset (iid). Also, we list the communication costs incurred by each algorithm in reaching a certain accuracy threshold. The number in parentheses indicates the rounds in which the algorithm reaches that accuracy threshold, by which we can roughly determine the convergence speed of each algorithm. It can be seen that HFN not only has a significant advantage in convergence speed, but also has an unrivaled advantage in communication cost. It is worth noting that the advantage of HFN in reducing communication costs is due to its ability to generate model parameters locally without the need to transmit large amounts of main network parameters. This local generation of parameters reduces the amount of data transfer between users and improves the efficiency of FL. Therefore, HFN can better solve the problem of communication overhead in FL by reducing the amount of data transfer between users while producing better accuracy.

5.6 HFN with Other FL Algorithm

Another major advantage of HFN is that it can be easily combined with existing algorithms, by which the advantages of HFN can be attached to existing algorithms and benefit from other algorithms. As shown in Table 4, where ACC indicates how much the algorithm has improved on the original accuracy after fusion, for example, the accuracy of FedAvg at CIFAR10 non-iid is 69.95%, and HFN+FedAvg is 83.33%, then the corresponding value of ACC in the table should be recorded as 13.38%. We also counted the corresponding communication consumption of the algorithms from the beginning of training, all the way to the final accuracy, compared with the fused algorithms. The percentage in Comm means the percentage of the total communication consumption of the new algorithms after fusing the HFN algorithms to the original total consumption. It can be noticed that the communication efficiency of all the baseline algorithms is greatly improved by combining HFN. In some cases, the total amount of communication exceeds 100%, which is reasonable due to the fact that the convergence accuracy has been improved more than the original and more communication rounds are needed to achieve higher accuracy.

6 CONCLUSION

In this work, we explore and utilize the potential of hypernetwork in FL. Compared with the previous traditional architecture, using the hypernetwork instead of the main network for communication learning greatly saves the communication cost, improves safety and the powerful learning ability of the hypernetwork also improves the accuracy of the main network. We verified the performance of traditional algorithms and HFN under different datasets with different distributions through a large number of experiments, and then did many targeted experiments on HFN and fused this novel approach into the traditional FL method to achieve better results.

7 CITATIONS

REFERENCES

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818* (2019).

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

[3] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 59–71.

[4] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems* 33 (2020), 3557–3568.

[5] David Ha, Andrew Dai, and Quoc V. Le. 2016. HyperNetworks. arXiv:1609.09106 [cs.LG]

[6] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems* 33 (2020), 14068–14080.

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

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[9] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[10] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[11] 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.

[12] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2021. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042* (2021).

[13] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2020. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021–2031.

[14] Suhail Mohmad Shah and Vincent KN Lau. 2021. Model compression for communication efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[15] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*. PMLR, 9489–9502.

[16] Zhiqiang Xia, Ce Zhu, Zhengtao Wang, Qi Guo, and Yipeng Liu. 2016. Every filter extracts a specific texture in convolutional neural networks. *arXiv preprint arXiv:1608.04170* (2016).

[17] 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).

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

[19] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).