sensors

MDPI

Article

FedPSFV: Personalized Federated Learning via Prototype Sharing for Finger Vein Recognition

Haoyan Xu [1,2], Yuyang Guo [1,2], Yunzan Qu [1,2], Jian Guo [1,2,*] and Hengyi Ren [3]

[1] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; b22140827@njupt.edu.cn (H.X.)
[2] Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[3] College of Information Science and Technology & Artificial Intelligence, Nanjing Forestry University, Nanjing 210037, China
* Correspondence: guoj@njupt.edu.cn; Tel.: +86-189-5189-6722

Abstract

Finger vein recognition algorithms based on deep learning techniques are widely used in many fields. However, the training of finger vein recognition models is hindered by privacy issues and the scarcity of public datasets. Although applying federated learning techniques to finger vein recognition can effectively address privacy concerns, data heterogeneity across clients limits the performance of the models, especially on small datasets. To address these problems, in this paper, we propose a new federated finger vein recognition algorithm (FedPSFV). The algorithm is based on the federated learning framework, which increases the interclass distance of each dataset by sharing the prototypes among clients to solve the data heterogeneity problem. The algorithm also integrates and improves the margin-based loss function, which advances the feature differentiation ability of the model. Comparative experiments based on six public datasets (SDUMLA, MMCBNU, USM, UTFVP, VERA, and NUPT) show that FedPSFV has better accuracy and generalizability; the TAR@FAR = 0.01 is improved by 5.00–11.25%, and the EER is reduced by 81.48–90.22% compared to the existing methods.

Keywords: finger vein recognition; federated learning; margin-based loss function; prototype sharing

check for updates

Academic Editors: Sheryl Berlin Brahnam and Fangmin Xu

Received: 11 May 2025
Revised: 3 August 2025
Accepted: 12 August 2025
Published: 6 November 2025

Citation: Xu, H.; Guo, Y.; Qu, Y.; Guo, J.; Ren, H. FedPSFV: Personalized Federated Learning via Prototype Sharing for Finger Vein Recognition. *Sensors* **2025**, *25*, 6790. https://doi.org/10.3390/s25216790

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Finger vein recognition is a biometric technology that authenticates individuals by detecting vein patterns within a finger. In contrast to other recognition technologies, such as facial recognition and fingerprint recognition, finger vein recognition relies on blood flow imaging to enable liveness detection. This feature makes counterfeiting and stealing more difficult and renders the technology more suitable for locations with high security requirements.

Recently, deep learning techniques based on neural networks have become the mainstream methods for finger vein recognition [1–4]; these methods can automatically learn features and are highly scalable. Neural network training usually requires large datasets [5]. However, owing to legal restrictions such as privacy protection, research institutes and related organizations face many limitations in collecting and sharing finger vein data [6–8]. As a result, these organizations are only able to train models on a limited number of local datasets, resulting in low recognition accuracies of the obtained models.

Federated learning (FL), which allows participating clients to jointly train models without sharing datasets and only exchanges training parameters, is a potential solution to data privacy issues [9–12]. Currently, federated learning techniques have been applied in the fields of facial recognition and iris recognition [13–17] with good results, and some methods, such as FedFV [18], DDP-FedFV [19], and FVRFL [20], have been proposed in the field of finger vein recognition. These methods avoid privacy risks. However, due to differences in the acquisition methods, external environment, users, imaging quality, number of images, etc., the data from each client are extremely varied, which results in poor generalizability of the models trained by the existing methods. Performance degradation is particularly obvious for some clients with small data volumes [19].

To address the above limitations, this paper presents an algorithm named FedPSFV. FedPSFV addresses the data heterogeneity problem by sharing prototypes among different clients, which increases the generalizability of the model. Moreover, FedPSFV integrates a margin-based loss function to further increase the model's feature differentiation ability. The main research contributions of this paper are as follows.

(1) A novel federated finger vein recognition algorithm named FedPSFV is proposed. FedPSFV applies the concept of prototype sharing, which addresses the client data heterogeneity problem in federated learning. By sharing the prototype, the recognition performance of each client is improved.

(2) The margin-based loss function is presented and improved in the federated learning framework. This framework enables the margin-based loss function to adjust the spacing between samples more accurately on the basis of the edge intensity and increases the feature differentiation ability of the model.

(3) Open-set tests on six public heterogeneous finger vein datasets (SDUMLA, MMCBNU, USM, UTFVP, VERA, and NUPT) reveal that the FedPSFV algorithm is effective in addressing the data heterogeneity problem and outperforms the recognition performance of the existing classical methods.

The following sections are organized as follows. Section 2 reviews related work in the fields of finger vein recognition and federated learning. Section 3 presents the detailed implementation of FedPSFV. Section 4 discusses the performance of FedPSFV on the basis of the experimental results. Section 5 presents the conclusion and discusses the limitations of FedPSFV and potential directions for future work.

2. Related Work

2.1. Finger Vein Recognition

Finger vein recognition is an emerging biometric identification technology, and the main process has four stages: image acquisition, image preprocessing, feature extraction, and matching recognition. Feature extraction is the most critical step and affects the recognition accuracy of the entire system directly. Classical manual feature extraction algorithms include the repetitive line tracking (RLT) algorithm, the local maximum curvature (LMC) algorithm, the principal curvature (PC) algorithm, and the local binary pattern (LBP) method [21–23]. Overall, these methods have average recognition accuracy and poor scalability.

Recently, an increasing number of studies have applied deep learning algorithms to finger vein recognition. Feature extraction algorithms based on deep learning not only improve recognition performance but also simplify data processing operations. Hu et al. [24] trained a CNN containing 10 convolutional layers for finger vein recognition and achieved good results. Zhao et al. [25] combined center loss and Softmax loss and increased the accuracy and speed of finger vein recognition via dynamic regularization, which outperformed the classical deep learning algorithms on the MMCBNU_6000 and FV-USM datasets.

Song et al. [26] used composite images to train a modified DenseNet, which increased the feature extraction capability of the model. These methods can often achieve better results than traditional methods; however, they usually require large datasets to train the neural network [19]. It is often difficult for research institutions to obtain and share large finger vein datasets because of privacy protection, data security, and other limitations, which makes training models with high generalizability challenging.

2.2. Finger Vein Recognition via Federated Learning

Federated learning is a distributed machine learning framework that enables multiple participants to jointly train models without the sharing of local data. This framework protects the privacy of customer data and satisfies the requirements of the biometrics domain. Currently, federated learning-related algorithms have been continuously proposed [27–30] and are gradually being used in facial recognition, iris recognition, and other fields [31]. In the field of finger vein recognition, much related research is still in its infancy. FedFV [18] proposed an innovative parameter aggregation method that reduces the computational cost of the personalized aggregation method. DDP-FedFV [19] proposed a two-phase federated learning algorithm for the field of finger vein recognition. However, multiparty collaborative training also introduces a new problem: different clients have data with different data qualities and different data sizes. The existing federated learning frameworks have difficulty addressing these differences, and the trained models are not sufficiently discriminative of features and have large performance fluctuations on different client datasets.

To address these limitations, this paper increases the adaptability of the model to different datasets by sharing prototypes among different clients. Moreover, an improved margin-based loss function that improves the feature discrimination ability of the model is presented.

3. Methodology

This section describes the problem of the FL-based finger vein recognition algorithm, presents the overall framework of the FedPSFV algorithm, and presents the specific implementation of each module of the algorithm.

3.1. Problem Description

The method in this paper uses the federated learning framework. Assuming a total of m different clients, each client is denoted as C_k (k = 1, 2. . .m), and the total number of categories in the dataset of all clients is N. For the k-th client, its local dataset is denoted as $D_k \in X_k \times Y_k$. X_k and Y_k denote the feature space and label space, respectively, of the dataset of the k-th client.

In the FedPSFV algorithm, the model $\mathcal{F}(\omega; x)$ with the same architecture is shared among clients, where ω denotes the learnable weights of the model and x denotes the input features. The objective function of the FedPSFV algorithm is defined in Equation (1),

$$\min_{\omega} \sum_{k=1}^{m} \frac{|D_k|}{N} \mathcal{L}_{FedPSFV}(\mathcal{F}(\omega; x), y) \tag{1}$$

where y denotes the corresponding labeled value and $\mathcal{L}_{FedPSFV}$ denotes the loss function.

3.2. Framework of the FedPSFV Method

This section briefly describes the framework of the FedPSFV method. The overall framework of the algorithm, which is shown in Figure 1, contains two parts: the client side and the server side. The client side completes the preprocessing of the finger vein data before training, and then during each communication round of federated learning, after

the client has trained the local model, it computes the prototypes and uploads them and the model parameters to the server side. The server side receives and aggregates all the uploaded prototypes and model parameters and then sends the prototypes and the model parameters down to each client to participate in the next round of training.

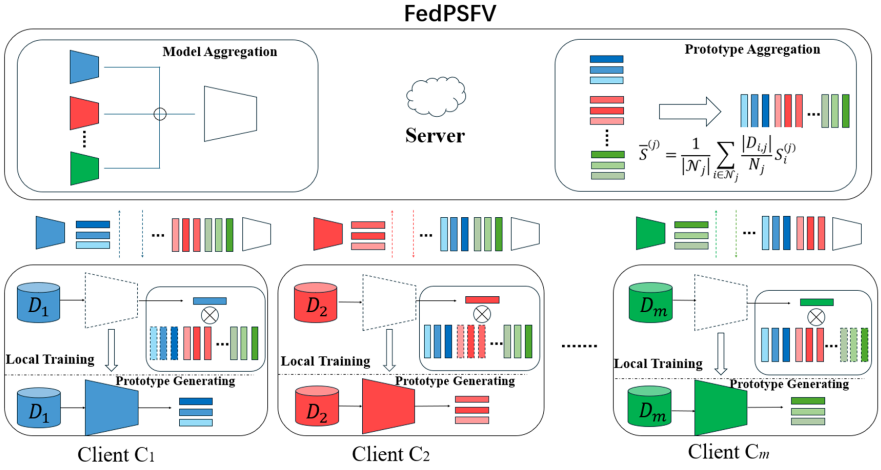


Figure 1. Framework of the FedPSFV algorithm, which contains m clients. The trapezoidal shape represents the feature representation layer of the model, and the horizontal rectangle represents the prototype. The vertical dashed rectangle represents the classifier weight matrix obtained from the local initialization of the clients, and the vertical solid rectangle represents the expanded classifier weight matrix. After completing a training epoch, each client generates prototypes and uploads them. The server then aggregates the prototypes and model parameters and sends the updated information back to the clients.

The specific implementation details of the FedPSFV algorithm are described in Sections 3.3 and 3.4. Section 3.3 describes the client-side training framework, which contains the data preprocessing, classification layer, and loss function of the FedPSFV algorithm, and Section 3.4 describes how to aggregate the model parameters and the prototypes on the server side.

3.3. Local Training Process

This section describes the training process of the local client, which includes data preprocessing, network training, prototype calculation, and loss calculation. Figure 2 shows the training process of the local client.

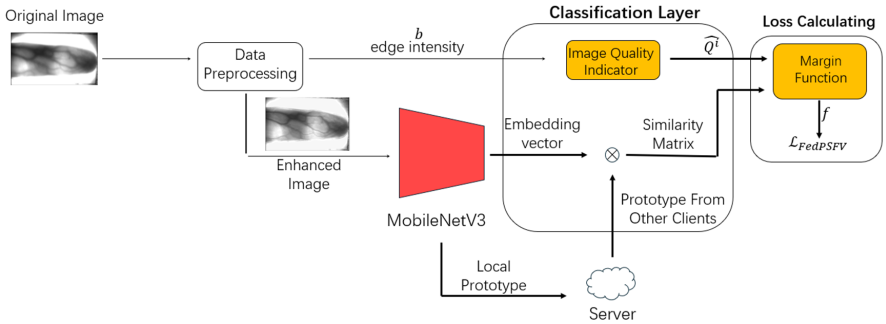


Figure 2. Local training process for the client. The data preprocessing module performs image enhancement and edge intensity calculations on the finger vein samples. The edge intensity is used as a quality evaluation metric to calculate the quality score of the image. The enhanced image enters the feature extraction layer to obtain the embedding vector and then enters the classification layer, which is multiplied by the classifier weight matrix composed of the prototype to obtain the vector angle and similarity matrix. Finally, the loss function is computed using an image quality indicator and the vector angle.

The next section describes the various client modules in detail.

3.3.1. Data Preprocessing

The FedPSFV algorithm has two main preprocessing tasks. The first task is image enhancement, which highlights the key details of the finger veins and facilitates the extraction of discriminative features by the subsequent model. The second task is to calculate the edge intensity for the subsequent image quality indicator. The main preprocessing process is shown in Figure 3.

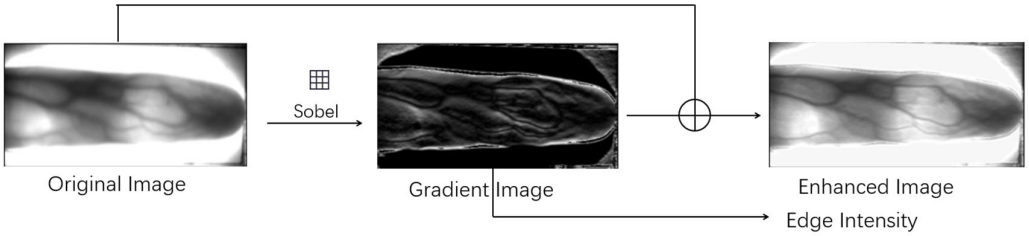


Figure 3. Preprocessing of finger vein samples. The gradient image is obtained by applying the Sobel operator to the original image. The enhanced image is then generated by adding the original image and the gradient image. The gradient image is also used to compute the edge intensity.

When the image enhancement process is performed on the client side, the gradient image I_g of the finger vein image I is calculated and then added to the original image to obtain the enhanced image I'. The process is shown in Equation (2),

$$I' = I + I_g \tag{2}$$

where the gradient image is computed via the Sobel operator, as expressed in Equation (3).

$$I_g = \begin{cases} 0, & if\ |G \times I| > T \\ |G \times I|, & if\ |G \times I| \leq T \end{cases} \tag{3}$$

Here, T represents the gradient threshold, which is usually 100, and G represents the Sobel operator. When the finger orientation of the finger vein image is vertical, the value of G is as shown in Equation (4). When the finger orientation is horizontal, G is transposed and substituted into Equation (3) for calculation.

$$G = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{4}$$

While the image is enhanced, the client will synchronize the calculation of the edge intensity b of the finger vein. The process is shown in Equation (5),

$$b = \sum_{px,py} I_g(px, py) \tag{5}$$

where $I_g(px, py)$ denotes the pixel value with coordinates (px, py) in the gradient image.

For the same category, the client calculates the edge intensity of all the samples of the category and calculates the average value as the edge intensity of the category.

3.3.2. Local Network Architecture

The local neural network model of the client consists of two parts: a feature extraction layer and a classification layer. The feature extraction layer of the FedPSFV algorithm uses

the MobileNetV3 network [32], and the classification layer uses the designed EdgeAdavein module. Each of these layers is described below.

The preprocessed images enter the feature extraction layer, which is the MobileNetV3 network. The feature extraction layer needs to calculate the local prototype after each round of training and upload it to the server; additionally, it needs to output the embedding vector to the classification layer during training for the calculation of the loss function.

The calculation process of the prototype is presented. A prototype is the center of a feature representation of a class and is usually expressed as the mean of the embedding vectors. Assume that client k has a dataset D_k of N_k categories and that $D_k = \left\{ D_k^1, D_k^2, \ldots\ldots D_k^{N_k} \right\}$, where D_k^j denotes the set of all training samples of category j on client k and S_k^j represents its corresponding prototype, which is calculated as shown in (6),

$$S_k^j = \frac{1}{\left| D_k^j \right|} \sum_{(x,y) \in D_k^j} f_k(x) \tag{6}$$

where $f_k(x)$ denotes the embedding vector obtained after the input feature x passes through the feature extraction layer and $\left| D_k^j \right|$ denotes the number of training samples of category j. At the end of each training batch, the client inputs the image data into the feature extraction layer to obtain the embedding vector and computes the local prototype via Equation (6), which is then uploaded to the server side with the model parameters.

After the feature extraction layer is processed, the embedding vectors of its output are fed into the classification layer. The FedPSFV algorithm, which designs the EdgeAdavein module as the classification layer, has two main tasks: (1) calculating the similarity matrix and (2) calculating the image quality indicator using the edge intensity.

The similarity matrix is constructed. The similarity matrix represents the degree of similarity between any two objects in the dataset, where each column vector vc_j represents the degree of similarity between all the samples of the current batch and the representative vector of category j. The similarity matrix is obtained by multiplying the normalized embedding vector by the classifier weight matrix. In the classification layer, the similarity matrix is obtained by multiplying the embedding vector by the classifier weight matrix.

A classifier weight matrix is a parameter matrix used in machine learning to map input features into category predictions and to represent the centers of categories [33,34]. In federated learning, the traditional classifier weight matrix represents only the centers of all the categories in the local dataset. This approach results in a similarity matrix that represents only the similarity between the current batch of data and all the local data but not the similarity between the current batch of data and all the client data, which affects the feature differentiation ability of the model. To address this limitation, the FedPSFV algorithm expands the local classifier weight matrix of the client to include information about the data of other clients. However, instead of simply using the classifier weight matrices of other clients to expand the local classifier weight matrix, FedPSFV uses the prototypes of the other clients. The reasons and benefits of this treatment are as follows.

(1) In practice, the embedding vectors obtained from different categories of samples through the embedding function, not the weight vectors of the weight matrix, must be distinguished. The prototype is the average of the embedding vectors of all the samples of a category, so the prototype more effectively enhances the feature discrimination capability of the model.

(2) The weight vector of the classifier weight matrix, although it also considered the center of the category, is randomly initialized and updated by backpropagation several times at the end of each round of training, which is highly affected by the hyperparameters

of the optimizer, and the values are not sufficiently stable in comparison to those of the prototype.

Next, we discuss ways to expand the local classifier weight matrix with prototypes in the EdgeAdavein module. Assume that there are m clients and that the local classifier weight matrix of the k-th client is W_k, as shown in Equation (7). The prototype matrix of other clients a is P_a, as shown in Equation (8),

$$W_k = \begin{pmatrix} w_{1,1} & \cdots & w_{1,N_k} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,N_k} \end{pmatrix} \tag{7}$$

$$P_a = \begin{pmatrix} S_a^1 & \cdots & S_a^{N_a} \end{pmatrix} \tag{8}$$

where $w_{v,z}$ denotes the local classifier weight of client k, and $1 \leq v \leq d$, $1 \leq z \leq N_k$; S_a^j denotes the prototype of category j in client a sent by the server, which is a vector of d dimensions, and $1 \leq a \leq m$, $1 \leq j \leq N_a$; N_k and N_a denote the total number of categories for clients k and a, respectively; and d represents the number of output dimensions of the feature extraction layer MobileNetV3.

The expanded local classifier weight matrix W'_k for client k is shown in Equation (9).

$$W'_k = \begin{pmatrix} P_1 & P_2 & \cdots P_{k-1} & W_k & P_{k+1} \cdots P_{m-1} & P_m \end{pmatrix} \tag{9}$$

The similarity matrix of the FedPSFV algorithm is obtained by multiplying all the embedding vectors of a batch by the expanded classifier weight matrix.

The next section describes the computation of the image quality indicator in the EdgeAdavein module, which consists of two main parts: normalization of the edge intensity and a quality score computation. Assume that the set of edge intensities $B_k = \left\{ b_k^1, b_k^2, \ldots, b_k^i, \ldots b_k^{N_k} \right\}$ for the dataset of client k and that b_k^i is the edge intensity of category i. Then, for category i, its quality fraction Q^i is calculated as shown in Equations (10) and (11),

$$b'^i_k = \frac{b_k^i - min(B_k)}{max(B_k) - min(B_k)} \tag{10}$$

$$Q^i = \frac{100}{1 + e^{-u \cdot (b'^i_k - 0.5)}} \tag{11}$$

where b'^i_k denotes the normalized edge intensity; the *min* and *max* functions denote the minimum and maximum values, respectively, taken over the set; and u represents a parameter that controls the steepness of the function curve, which is typically 10. The above processing converts the edge intensities to a quality score ranging from 0 to 100. The quality fraction is subsequently normalized to $\widehat{Q^i}$, as shown in Equation (12), and it is used as an image quality indicator to calculate the margin loss.

$$\widehat{Q^i} = \left[\frac{Q^i - \mu_Q}{\sigma_Q} \cdot h \right]_{-1}^{1} \tag{12}$$

Here, μ_Q and σ_Q represent the mean and standard deviation, respectively, of all Q^i s within a batch; $[\cdot]_{-1}^{1}$ indicates that the value is intercepted in the range of -1 to 1 to increase numerical stability; and h is a hyperparameter that is typically 0.33.

3.3.3. Loss Calculation

The loss function is often used in deep learning to quantify the difference between the prediction result of the model and the true label value, and the setting of the loss function often determines the feature discrimination ability of the model. Traditional loss functions, such as the Softmax loss function, focus only on maximizing the probability of the correct class and do not sufficiently consider the distance within the class, resulting in insufficient feature discrimination. For this reason, this paper presents the margin-based loss function [35–37], which not only reduces the intraclass distance of the local dataset but also increases the interclass distance from other client datasets in combination with the shared prototype.

The margin-based loss function $f(\theta_j, m)_{Edge}$ designed in this paper is shown in Equation (13),

$$f(\theta_j, m)_{Edge} = \begin{cases} s(cos(\theta_j + g_{angle}) - g_{add}) & j = y_i \\ scos\theta_j & j \neq y_i \end{cases} \tag{13}$$

where s represents the hyperparameter scaling factor, θ_j denotes the value of Row i and Column j of the similarity matrix, representing the angle between the embedding vector of category i and the column vector of category j in the expanded classifier weight matrix; y_i denotes the labeled value of category i; and g_{angle} and g_{add} denote the additive margins in angular space and cosine space, respectively, with the formulas shown in (14) and (15),

$$g_{\text{angle}} = -m \cdot \widehat{Q^i} \tag{14}$$

$$g_{\text{add}} = m \cdot \widehat{Q^i} + m \tag{15}$$

where m represents the margin, a type of hyperparameter, which is typically 0.4.

The pseudocode that calculates edge intensity and margin loss is shown in Algorithm 1.

Algorithm 1. Calculating edge intensity and margin loss

Parameters: dataset of client D_k, sample set $d_k^a \in D_k$ belonging to a class of a, sample image I belongs to a sample set, edge intensity set B_k of the client, and quality score set Q_k of the client.

Every client executes:

Initialize the local classifier weight matrix W_k and hyperparameter margin m
for $d_k^i = d_k^1, d_k^2, \ldots\ldots d_k^{N_k}$ **do**
  set the set of edge intensities b^i
  for $I \in d_k^i$ **do**
    Compute I_g by Equation (3) via I
    Compute the enhanced image I' via Equation (2) using I_g and I
    set $I \leftarrow I'$
    Compute the edge intensity b via Equation (5) using I_g. Add it to b^i
  end for
  Compute $\overline{b^i}$, the mean of b^i. Add it to B_k
end for
Normalize the mean value of edge intensity $\overline{b^i}$ for each category in B_k according to Equation (10) to b'^i_k
Compute the quality score Q^i for each category by Equation (11) using b'^i_k. Add it to Q_k
Compute θ using D_k, W_k, the embedding vectors during training, and other client prototypes shared by the FedPSFV algorithm
Compute the margin loss by Equations (12) and (13) using Q_k and θ

On this basis, the loss function designed in this paper is shown in Equation (16), where N represents the total number of categories in all the client datasets, i.e., the total number of uploaded prototypes; n denotes the total number of categories of local clients; θ_j indicates the angle between the feature vector of category i and the j-th local classifier weight vector; φ_J represents the angle between feature vector i and the J-th category center vector of the prototype matrix of all the other clients; y_i denotes the true label index; m indicates the margin; and f represents the margin-based loss function mentioned previously.

$$\mathcal{L}_{FedPSFV} = -\log \frac{e^{f(\theta_{y_i},m)_{Edge}}}{e^{f(\theta_{y_i},m)_{Edge}} + \sum_{j \neq y_i}^{n} e^{scos\,\theta_j} + \sum_{J \neq y_i}^{N-n} e^{scos\,\varphi_J}} \tag{16}$$

The loss function of the FedPSFV algorithm has not only the comparison term $e^{scos\,\theta_j}$, with the local dataset in the denominator, but also the comparison term $e^{scos\,\varphi_J}$, with the prototypes of the datasets of the other clients. This feature motivates the model to differentiate between the local embedding vectors and the prototypes of the other clients to increase the feature differentiation ability of the model and compensate for problems such as the slow convergence speed of the individual clients due to the small dataset.

The pseudocode of the client training process and the FedPSFV loss function are shown in Algorithm 2.

Algorithm 2. Client training process and FedPSFV loss function

Parameters: client serial number k, model parameter ω after the t-th round aggregation, and the set of global prototypes $\overline{S}$ after the t-th round aggregation
ClientUpdate (k, ω_t, $\overline{S}$):
set $\omega_{t,k} \leftarrow \omega_t$
Compute W_k by Equation (9) via $\overline{S}$ and initialize hyperparameter margins m
for each local epoch **do**
for batch $(x_k, y_k) \in D_k$ **do**
    Compute loss via Equations (13) and (16) using x_k, $\omega_{t,k}$, W_k, and y_k
    Update local model parameter $\omega_{t+1,k}$ according to the loss.
  end for
end for
Compute local prototype S_k via Equation (6) using x_k and $\omega_{t+1,k}$
return $\omega_{t+1,k}$, S_k

3.4. Global Aggregation

This section introduces the server-side aggregation process, which involves the aggregation of model parameters and the aggregation of prototypes. The server sends the aggregated model parameters and prototype matrix to each client. The entire process is described below.

At the end of each round of communication, the server side receives the updated model parameters of the feature extraction layer from the clients, as well as the prototypes of all the categories of each client. Assuming that the model height of the k-th client is represented as ω_k, the server side first aggregates the parameters of the model of the client, as shown in Equation (17),

$$\omega = \sum_{k=1}^{m} \frac{|D_k|}{N} \omega_k \tag{17}$$

where ω denotes the aggregated global model weights and D_k represents the dataset owned by client k.

The server then aggregates prototypes from different clients, as shown in Equation (18),

$$\overline{S}^{(j)} = \frac{1}{|CS_j|} \sum_{k \in CS_j} \frac{\left|D_k^j\right|}{N_j} S_k^{(j)} \tag{18}$$

where $\left|D_k^j\right|$ represents the number of training samples belonging to category j in client k, N_j denotes the total number of samples of category j in the dataset of all the clients, CS_j denotes the set of clients containing category j, and $S_k^{(j)}$ denotes the prototype of category j uploaded by client k. Instead of aggregating multiple prototypes of the same category into a single one, FedPSFV computes the aggregated vector after aggregation and then updates the prototype matrix for each client that owns the category to ensure that the total number of prototypes before and after aggregation remains the same. This process ensures that the shape of the updated weight matrix remains consistent.

The pseudocode for the server-side aggregation algorithm is shown in Algorithm 3.

Algorithm 3. Global aggregation

Parameters: The global prototype set $\{\overline{S}\}$ excludes the prototypes $\overline{S_{N-i}}$ owned by client i.

The server executes:

Initialize the feature representation layer parameters ω of the global model and global prototype set $\{\overline{S}\}$
for each round T = 1,2. . .. . .**do**
  for each client k in parallel **do**
    $\omega_{t+1,k}$, $S_k \leftarrow$ **ClientUpdate** (k, ω, $\overline{S_{N-k}}$)
  end for
  Update parameters ω of the global model via Equation (17) using $\omega_{t+1,k}$
  Update the global prototype set $\{\overline{S}\}$ via Equation (18) using S_k
end for

4. Experiments

4.1. Datasets and Evaluation Methods

In this work, experiments are performed on six public datasets of finger vein images, including SDUMLA-HMT [38], MMCBNU-6000 [39], FV-USM [40], UTFVP [41], VERA [42], and NUPT-FV [43]. The SDUMLA-HMT dataset contains 636 categories, the MMCBNU-6000 dataset contains 600 categories, the FV-USM dataset contains 492 categories, the UTFVP dataset contains 360 categories, the VERA dataset contains 220 categories, and the NUPT-FV dataset contains 1680 categories. Each of the six datasets above contains 6, 10, 6, 4, 2, and 10 images per category, respectively.

In this work, there are six clients, each of which is assigned a public dataset as described above to simulate the process of different organizations collaborating on federated learning training. The background, illumination, and vein clarity of each public dataset vary significantly due to the sampling method. For example, the NUPT-FV dataset was sampled with fingers facing vertically, whereas the other datasets were horizontally oriented. The total number of samples in the VERA dataset was only 440, whereas the total number of samples in the NUPT-FV dataset was 16,800. These differences indicate that the six datasets have different data distributions, which exemplifies the classic non-IID problem in FL [44].

To better evaluate the generalizability of the models trained by the algorithm, this paper uses the open-set testing method from the field of biometrics. We divide the dataset into training and testing sets at a ratio of 8:2 by the number of categories. When federated

learning training is performed, only 80% of the categories in each client dataset are involved in training, and the remaining 20% of the categories are not visible to the model. When the performance of the model is tested, all the samples in 20% of the categories are output into embedding vectors via the feature representation layer of the model. Validation pairs are generated via permutations and combinations, and whether they belong to the same category is determined by evaluating whether the similarity of the embedding vectors of the validation pairs is greater than a certain threshold.

In this experiment, the equal error rate (EER) and TAR@FAR = 0.01 were used as evaluation metrics. The equal error rate is the value when the false acceptance rate (FAR) is equal to the false rejection rate (FRR), whereas TAR@FAR = 0.01 is the value of the true acceptance rate (TAR) when the false acceptance rate (FAR) is 0.01. The relevant calculation formulas are shown in (19)–(21),

$$FAR = \frac{N_{FA}}{N_{IRA}} \times 100\% \tag{19}$$

$$FRR = \frac{N_{FR}}{N_{GRA}} \times 100\% \tag{20}$$

$$TAR = 1 - FRR \tag{21}$$

where N_{IRA} represents the number of all interclass match pairs, N_{FA} denotes the number of incorrect acceptances in all interclass match pairs, N_{GRA} indicates the number of all intraclass match pairs, and N_{FR} represents the number of incorrect rejections in all intraclass match pairs.

4.2. Experimental Results and Analysis

To verify the effectiveness of the FedPSFV algorithm, we conducted four sets of experiments. In the first set of experiments, we compare the FedPSFV algorithm with two cases of client-independent training and client-centralized training to verify the effectiveness of the federated learning framework. In the second set of experiments, the effectiveness of the prototype sharing method in the FedPSFV algorithm and the EdgeAdavein module is verified via ablation experiments. In addition, we compared the performance of the FedPSFV algorithm using different backbone encoders, as well as its robustness under low image contrast conditions. The third set of experiments is compared with existing methods in the field of federated learning and finger vein recognition to verify the superiority of the method in this paper. The fourth set of experiments discusses the communication cost and inference efficiency of the FedPSFV algorithm.

4.2.1. Comparison with Client-Independent Training and Client-Centralized Training Methods

In this experiment, "Local" means that each client is trained independently and that each client does not communicate with the others during training. "Centralized" means that the six datasets are trained together centrally. These two tests simulate the situation under nonfederated learning, and both use the MobileNetV3 model.

The experimental results are shown in Table 1. The table shows that the centralized training method performs better than the local training method; however, the FedPSFV algorithm outperforms both methods. First, the centralized training method does not perform processing, such as image enhancement, and reduces the intraclass distance. Second, FedPSFV increases the interclass distances of different classes in the six datasets by sharing the prototype and the margin-based loss function while decreasing the intraclass distances, thereby increasing the feature extraction ability of the model. In addition, the FedPSFV algorithm is significantly better on VERA and UTFVP, two datasets with small

data volumes. Compared with the local and centralized methods, the EER is reduced by 89.15% and 82.82%, respectively, and TAR@FAR = 0.01 is increased by 14.90% and 5.31%, respectively, which indicates that the proposed method can also train a model with high recognition accuracy for clients with small data volumes.

Table 1. Comparison of local training, centralized training, and FedPSFV.

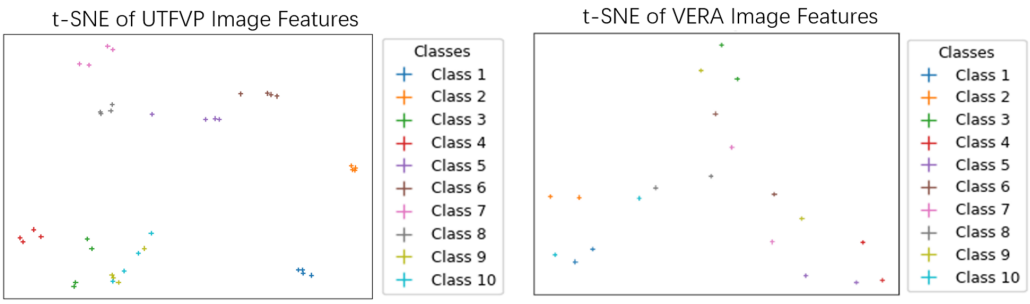
Datasets	Local		Centralized		FedPSFV	
	EER	**TAR@FAR = 0.01**	**EER**	**TAR@FAR = 0.01**	**EER**	**TAR@FAR = 0.01**
SDUMLA	5.15%	89.35%	3.52%	91.84%	0.65%	99.79%
MMCBNU	0.80%	99.31%	0.93%	99.15%	0.08%	100.00%
USM	5.21%	81.31%	4.64%	95.85%	0.40%	99.69%
UTFVP	9.57%	80.71%	4.77%	91.39%	0.53%	99.75%
VERA	6.04%	70.00%	2.60%	90.12%	1.06%	99.29%
NUPT	0.86%	99.26%	1.01%	98.97%	0.26%	98.92%
Average	4.61%	86.66%	2.91%	94.55%	0.50%	99.57%

To further compare the performance of the three methods on small datasets, we also use the t-distributed stochastic neighbor embedding (t-SNE) [45] method to downscale the d-dimensional embedding vectors of different instances into 2-dimensional vectors to determine whether the model can indeed increase the interclass distance and decrease the intraclass distance. We randomly selected samples from 10 categories of the UTFVP dataset and the VERA dataset to compute the t-SNE images of their embedding vectors. The results are shown in Figure 4. For the local method, after the extracted embedding vectors are downscaled, the interclass distance of many categories is small, indicating that the model does not distinguish between different categories of samples very well. For the centralized method, the interclass distance is better than that of the Local method because it uses only one model with a larger amount of training data, but the intraclass distance is also larger. For the FedPSFV algorithm, the embedding vectors extracted by its model are able to not only increase the interclass distance well but also reduce the intraclass spacing. These findings prove that FedPSFV is more capable of extracting image features and can increase the distinguishability of sample data.

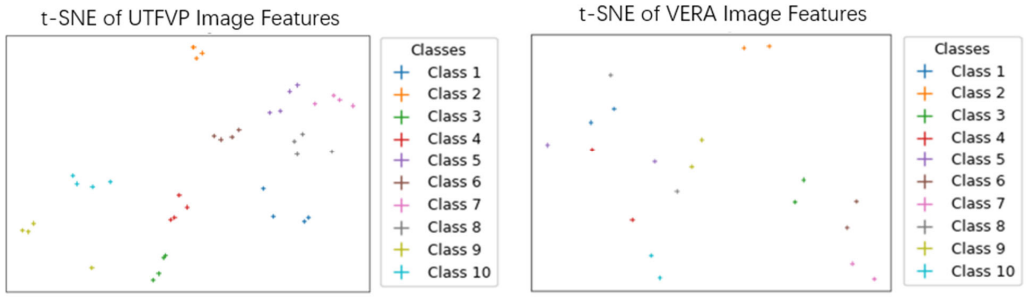
In addition, we visualize the t-SNE plots of embedding vectors from all six client datasets at different communication rounds to verify the effectiveness of FedPSFV for increasing global class separability. Specifically, we select three classes from each client dataset and extract embedding vectors by feeding all 18 classes of data samples into the encoder at communication rounds 0, 5, 10, and 15. The resulting t-SNE visualizations are shown in Figure 5. At the beginning of training, data samples from different clients are combined. However, as training progresses, the interclass distances between data samples from different clients gradually increase.

4.2.2. Ablation Study

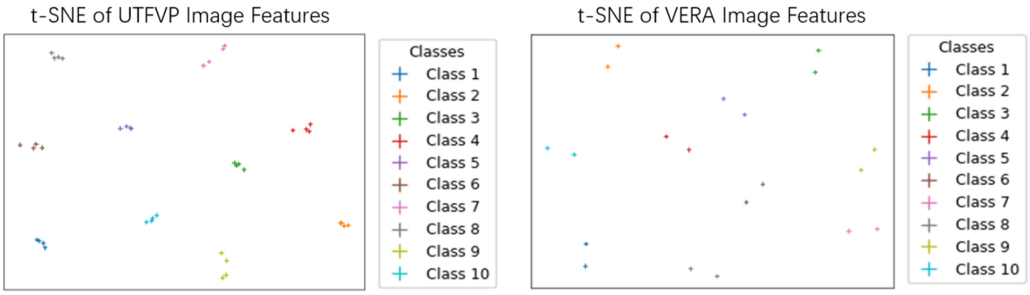
In this experiment, first, we conducted ablation studies on the FedPSFV algorithm. Second, we compared the performance of the FedPSFV algorithm when three different backbone encoders were used. Last, we evaluated the robustness of the FedPSFV algorithm under image contrast levels of 0.6, 0.8, and 1.0.



(**a**) Local



(**b**) Centralized



(**c**) FedPSFV

Figure 4. t-SNE images of embedding vectors for the UTFVP and VERA datasets with different training methods.

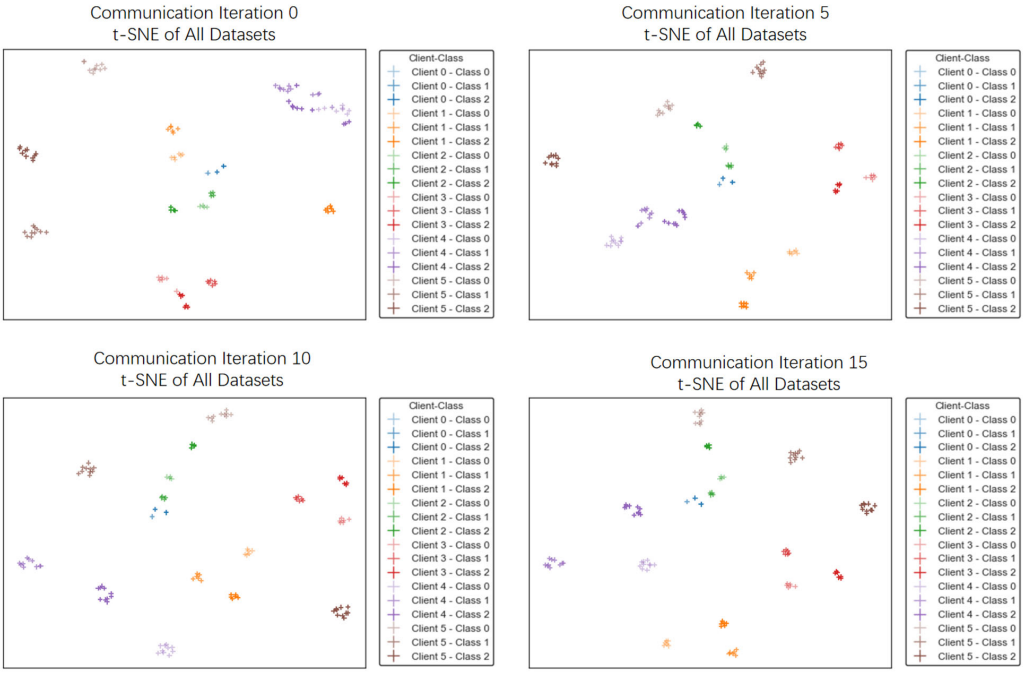


Figure 5. t-SNE image of client embedding vector evolution over communication rounds.

First, the submethod is ablated to verify the effectiveness of the proposed submethod. The FedPSFV algorithm consists of two main submethods: (1) DPS, a method for expanding the classifier weight matrix by sharing prototypes; and (2) D_{AA}, a method for improving both the image and the image quality indicator of the margin-based loss function. The selection or nonselection of submethods involves the following four methods.

(1) W/O D_{PS} and D_{AA} (i.e., baseline method): In the federated learning scenario, MobileNetV3 is used as the basic model, and the margin-based loss function before improvement [37] is used.

(2) W/O D_{AA}: Based on the baseline method, the classifier weight matrix is expanded by sharing their respective prototypes among different clients and using the aggregated prototypes.

(3) W/O D_{PS}: Based on the baseline method, image enhancement is used, and the image quality indicator is improved.

(4) FedPSFV: Based on the baseline method, the classifier weight matrix is expanded by sharing the prototypes. The image enhancement and improved image quality indicators are also used.

The experimental results are shown in Table 2. The results of the W/O D_{PS} and D_{AA} method is better than those of both the local training method and the centralized training method in Experiment 1, with average EER and TAR@FAR = 0.01 of 1.25% and 97.86%, respectively, which is attributed to the use of the margin-based loss function in the W/O D_{PS} and D_{AA} methods. The margin-based loss function can narrow the intraclass distance of the category samples and thereby increase the feature differentiation ability of the model. Compared with the W/O D_{PS} and D_{AA} methods, the W/O D_{AA} and W/O D_{PS} methods reduce the average EER by 21.60% and 28.80%, respectively, and increase the average TAR@FAR = 0.01 by 1.55% and 0.67%, respectively. This finding indicates that both submethods can increase the feature extraction capability of the model and that the W/O D_{AA} method provides a greater performance improvement. The FedPSFV algorithm with both submethods achieves the optimal overall performance.

Table 2. Ablation study.

Datasets	W/O D_{PS} and D_{AA}		W/O D_{AA}		W/O D_{PS}		FedPSFV	
	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01
SDUMLA	2.71%	95.37%	1.92%	99.25%	2.23%	95.98%	0.65%	99.79%
MMCBNU	0.08%	100.00%	0.39%	99.89%	0.72%	98.94%	0.08%	100.00%
USM	2.64%	95.42%	0.62%	99.45%	0.63%	99.70%	0.40%	99.69%
UTFVP	0.36%	99.98%	0.50%	99.82%	0.29%	100.00%	0.53%	99.75%
VERA	1.36%	98.29%	0.82%	100.00%	1.08%	96.67%	1.06%	99.29%
NUPT	0.33%	98.08%	1.62%	97.85%	0.38%	99.84%	0.26%	98.92%
Average	1.25%	97.86%	0.98%	99.38%	0.89%	98.52%	0.50%	99.57%

Second, the effect of different encoders on the performance of the FedPSFV algorithm is explored. The main reason for choosing MobileNetV3-Large as the encoder in the FedPSFV framework is its lightweight design, with approximately 3.00 M parameters (excluding the classification layer), making it highly suitable for deployment on mobile devices in finger vein recognition scenarios. To further investigate the effect of the encoder on the performance of FedPSFV, we additionally compare it with two more recent encoders: EfficientNetV2-S [46] and ConvNeXt-Tiny [47]. EfficientNetV2-S contains approximately 24 M parameters, whereas ConvNeXt-Tiny has approximately 29 M parameters. The experimental results are shown in Table 3. The average EERs achieved using EfficientNetV2-S and ConvNeXt-Tiny are 4.19% and 1.69%, respectively. This finding shows that the FedPSFV algorithm has good adaptability to different encoders.

Table 3. Performance comparison of FedPSFV via different encoders.

Datasets	FedPSFV with Efficientnetv2-S		FedPSFV with ConvNeXt-Tiny		FedPSFV with MobileNetV3 Large	
	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01
SDUMLA	3.60%	91.69%	1.34%	98.61%	0.65%	99.79%
MMCBNU	1.35%	98.20%	0.84%	99.49%	0.08%	100.00%
USM	0.47%	99.83%	0.31%	99.90%	0.40%	99.69%
UTFVP	5.38%	84.32%	0.48%	99.60%	0.53%	99.75%
VERA	13.57%	60.00%	6.62%	85.52%	1.06%	99.29%
NUPT	0.79%	99.29%	0.54%	99.45%	0.26%	98.92%
Average	4.19%	88.89%	1.69%	97.10%	0.50%	99.57%

Last, the robustness of the FedPSFV algorithm under low image contrast conditions is discussed. FedPSFV uses gradient-enhanced images, with edge strength serving as an image quality indicator within the classification layer. However, under low-contrast conditions, the gradient-based method may fail to accurately segment vein patterns. To investigate this issue, we manually decreased the image contrast and evaluated the performance of FedPSFV at contrast levels of 0.6, 0.8, and 1.0. As shown in Table 4, lower contrast levels indeed result in performance degradation. For example, on the NUPT dataset, the EER increases from 0.26% to 3.14% and 5.45% at contrast levels of 0.6 and 0.8, respectively, whereas TAR@FAR = 0.01 decreases from 98.92% to 98.84% and 95.12%, respectively. Similar trends can be observed for the VERA dataset. On average, the EER increases from 1.06% to 2.45% and 1.85%, and TAR@FAR = 0.01 decreases from 99.29% to 97.43% and 98.54%, respectively. These results reveal that the FedPSFV algorithm remains robust under low image contrast conditions.

Table 4. Performance of FedPSFV at different image contrast levels.

Datasets	Contrast 0.6		Contrast 0.8		Contrast 1.0	
	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01
SDUMLA	0.56%	99.79%	0.92%	99.18%	0.65%	99.79%
MMCBNU	0.04%	99.98%	0.28%	99.95%	0.08%	100.00%
USM	0.40%	99.87%	0.44%	99.87%	0.40%	99.69%
UTFVP	0.80%	99.60%	0.21%	100.0%	0.53%	99.75%
VERA	2.45%	97.43%	1.85%	98.54%	1.06%	99.29%
NUPT	3.14%	98.84%	5.45%	95.12%	0.26%	98.92%
Average	1.23%	99.25%	1.53%	98.78%	0.50%	99.57%

Figure 6 shows the training loss curve of the FedPSFV algorithm, which contains the training losses of the six clients involved in training. The training loss of the six clients decreases rapidly in the first 15 rounds and then stabilizes gradually, which shows that the FedPSFV algorithm has good generalizability and converges smoothly on six different datasets.

4.2.3. Comparison with Existing Methods

In this experiment, we compare the FedPSFV algorithm with pfedsim [29], FedFV [18], MOON [30], DDP-FV [19], Ditto [10], and FedAS [48]. The results are shown in Table 5, and the FedPSFV algorithm outperforms the comparison methods on the vast majority of datasets, with an average EER of 0.50%, which is 82.82%, 81.48%, 82.27%, 81.95%, 90.22% and 83.97% lower than that of pfedsim, MOON, FedFV, DDP-FV, Ditto, and FedAS, respectively. The average value of TAR@FAR = 0.01 is 99.57%, which is 8.81%, 7.76%, 8.89%, 5.00%, 11.25% and 8.64% higher than those of pfedsim, MOON, FedFV, DDP-FV,

Ditto, and FedAS, respectively. For the small dataset UTFVP, the EER of FedPSFV is 75.69–94.54% lower than those of the other methods, and TAR@FAR = 0.01 is 4.58–29.80% higher than those of the other methods. For the small dataset VERA, the EER of FedPSFV is 80.19–88.10% lower than those of the other methods, and TAR@FAR = 0.01 is 10.38–54.80% higher than those of the other methods. These findings prove that FedPSFV can effectively solve the data heterogeneity problem in federated learning because of the difference in data volume and better performance.



Figure 6. Training loss curves of six clients in the FedPSFV algorithm. All clients are able to converge rapidly.

Table 5. Performance of FedPSFV and the comparison methods.

Datasets	pfedsim		MOON		FedFV		DDP-FV		Ditto		FedAS		FedPSFV	
	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01	EER	TAR@FAR = 0.01
SDUMLA	2.20%	96.33%	3.31%	93.05%	2.64%	94.93%	2.52%	96.23%	6.21%	90.19%	2.92%	95.34%	0.65%	99.79%
MMCBNU	0.45%	99.78%	0.79%	99.42%	0.95%	99.04%	0.61%	99.63%	0.94%	99.12%	1.08%	98.82%	0.08%	100.00%
USM	2.65%	94.64%	1.71%	96.77%	2.19%	95.17%	3.24%	90.26%	4.36%	88.96%	3.62%	89.04%	0.40%	99.69%
UTFVP	2.18%	95.38%	2.92%	92.36%	3.20%	91.27%	2.90%	93.81%	9.71%	76.85%	4.81%	86.61%	0.53%	99.75%
VERA	8.91%	64.14%	6.61%	73.60%	7.04%	69.05%	6.40%	89.95%	8.33%	83.12%	5.35%	80.90%	1.06%	99.29%
NUPT	1.09%	98.79%	0.83%	99.17%	0.89%	99.19%	0.92%	99.12%	1.13%	98.75%	0.92%	99.19%	0.26%	98.92%
Average	2.91%	91.51%	2.70%	92.40%	2.82%	91.44%	2.77%	94.83%	5.11%	89.50%	3.12%	91.65%	0.50%	99.57%

4.2.4. Communication Cost and Inference Efficiency

Our FedPSFV achieves superior performance while maintaining manageable communication overhead, as shown in Table 6. The primary contributors to the communication cost are the encoder parameters of MobileNetV3-Large, which account for approximately 3.00 M. Compared with other methods that transmit only model parameters [49,50], FedPSFV introduces additional communication overhead due to prototype sharing. For a prototype dimension of 512, the average upload cost per client increases by only 0.26 M, whereas the download cost increases by 1.56 M, which is acceptable in real-world scenarios where the uplink bandwidth is typically lower than the downlink bandwidth.

Table 6. Per-round upload and download overhead using the MobileNetV3 encoder.

	Upload	**Download**
pfedsim, MOON, FedFV, DDP-FV, Ditto, FedAS	3.00 M	3.00 M
FedPSFV	3.00 M + 0.26 M	3.00 M + 1.56 M

Regarding inference time, the primary components of latency in real-time finger vein recognition systems are encoding time and matching time, that is, the time required to process a finger vein image through the encoder to obtain an embedding vector and the time to match this vector against all stored templates in the database. On our mobile-class device, the NVIDIA RTX 4060 laptop (Santa Clara, CA, USA), we evaluated these two components by testing individual samples obtained from all datasets. The encoding time relies on the encoder used; for example, when MobileNetV3-Large is used, the average encoding time is approximately 3.655 ms. The matching time is affected by the number of enrolled identities; for approximately 3980 categories across the six datasets, the average matching time is approximately 1.001 ms. Therefore, the total average inference time is approximately 4.656 ms, which meets the real-time performance requirements of practical finger vein recognition systems.

5. Conclusions

In this paper, the finger vein recognition problem in federated learning scenarios is investigated, and the FedPSFV algorithm is designed and implemented. On the basis of the federated learning framework, prototype sharing and margin-based loss functions are presented, which increase the feature discrimination ability of the model and address the performance fluctuation problem caused by differences in datasets. Meanwhile, FedPSFV retains the advantages of a lightweight model and communication efficiency. Comparative experiments show that the FedPSFV algorithm is superior to the existing methods and that the trained model also performs well on clients with small data volumes. For limitations, this study focused exclusively on the application of FedPSFV in federated finger vein recognition. However, FedPSFV is not restricted to this specific domain. The method of improving the loss function by sharing prototypes highlighted in this paper can also be used in general federated learning scenarios, providing a general and efficient solution for model training in heterogeneous data scenarios. Future work will explore the generalizability and effectiveness of FedPSFV in broader federated learning contexts.

Author Contributions: Conceptualization, H.X. and J.G.; methodology, H.X.; software, H.X.; validation, H.X. and Y.G.; formal analysis, H.X.; investigation, H.X., Y.G. and Y.Q.; resources, J.G.; data curation, H.X., Y.G. and Y.Q.; writing—original draft preparation, H.X.; writing—review and editing, J.G.; visualization, H.X.; supervision, J.G. and H.R.; project administration, J.G. and H.R.; and funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (grant no. 62272242) and the Innovation and Entrepreneurship Training Program for College Students (grant no. CXXYB2024147).

Institutional Review Board Statement: Ethical review and approval were waived for this study because the study was conducted with publicly available images in finger vein datasets and did not involve experimental studies with humans.

Informed Consent Statement: Informed consent was obtained from all the subjects involved in the study.

Data Availability Statement: The data presented in this study are publicly available in the SDUMLA-HMT [38], MMCBNU-6000 [39], FV-USM [40], UTFVP [41], VERA [42], and NUPT-FV [43] datasets.

Acknowledgments: The authors thank everyone who participated in the testing process.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

References

1. Zhang, Z.; Liu, P.; Su, C.; Tong, S. A High-Speed Finger Vein Recognition Network with Multi-Scale Convolutional Attention. *Appl. Sci.* **2025**, *15*, 2698. [CrossRef]
2. Ren, H.; Sun, L.; Fan, X.; Cao, Y.; Ye, Q. IIS-FVIQA: Finger Vein Image Quality Assessment with intra-class and inter-class similarity. *Pattern Recognit.* **2025**, *158*, 111056. [CrossRef]
3. Bhushan, K.; Singh, S.; Kumar, K.; Kumar, P. Deep learning based automated vein recognition using swin transformer and super graph glue model. *Knowl.-Based Syst.* **2025**, *310*, 112929. [CrossRef]
4. Zhang, Z.; Zhou, Z.; Tian, Z.; Yu, S. CRCGAN: Toward robust feature extraction in finger vein recognition. *Pattern Recognit.* **2025**, *158*, 111064. [CrossRef]
5. Zhang, J.; Lu, Z.; Li, M.; Wu, H. GAN-based image augmentation for finger-vein biometric recognition. *IEEE Access* **2019**, *7*, 183118–183132. [CrossRef]
6. Stallings, W. Handling of personal information and deidentified, aggregated, and pseudonymized information under the California consumer privacy act. *IEEE Secur. Priv.* **2020**, *18*, 61–64. [CrossRef]
7. Goldman, E. An Introduction to the California Consumer Privacy Act (CCPA). Santa Clara University Legal Studies Research Paper 2020. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3211013 (accessed on 11 August 2025).
8. Bygrave, L.A. *Article 25 Data Protection by Design and by Default//The EU General Data Protection Regulation (GDPR)*; Oxford University Press: Oxford, UK, 2020.
9. Shamsian, A.; Navon, A.; Fetaya, E.; Chechik, G. Personalized federated learning using hypernetworks. In Proceedings of the 38th International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 9489–9502.
10. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and robust federated learning through personalization. In Proceedings of the 38th International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 6357–6368.
11. Husnoo, M.A.; Anwar, A.; Hosseinzadeh, N.; Islam, S.N.; Mahmood, A.N.; Doss, R. FedREP: Towards horizontal federated load forecasting for retail energy providers. In Proceedings of the 2022 IEEE PES 14th Asia-Pacific Power and Energy Engineering Conference (APPEEC), Melbourne, Australia, 20–23 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
12. Liu, J.; Ma, H. Finger Vein Recognition based on Self-Attention and Convolution. In Proceedings of the 2024 36th Chinese Control and Decision Conference (CCDC), Xi'an, China, 25–27 May 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 5750–5754.
13. Ghafourian, M.; Fierrez, J.; Vera-Rodriguez, R.; Tolosana, R.; Morales, A. SaFL: Sybil-aware federated learning with application to face recognition. In Proceedings of the 2024 IEEE International Conference on Image Processing Challenges and Workshops (ICIPCW), Abu Dhabi, United Arab Emirates, 27–30 October 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 4050–4056.
14. Gupta, H.; Rajput, T.K.; Vyas, R.; Vyas, O.P.; Puliafito, A. Biometric iris identifier recognition with privacy preserving phenomenon: A federated learning approach. In Proceedings of the International Conference on Neural Information Processing, Virtual, 22–26 November 2022; Springer Nature: Singapore, 2022; pp. 493–504.
15. Luo, Z.; Wang, Y.; Wang, Z.; Wang, Z.; Sun, Z.; Tan, T. Fediris: Towards more accurate and privacy-preserving iris recognition via federated template communication. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 3357–3366.
16. Meng, Q.; Zhou, F.; Ren, H.; Feng, T.; Liu, G.; Lin, Y. Improving federated learning face recognition via privacy-agnostic clusters. *arXiv* **2022**, arXiv:2201.12467.
17. Guo, J.; Mu, H.; Liu, X.; Ren, H.; Han, C. Federated learning for biometric recognition: A survey. *Artif. Intell. Rev.* **2024**, *57*, 208. [CrossRef]
18. Lian, F.Z.; Huang, J.D.; Liu, J.X.; Chen, G.; Zhao, J.H.; Kang, W.X. FedFV: A personalized federated learning framework for finger vein authentication. *Mach. Intell. Res.* **2023**, *20*, 683–696. [CrossRef]
19. Guo, Z.; Guo, J.; Huang, Y.; Zhang, Y.; Ren, H. DDP-FedFV: A Dual-Decoupling Personalized Federated Learning Framework for Finger Vein Recognition. *Sensors* **2024**, *24*, 4779. [CrossRef]
20. Ren, H.; Sun, L.; Cao, Y.; Qi, L.; Ye, Q. FVRFL: Federated Learning for Finger Vein Recognition under Distributed Scenario. *IEEE Trans. Instrum. Meas.* **2025**, *74*, 5017312. [CrossRef]
21. Miura, N.; Nagasaka, A.; Miyatake, T. Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification. *Mach. Vis. Appl.* **2004**, *15*, 194–203. [CrossRef]
22. Miura, N.; Nagasaka, A.; Miyatake, T. Extraction of finger-vein patterns using maximum curvature points in image profiles. *IEICE Trans. Inf. Syst.* **2007**, *90*, 1185–1194. [CrossRef]
23. Lee, E.C.; Lee, H.C.; Park, K.R. Finger vein recognition using minutia-based alignment and local binary pattern-based feature extraction. *Int. J. Imaging Syst. Technol.* **2009**, *19*, 179–186. [CrossRef]
24. Hu, H.; Kang, W.; Lu, Y.; Liu, H.; Zhao, J.; Deng, F. FV-Net: Learning a finger-vein feature representation based on a CNN. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 3489–3494.

25. Zhao, D.; Ma, H.; Yang, Z.; Li, J.; Tian, W. Finger vein recognition based on lightweight CNN combining center loss and dynamic regularization. *Infrared Phys. Technol.* **2020**, *105*, 103221. [CrossRef]
26. Song, J.M.; Kim, W.; Park, K.R. Finger-vein recognition based on deep DenseNet using composite image. *Ieee Access* **2019**, *7*, 66845–66863. [CrossRef]
27. Wang, S.; Gai, K.; Yu, J.; Zhang, Z.; Zhu, L. PraVFed: Practical heterogeneous vertical federated learning via representation learning. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 2693–2705. [CrossRef]
28. Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; Zhang, C. Fedproto: Federated prototype learning across heterogeneous clients. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022; Volume 36, pp. 8432–8440.
29. Tan, J.; Zhou, Y.; Liu, G.; Wang, J.H.; Yu, S. pFedSim: Similarity-Aware Model Aggregation Towards Personalized Federated Learning. *arXiv* **2023**, arXiv:2305.15706.
30. Li, Q.; He, B.; Song, D. Model-contrastive federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 10713–10722.
31. Zhang, J.; Liu, Y.; Hua, Y.; Cao, J. An Upload-Efficient Scheme for Transferring Knowledge from a Server-Side Pre-trained Generator to Clients in Heterogeneous Federated Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 12109–12119.
32. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
33. Zhang, Z.; Bai, D.; Fan, W.; Shi, X.; Tan, H.; Du, J.; Bai, X.; Zhou, F. A Time–Frequency-Aware Hierarchical Feature Optimization Method for SAR Jamming Recognition. *IEEE Trans. Aerosp. Electron. Syst.* **2025**, *61*, 10619–10635. [CrossRef]
34. Seddik, M.E.A.; Tamaazousti, M. Neural networks classify through the class-wise means of their representations. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022; Volume 36, pp. 8204–8211.
35. Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; Liu, W. Cosface: Large margin cosine loss for deep face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5265–5274.
36. Deng, J.; Guo, J.; Xue, N.; Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 4690–4699.
37. Kim, M.; Jain, A.K.; Liu, X. Adaface: Quality adaptive margin for face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 18750–18759.
38. Yin, Y.; Liu, L.; Sun, X. SDUMLA-HMT: A multimodal biometric database. In *Biometric Recognition, Proceedings of the 6th Chinese Conference, CCBR 2011, Beijing, China, 3–4 December 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 260–268.
39. Lu, Y.; Xie, S.J.; Yoon, S.; Wang, Z.; Park, D.S. An available database for the research of finger vein recognition. In Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP), Hangzhou, China, 16–18 December 2013; IEEE: Piscataway, NJ, USA, 2013; Volume 1, pp. 410–415.
40. Asaari, M.S.M.; Suandi, S.A.; Rosdi, B.A. Fusion of band limited phase only correlation and width centroid contour distance for finger based biometrics. *Expert Syst. Appl.* **2014**, *41*, 3367–3382.
41. Ton, B.T.; Veldhuis, R.N.J. A high quality finger vascular pattern dataset collected using a custom designed capturing device. In Proceedings of the 2013 International Conference on Biometrics (ICB), Madrid, Spain, 4–7 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–5.
42. Tome, P.; Vanoni, M.; Marcel, S. On the vulnerability of finger vein recognition to spoofing. In Proceedings of the 2014 International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany, 10–12 September 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–10.
43. Ren, H.; Sun, L.; Guo, J.; Han, C. A dataset and benchmark for multimodal biometric recognition based on fingerprint and finger vein. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2030–2043. [CrossRef]
44. Lu, Z.; Pan, H.; Dai, Y.; Si, X.; Zhang, Y. Federated learning with non-iid data: A survey. *IEEE Internet Things J.* **2024**, *11*, 19188–19209. [CrossRef]
45. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
46. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the 38th International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 10096–10106.
47. Liu, Z.; Mao, H.; Wu, C.Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A convnet for the 2020s. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 11976–11986.
48. Yang, X.; Huang, W.; Ye, M. FedAS: Bridging inconsistency in personalized federated learning. In Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 11986–11995.

49. Ren, H.; Sun, L.; Ren, J.; Cao, Y. FV-DDC: A novel finger-vein recognition model with deformation detection and correction. *Biomed. Signal Process. Control* **2025**, *100*, 107098. [CrossRef]
50. Mu, H.; Guo, J.; Han, C.; Sun, L. Pafedfv: Personalized and asynchronous federated learning for finger vein recognition. *arXiv* **2024**, arXiv:2404.13237. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.