Multimedia Systems
https://doi.org/10.1007/s00530-022-00927-5

RESEARCH ARTICLE

FedFV: federated face verification via equivalent class embeddings

Lingyun Liu[1,2,3] · Yifan Zhang[1,2] · Haoyuan Gao[1,2,4] · Xingtao Yu[5] · Jian Cheng[1,2]

Received: 17 December 2021 / Accepted: 23 March 2022

Abstract
Face verification models based on centralized training on large face datasets have achieved excellent performance on various test benchmarks. However, due to the increasingly sophisticated privacy protection law, centrally collecting large amount of face images becomes more difficult. We consider learning a face verification model in the federated setting, where each client has access to the face images of only one class and class embeddings cannot be shared to other clients because of data privacy. In this paper, we propose Federated face verification (FedFV), in which server transfers some equivalent class embeddings to clients so that the clients' class embeddings can be separated far away from each other. We show that our proposed method FedFV outperforms the existing approaches in several face verification benchmarks.

Keywords Equivalent class embeddings · Federated learning · Face recognition · Deep learning

1 Introduction

Due to the breakthrough of deep learning, computer vision [1–5] has been greatly developed in recent years, and face recognition [6–12], as a popular research field of computer vision, has attracted many researchers to study it. Face verification is a subfield of face recognition, which aims to identify whether the two given face images belong to the same identity. There has been many breakthroughs in this field in recent years, such as SphereFace [13], CosFace [14], ArcFace [15], etc. These methods greatly improve the face verification neural network's performance on many face verification test benchmarks and make it mature so that it can be widely used in real life.

The training process of all the methods mentioned above needs a large training dataset. The more data, the better performance. It is common to collect a large amount of data to form a huge dataset and then centrally train the neural network using these data on the server in the past. However, face image is a kind of privacy sensitive biometric data and due to the growing data privacy concerns and the strict legal restrictions, centrally collecting large amounts of face data becomes more difficult and thus centrallized training face verification neural network becomes hard.

Recently in 2016, Federated learning (FL) [16–20], which is a new training framework for machine learning

Communicated by B.-K. Bao.

✉ Yifan Zhang
yfzhang@nlpr.ia.ac.cn

Lingyun Liu
liulingyun2019@ia.ac.cn

Haoyuan Gao
gaohaoyuan2019@ia.ac.cn

Xingtao Yu
yxt601@126.com

Jian Cheng
jcheng@nlpr.ia.ac.cn

[1] The National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Haidian District, Beijing 100190, China

[2] Nanjing Artificial Intelligence Research of IA, Jiangning District, Nanjing 211135, Jiangsu, China

[3] School of Future Technology, University of Chinese Academy of Sciences, Haidian District, Beijing 100049, China

[4] School of Artificial Intelligence, University of Chinese Academy of Sciences, Haidian District, Beijing 100049, China

[5] Administrative Committee of Nanjing Chi-Lin Innovation Park, Jiangning District, Nanjing 211135, Jiangsu, China

Published online: 22 April 2022

to solve the data privacy problems, has been proposed by Google. In this framework, clients and server cooperate to train a model and in the training process, clients are not allowed to share their own data to each other so that it can achieve the goal of protecting the data privacy. Since then, a lot of federated optimization approaches have been proposed [21–26]. However, the conventional federated learning methods can not be directly applied to face verification tasks.

Face verification task needs a model that have the ability to generate face features with strong separability because the learned model is often used to identify images of the identities that do not appear in the training set which is called open-set problem. Thus, in the training process, a positive loss term is needed to keep a face feature as close to the embedding of the class it belongs to as possible and a negative loss term is needed to keep a face feature as far away to the embeddings of other classes as possible [27]. But the class embedding also contains highly sensitive information as it can be used to identify the users so that clients are also not allowed to share their own class embeddings to each other. When in the extreme settings that one client has only access to one class, which is common in real life, the client cannot calculate the negative loss term in the local training round because of lacking of other class embeddings and it will soon lead to a trivial solution [28].

Recently, two works have been proposed to this problem, one is FedAwS [29] and the other is FedUV [27]. These two methods enable that the federated training process can be carried out normally without sharing clients' class embeddings. FedAwS [29] proposed a regularization term which is used in the server to separate the class embeddings and FedUV [27] uses the property of error-correcting codes (ECCs) which can maximize the minimum Hamming distance between distinct codewords. These two methods have comparable performance on several test benchmarks. However, there is still a relatively large performance gap between these methods and centrally training methods.

In this paper, we proposed Federated Face Verification, a framework for training face verification models in the federated settings that one client has only access to one class and the clients cannot share class embeddings with each other. Our contributions are summarized as follows:

- We propose a new kind of vector named equivalent class embeddings which is generated by fusing some clients' original class embeddings. Keeping class embedding far away from the equivalent class embeddings can also make it far away from the original class embeddings. Meanwhile, it is unable to restore the original class embeddings from the equivalent class embedding generated by our proposed method so that it does not lead to the leakage of class embeddings.
- We propose a framework FedFV to train face verification models using equivalent class embeddings in the federated setting mentioned above.
- We test our proposed method FedFV on several famous face verification benchmarks, including LFW [30], Agedb-30 [31] and CFP-FP [32], and it shows that our methods outperforms the existing methods FedAwS [29] and FedUV [27].

2 Related work

2.1 Face verification

Given two face images x_1, x_2, a neural network $g_\theta : X \to \mathbb{R}^d$ parameterized by θ which maps an input from the input space X to a d-dimensional embedding, and a threshold t, we consider x_1, x_2 belong to the same identity if the similarity s between their features $g_\theta(x_1), g_\theta(x_2)$ calculated by $s = g_\theta(x_1)g_\theta(x_2)/(||g_\theta(x_1)|| \cdot ||g_\theta(x_2)||)$ is greater than t and otherwise we consider that they belong to different identities. This is the testing process of face verification. Thus, it is important to train a model that can produce features with strong separability.

In the training process, lots of methods have been proposed to improve the performance of the neural network. Among them, SphereFace [13], CosFace [14], and ArcFace [15] are recent excellent works by modifying the loss function to strengthen the supervision signal. Actually, they are all variants of Softmax Loss described in Eq. (1).

$$L_{\text{softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{w_{y_i}f_i+b_{y_i}}}{\sum_{j=1}^{C}e^{w_j f_i+b_j}} \tag{1}$$

where N is the number of samples in one mini-batch, C is the number of classes, f_i is the d-dimensional feature of the input x_i calculated by $f_i = g_\theta(x_i)$ and y_i is the class label of x_i. $w_j \in \mathbb{R}^d$ denotes the class embedding of class j and b_j is the bias term. We refer to the settings in previous works [13–15, 33–35] and then set $b_j = 0, ||w_j|| = 1, ||f_i|| = 1$. The Softmax Loss in Eq. (1) can be written as follows:

$$\begin{aligned} L'_{\text{softmax}} &= -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s_{y_i,i}}}{\sum_{j=1}^{C}e^{s_{j,i}}} \\ &= -\frac{1}{N}\sum_{i=1}^{N}\log e^{s_{y_i,i}} + \frac{1}{N}\sum_{i=1}^{N}\log\sum_{j=1}^{C}e^{s_{j,i}} \end{aligned} \tag{2}$$

where $s_{j,i}$ denotes the cosine similarity between the class embedding w_j and the face feature f_i. In Eq. (2), the first term is the positive loss term which forces the feature of the training samples close to the embeddings of the classes they belong to, and the second term can be seen approximated

as the negative loss term that keep features far away from the other class embeddings although it includes a term $e^{s_{y_i,i}}$. Thus, the training process in face verification is a process that maximizing the inter-class distances and minimizing the intra-class distances. The methods mentioned above add a margin term on the basis of Softmax Loss to make it more difficult to classify correctly so that the trained neural network has better performance. Actually, when there is only one class, the Softmax Loss equals to 0 all the time so that it cannot be used in FedAwS [29] and FedUV [27].

2.2 Federated learning

Federated learning (FL) is a framework of machine learning in which there exists a server and multiple clients with their own data and the server and clients cooperate to train a machine learning model without sharing data with each other to ensure the data privacy. It was first proposed in 2016 by Google along with an optimization algorithm FederatedAveraging (FedAvg) [18]. FedAvg [18] makes some changes to the traditional SGD algorithm to adapt to the setting of federated learning and is described in Algorithm 1. Now federated learning has been applied in users' digital products, for example, Google keyboard prediction [36]. In the future, federated learning is expected to play a more significant role in real life.

Algorithm 1 FederatedAveraging [18]

Input: C: number of clients, ϵ: fraction of clients selected in each round, T: total round, B: mini-batch size, E: epochs in local training, n_i: number of samples of client i.
Parameters: θ^t: the global model in t-th round, θ_i^t: the local model on client i in t-th round.
Output: θ: the global model.

Server:

Initialize the global model θ^0
$m \leftarrow \max(1, \epsilon \times C)$
for $t = 1, 2, \ldots, T$ **do**
  $S_t \leftarrow$ (random set of m clients)
  Send θ^{t-1} to client i for $i \in S_t$
  Receive θ_i^t from client i for $i \in S_t$
  $\theta^t \leftarrow \frac{\sum_{i \in S_t} n_i \theta_i^t}{\sum_{i \in S_t} n_i}$
end for
return θ^T

Client:

Receive θ^{t-1} from the server.
for $e = 1, \ldots, E$ **do**
  Forward and Backward normally with mini-batch size of B.
end for
Send θ_i^t to the server.

The non-independent and identically distributed (Non-IID) [18] data are a major challenge for federated learning. Lots of approaches were proposed to solve this problem, such as FedProx [24], FedNova [25], etc. These methods are effective in the normal settings. But when in the federated setting that each client has only access to one class and the clients' class embeddings cannot be shared with each other, these methods do not work as the client lacks the negative loss term in the local training process and it will soon lead to a trivial solution.

To our best knowledge, two recent works focused on this problem. The first one is FedAwS [29]. FedAwS proposed a regularization term described in Eq. (3) .

$$\mathrm{reg}_{\mathrm{sp}}(W) = \sum_{i\in[C]} \sum_{i'\neq i} (\max\{0, \delta - d(w_i, w_{i'})\})^2 \tag{3}$$

where $W \in \mathbb{R}^{C\times d}$ is the class embedding matrix and δ is a margin. When clients finished their local training, they send their local models and their own class embeddings to the server and the server uses Eq. (3) to perform an optimization step on the class embedding matrix W to make sure all the class embeddings are separated from each other by at least a margin of δ so that the training process can be carried out normally. It is proved that FedAwS [29] can approach a consistent classifier in performance.

The other work is FedUV [27], which noticed that ECCs can maximize the minimum Hamming distance between distinct codewords. Thus, FedUV [27] let each client i generate a unique vector, and then use ECC algorithm to encrypt the unique vector to get a secret vector v_i. These vectors form a set V. Due to the properties of ECCs, the secret vectors in V are already separated from each other so that in the training process, only the positive loss term described in Eq. (4) is required to make feature close to its corresponding secret embedding.

$$L_{\mathrm{pos}} = -\frac{1}{N}\sum_{i=1}^{N} \max\left(0, 1 - \frac{1}{||v_{y_i}||^2} v_{y_i}^T W' g_\theta(x_i)\right) \tag{4}$$

In Eq. (4), W' is a parameter matrix that maps the secret vector v_{y_i}, which belongs to class y_i, to a d-dimensional secret class embedding. Different from FedAwS [29], these secret vectors are invisible to the server so that the client is also confidential to the server.

Although both of these two methods enable the model to be trained normally in extreme federated settings mentioned above and can protect the class embeddings of the clients to a certain extent, our later experiments results show that they still have a large gap from the performance of centralized training.

3 Method

In this paper, we proposed a new method Federated face verification (FedFV) which uses equivalent class embeddings to train a network in federated setting. We will introduce our proposed method in detail in this section.

3.1 Equivalent class embedding

As is mentioned above, the reason why direct application of conventional federated methods does not work is that client cannot calculate the negative loss term so that the class embeddings cannot be separated from each other. A naive idea to solve this problem is to find a way to let a client have negative loss term without disclosing other clients' class embeddings. Thus, we consider to fuse the information of the class embeddings to generate new vectors and meanwhile, the client cannot restore the information of the original class embeddings from the new vectors. Specifically, our

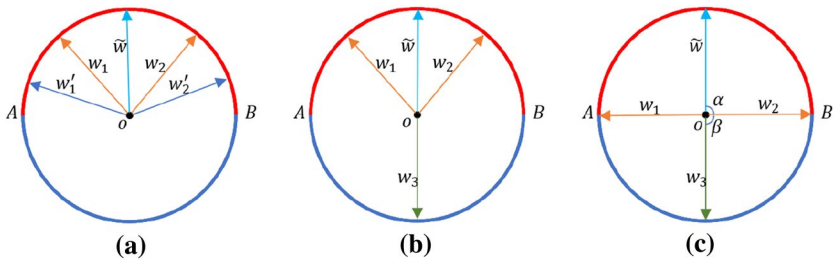

Fig. 1 Illustration of Equivalent Class Embedding. w_i denotes class embedding and $\tilde{w}$ is the generated equivalent class embedding which is obtained by applying Eq. (5) to w_1 and w_2. **a** It is unable to restore the original class embeddings w_1 and w_2 from $\tilde{w}$ because any two vectors symmetric about $\tilde{w}$ on the red arc, such as w_1' and w_2', can produce $\tilde{w}$ using Eq. (5). **b** When a class embedding w_3 is trained to keep far away from the equivalent class embedding $\tilde{w}$, it will also be far away from the original class embeddings w_1 and w_2. **c** Even in the extreme case that α is extremely close to 90°, w_3 will still maintain a large included angle with w_1 and w_2

proposed method simply uses averaging and normalization operations and is described as follows:

$$\begin{aligned} \bar{w} &= \frac{1}{2}(w_i + w_j) \\ \tilde{w} &= \frac{\bar{w}}{||\bar{w}||} \end{aligned} \tag{5}$$

where w_i and w_j are any two class embeddings that have been normalized and $\tilde{w} \in \mathbb{R}^d$ is the vector we need.

We call the generated vector $\tilde{w}$ the Equivalent Class Embedding. As is shown in Fig. 1, the client cannot restore w_1 and w_2 from the generated vector $\tilde{w}$. The only information client can obtain from $\tilde{w}$ is that the original class embeddings are on the red semicircle and actually in the training process, it is a d-dimensional semi-hypersphere which is a very huge space. Even if the client uses reverse method, such as [37], to restore the face from $\tilde{w}$, it cannot know the similarity between this face and the original face as there are infinite pairs of embeddings that can be used to generate $\tilde{w}$ by our proposed method. In addition, we also provided an experiment (FedFV(k)) in Sect. 4.4 about using more than two original embeddings to generate one equivalent class embedding. It also works well and in this time it is even more impossible to obtain the private information of other clients through the equivalent class embedding $\tilde{w}$. Meanwhile, when in local training process, client can bring $\tilde{w}$ into the negative loss term in Eq. (2) so that the client can make its class embedding w_3 far away from $\tilde{w}$ which will also make w_3 far away from the original class embeddings w_1 and w_2 and this is why we name $\tilde{w}$ Equivalent Class Embedding.

When a client gets an Equivalent Class Embedding $\tilde{w}$, the client cannot restore the original class embeddings from it. The detailed proof is as follows: Firstly, $\tilde{w}$ is generated by equation $\tilde{w} = \bar{w}/||\bar{w}||$. According to the properties of normalizing, there are innumerable solutions for $\bar{w}$ when given $\tilde{w}$. And for any solution $\bar{w}$, it is generated by equation $\bar{w} = \frac{1}{2}(w_1 + w_2)$, or by equation $\bar{w} = \frac{1}{k}(w_1 + w_2 + \cdots + w_k)$ in Sect. 4.4 where we use k original class embeddings whose norms are all 1 to generate $\tilde{w}$ instead of 2. No matter what the value of k is, as long as the norm of $\bar{w}$ is less than 1, the above equation has at least one set of solutions. And there are innumerable $\bar{w}$ whose norm is less than 1. In addition, the class embeddings used to generate Equivalent Class Embeddings are randomly selected. Therefore, no matter what the value of k is, the client cannot restore the original class embeddings from the given Equivalent Class Embeddings.

3.2 Federated face verification

Now we already have the equivalent class embeddings, and in this section, we will introduce our proposed framework Federated face verification (FedFV) for training face verification model in detail.

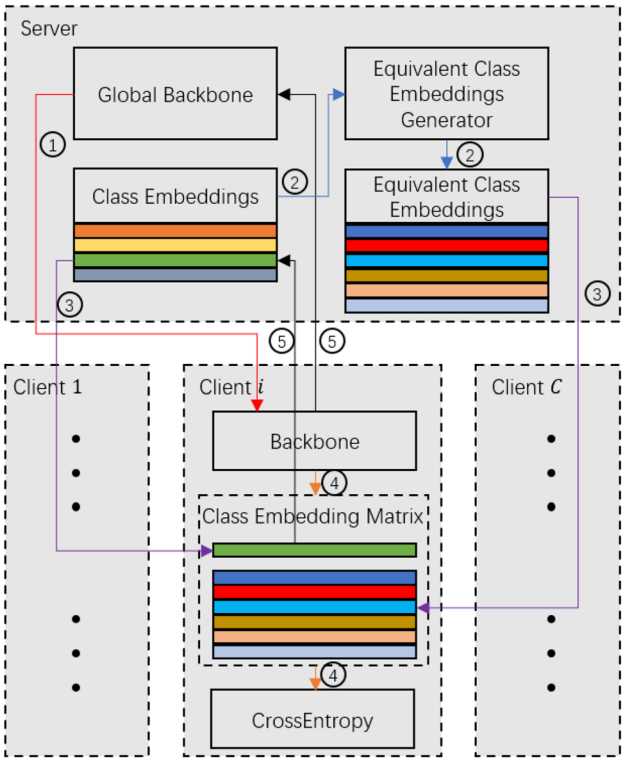

Fig. 2 The framework of FedFV. The numbers with circles represent the steps of the training process. ①: sending global backbone parameters to clients; ②: generating Equivalent Class Embeddings with original class embeddings; ③: sending class embedding and Equivalent Class Embeddings to the corresponding client; ④: training model using the data in the client; ⑤: sending the gradients of the parameters back to the server. In our proposed framework, there exists a generator in the server to generate equivalent class embeddings using clients' class embeddings. Then the server sends them to each client selected and the clients use them and its own class embedding to form a matrix to calculate Softmax Loss so that all class embeddings can be separated from each other

Different from the conventional federated methods, in FedFV, the server has an equivalent class embedding generator as shown in Fig. 2. At each round of global training, the server first selects $m = \max(1, \epsilon \times C)$ clients to participate in this round of training where $\epsilon \in (0, 1]$. Then the generator uses the class embeddings of clients which are not selected to generate n equivalent class embeddings because it is impossible to let a class embedding far away from itself. Specifically, the generator randomly selects two clients' class embeddings at a time, and then generates an equivalent class embedding according to the Eq. (5) until n embeddings are all generated. After that, the server sends the global model θ, the corresponding class embedding w_i and the generated equivalent class embedding matrix $\tilde{W}$, which is formed by stacking all equivalent class embeddings in the vertical direction, to client i. The clients then start there local training process using their

own data. Significantly, $\tilde{W}$ is fixed during the local training process because if it is changeable, it will also lead to trivial solution where each row of $\tilde{W}$ would be the same and the client's class embedding w_i would equal to $-\tilde{w}$ which will make our method not work. When finished the local training, the client i only sends its local model θ_i and its class embedding w_i to the server. Then server aggregates all the local model to obtain the new global model and uses the new class embeddings to generate new equivalent class embeddings for the next round of training. This is one round of training process in FedFV and the complete training process is detailed in Algorithm 2.

Algorithm 2 Federated Face Verification

Input: C: number of clients and also number of classes, ϵ: fraction of clients selected in each round, T: total round, B: mini-batch size, E: epochs in local training, n: number of equivalent class embeddings, n_i: number of samples of client i.
Parameters: θ^t: the global model in t-th round, θ_i^t: the local model on client i in t-th round, w_i^t: class embedding of client i in t-th round.
Output: θ: the global model.

Server:

Initialize the global model θ^0 and class embeddings $w_i^0, i = 1, \ldots, C$
$m \leftarrow \max(1, \epsilon \times C)$
for $t = 1, 2, \ldots, T$ **do**
  $S_t \leftarrow$ (random set of m clients)
  $R_t \leftarrow$ (rest set of clients)
  for $j = 1, 2, \ldots, n$ **do**
    Randomly select two class embeddings $w_{i_1}^{t-1}$ and $w_{i_2}^{t-1}$ from R_t
    Generate $\tilde{w}_j^t$ using $w_{i_1}^{t-1}$ and $w_{i_2}^{t-1}$
  end for
  $\tilde{W}^t \leftarrow [\ \tilde{w}_1^t\ \tilde{w}_2^t \cdots\ \tilde{w}_n^t]$(vertically stack)
  Send θ^{t-1}, w_i^{t-1} and $\tilde{W}^t$ to client i for $i \in S_t$
  Receive θ_i^t, w_i^t from client i for $i \in S_t$
  $\theta^t \leftarrow \frac{\sum_{i \in S_t} n_i \theta_i^t}{\sum_{i \in S_t} n_i}$
end for
return θ^T

Client:

Receive θ^{t-1}, w_i^{t-1} and $\tilde{W}^t$ from the server.
$W = [w_i^{t-1}\ \tilde{W}^t]$(vertically stack)
for $e = 1, \ldots, E$ **do**
  Forward and Backward normally with $\tilde{W}^t$ fixed.
end for
Send θ_i^t, w_i^t to the server.

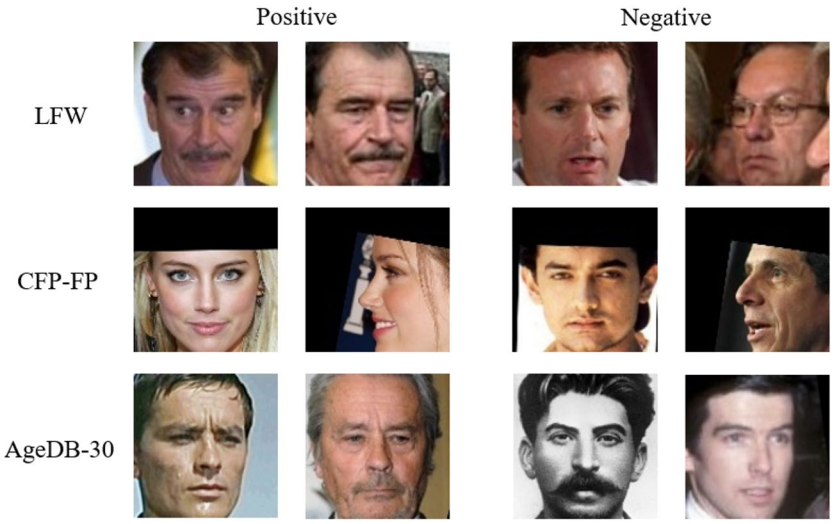

Fig. 3 Example pairs of all test sets. Each face image has already been aligned and resized to 64×64. The first column is the positive pairs and the second column is the negative pairs. LFW [30] focuses on the unconstrained face verification, CFP-FP [32] focuses on the frontal profile face verification and AgeDB-30 [31] focuses on the face verification of age invariance

Table 1 Verification accuracy on several test benchmarks. FedFV-*n* denotes the FedFV method with *n* equivalent class embeddings

Method	LFW(%)	AgeDB-30(%)	CFP-FP(%)
FedFV-1	63.55	55.32	58.56
FedFV-5	78.73	59.58	70.22
FedFV-10	81.18	64.45	73.09
FedFV-50	82.50	66.48	74.50
FedFV-100	82.63	66.68	74.10
FedFV-500	**83.22**	66.83	74.23
FedFV-1000	83.03	**67.05**	**74.94**

Bold values represent the highest accuracy on the current test set

Table 2 Verification accuracy when using different number of class embeddings to generate one equivalent class embedding

Method	LFW(%)	AgeDB-30(%)	CFP-FP(%)
FedFV(2)	82.63	**66.68**	74.10
FedFV(3)	82.90	66.23	**74.30**
FedFV(5)	**83.00**	66.52	73.76
FedFV(10)	82.43	66.27	73.69
FedFV(50)	81.30	64.55	73.37
FedFV(100)	81.47	64.87	72.87

Bold values represent the highest accuracy on the current test set

We use 100 equivalent class embeddings by default. Thus, FedFV(k) denotes FedFV with 100 equivalent class embeddings, each of which is generated by fusing k real class embeddings. Basically, the more class embeddings are fused, the lower accuracy, but the impact is not great

4 Experiment

4.1 Datasets

CASIA-Webface [38] contains nearly 0.5M face image data of about 10K identities. In our experiments, we randomly select 1000 identities from those who had at least 30 images to form an 30K face image dataset as our training set. Thus, we have 1000 clients in our experiments. To test the performance of FedFV on face verification task, we evaluate on three test benchmarks, LFW [30], CFP-FP [32] and AgeDB-30 [31] (Fig. 3). All face images in training set and test set were re-sized to 64×64.

LFW [30] is a face dataset used for unconstrained face recognition which contains about 13,000 face images of 5749 identities, and among them there are 1680 identities who have two or more face images. This is a dataset often used for face verification test. Its testing protocol provides a list containing 6000 pairs of face images.

CFP [32] contains 500 identities and each identity has 10 frontal face images and 4 profile face images. It is used for frontal-frontal (FF) and frontal profile (FP) face verification. In this paper, we use CFP-FP [32] to evaluate FedFV, which contains 7000 pairs of frontal profile face images.

AgeDB-30 [31] focus on the problem of face verification of different ages. It contains 6000 pairs of face images with a age difference of 30 years.

The face image pairs of all the datasets above are half-positive and half-negative.

4.2 Experiment settings

In our experiments, we use the same backbone as FedUV [27] in face verification test which contains five basic blocks and a FC layer. Each basic block consists a Convolution layer, a Relu layer, a Maxpool layer and a Groupnorm [39]

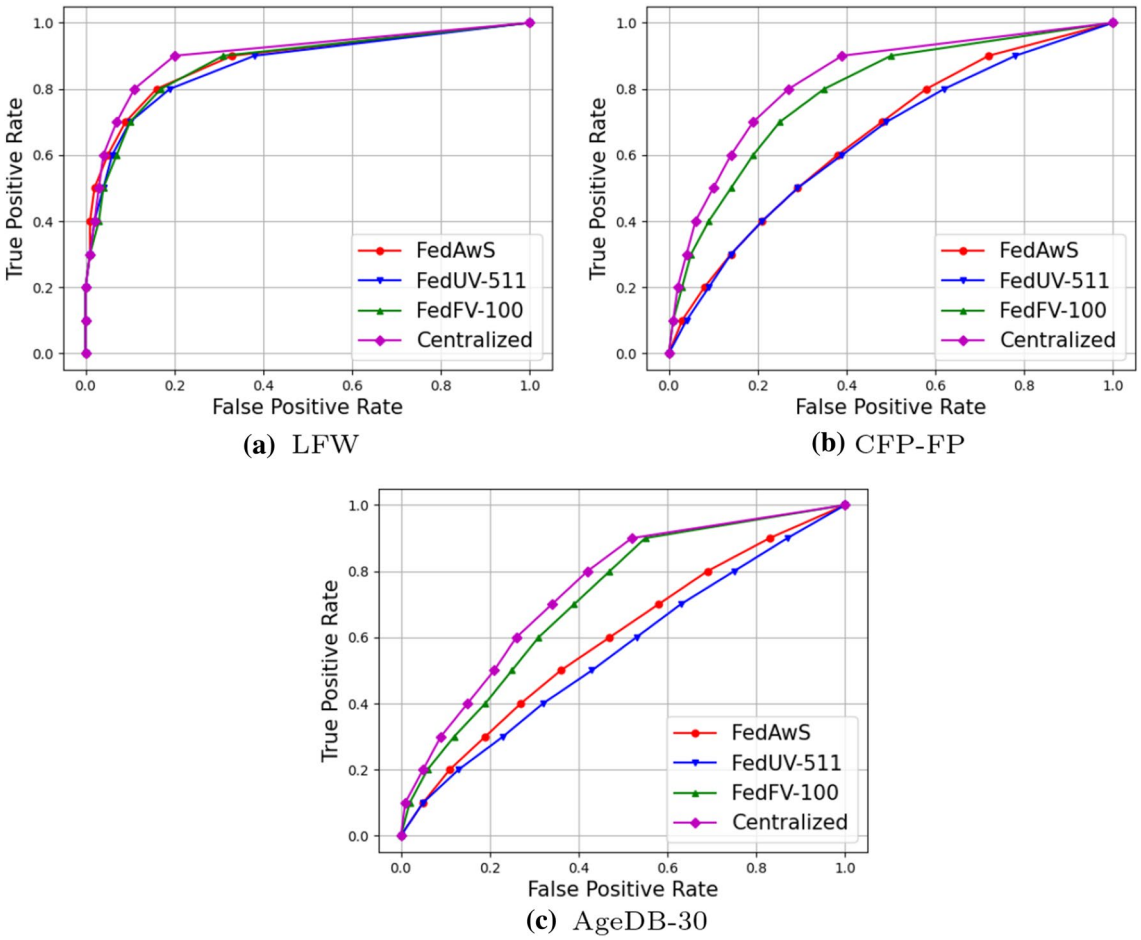

Fig. 4 ROC curves for models trained with FedAwS [29], FedUV-511 [27], FedFV-100 and centralized training on three benchmarks. FedUV-511 denotes FedUV with code length of 511. It shows that three federated methods have comparable performances on LFW [30] but FedFV-100 outperforms other federated methods on CFP-FP [32] and AgeDB-30 [31]. The method of centralized training always performs best on all benchmarks

Table 3 Verification accuracy of different methods. Centralized means that training model in a non-federated manner

Method	LFW(%)	AgeDB-30(%)	CFP-FP(%)
Centralized	**86.68**	**69.27**	**77.17**
FedAwS [29]	82.47	59.53	69.56
FedUV-127 [27]	78.83	56.35	68.03
FedUV-255 [27]	79.42	57.23	68.41
FedUV-511 [27]	81.60	58.93	68.63
FedFV-100	*82.63*	*66.68*	*74.10*

Bold values indicate that Centralized is the best method. Italic values represent the highest accuracy of the current test set when using federated learning method

Our method performs best among all federated methods in these three face verification benchmarks

layer. We set the dimension of image feature to 512 so the number of channels for all Convolution layer is 64, 128, 256, 512 and 512 respectively. If FedUV [27] is used, there is a Scaling layer in the last.

We train all the models using FedAvg [18] algorithm with 1 local epoch and 15,000 rounds with eight clients selected in each round. In client's local training, we use SGD optimizer with initial learning rate of 0.1 and the learning rate is divided by 10 at 10,000 and 13,000 round. FedAwS [29] and FedUV [27] are trained with the loss functions proposed in their own paper respectively. Centralized training and our proposed method FedFV are trained with the loss function described in Eq. (2).

4.3 Ablation study

In FedFV, we have a hyper-parameter n which is the number of the equivalent class embeddings. In Table 1, we explore the effect of the number of the equivalent class embeddings on the accuracy on test benchmarks. The results show that

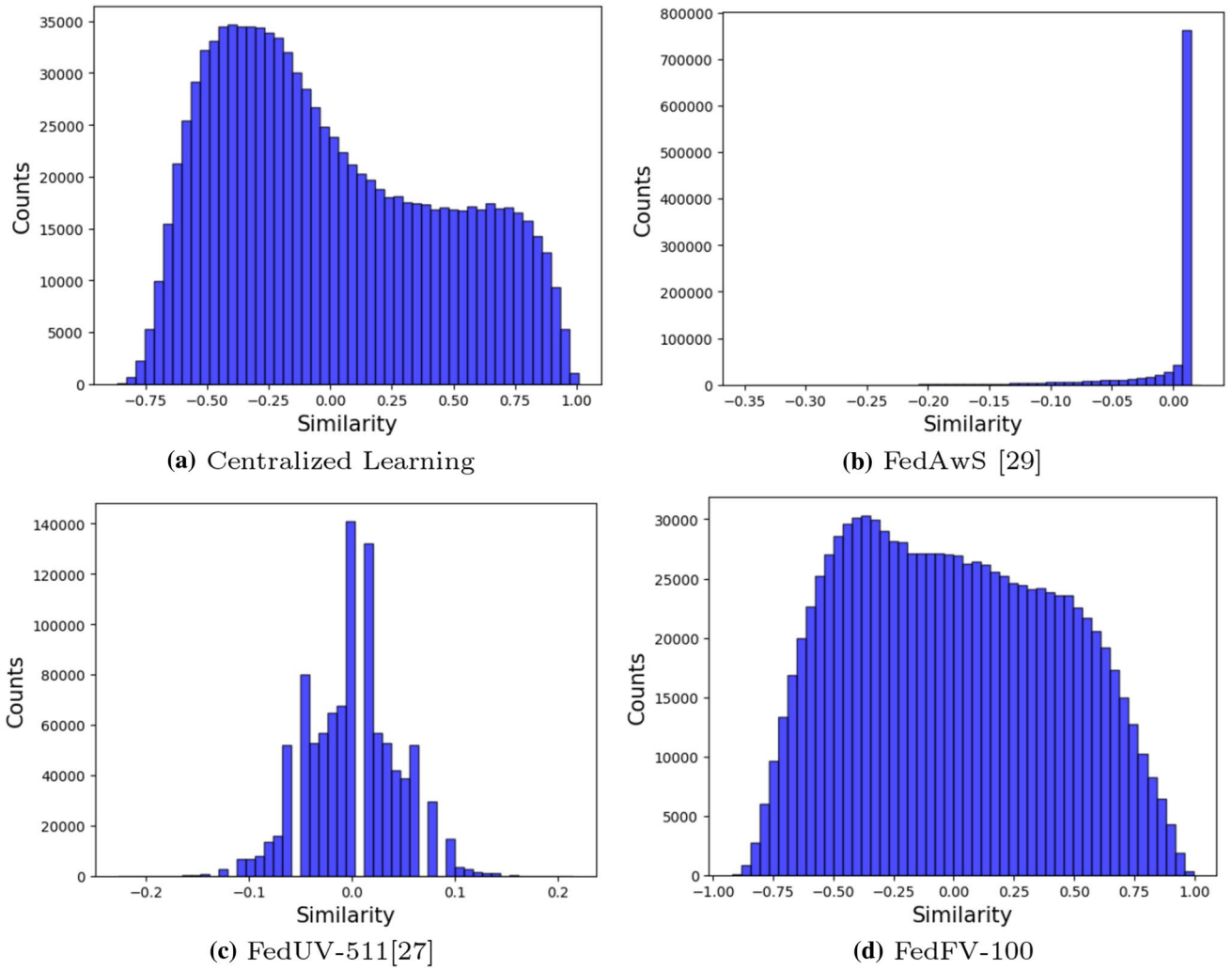

Fig. 5 Similarity distribution of class embeddings of different methods. It shows that most of the cosine similarities of all pairs of class embeddings in centralized learning are around −0.35 which is about 110°. Most of the similarities of FedAwS and FedUV are around 0 (90°) while the similarity distribution obtained by our method is most similar to that of centralized learning and most of them are also around −0.35 so that our method can obtain performance close to centralized learning

more equivalent class embeddings lead to higher performance of the models. We can see that the fastest increment occurs when n changes from 1 to 5 and when n gets larger and larger, the accuracy of the model increases more and more slowly until saturation. Thus, when only considering accuracy, the larger n is better. However, in real life, communication cost is a problem that must be considered in federated learning. In FedFV, larger n means larger communication cost of download as there is no need to upload the equivalent class embeddings. If the data type is float32, the size of one 512-d equivalent class embedding is 2K. When n is 100, the size is 200K which basically puts no pressure on today's download bandwidth. However when n is 1000, the size is 2M, which puts more pressure on download bandwidth without a significant improvement on accuracy. Thus, we finally set n to 100 in rest experiments.

4.4 Number of class embeddings used

In each of the above experiments, each equivalent class embedding is generated by fusing two real class embeddings. If only using two class embeddings will still leak some information, such as the semi-hypersphere information of the original class embeddings, causing a certain risk, it will be safer to fuse with three or more class embeddings, because in this time, clients cannot even obtain the semi-hypersphere information. Then, whether the fusion of three or more class embeddings will work, we have carried out some experiments.

The results are shown in Table 2. We can see that, basically, when generating a equivalent class embedding, the more class embeddings used, the lower accuracy we will obtain because more information will be lost. But even using

a lot of class embeddings, this accuracy drop will not be very large, especially when the number is less than 10, the accuracy is nearly the same. Thus, when considering to safer generate equivalent class embeddings, the best number of class embedding used is 3 as it will not reduce the accuracy and will not bring more computational overhead to the server.

4.5 Comparison with other methods

In this section, we compare our method with the existing federated methods and centralized training on the test benchmarks mentioned above. The results are shown in Fig. 4 and Table 3. We can see that there is no doubt that centralized training always performs best on all benchmarks. In the remaining federated methods, our method FedFV outperforms other methods on all benchmarks and improves the accuracy of LFW [30], CFP-FP [32] and AgeDB-30 [31] by 0.16%, 4.54% and 7.15% respectively. Although our method has little improvement in LFW [30], we can see that the accuracy of three federated methods is very close to that of centralized training. We believe that LFW [30] is a simpler test benchmarks than the other two benchmarks as it does not have the interference of complex changes, such as age changes, perspective changes, etc. Therefore, the federated methods are easy to achieve the performance that is not much different from centralized training and thus our method has not improved much. When it turns to CFP-FP [32] and AgeDB-30 [31], there is a large gap between the accuracy of centralized training and the two federated methods FedAwS [29] and FedUV [27]. Our method can effectively narrow the performance gap between federated methods and centralized training on these difficult test benchmarks.

To explore the learned embedding, we further conduct an experiment about the similarities between the learned class embeddings. When the training process is finished, we calculate the cosine similarities of all possible pairs of class embeddings and make statistics on them. The results of different methods are shown in Fig. 5 .

From Fig. 5, it is shown that firstly the cosine similarity distribution obtained by our method is most similar to that of centralized learning and this is an important reason why FedFV can achieve good performance that close to the method of centralized learning comparing to FedAwS and FedUV. In addition, most of the cosine similarities of all pairs of class embeddings in centralized learning and in FedFV are around -0.35 which is about $110°$ while most of the similarities in FedAwS and FedUV are around 0 which is $90°$. This indicates that FedFV is indeed able to keep the classes embeddings away from each other. In conclusion, our method FedFV can obtain the similarity distribution similar to that of centralized learning and can keep the class embedding away from each other so that FedFV can get better performance than FedAwS and FedUV.

5 Conclusion

In this paper, we proposed a new framework FedFV to train a face verification model in the federated setting where each client has only access to one class and clients are not allowed to share class embeddings with each other. In our method, the server will gather all clients' class embeddings and use them to generate some equivalent class embeddings. The clients will receive these equivalent class embeddings and use them to calculate the negative loss term so that all class embeddings can be equivalently separate from each other. Our experiment results show that our method outperforms the existing methods FedAwS [29] and FedUV [27] on several face verification test benchmarks. However, our method still has a disadvantage that our method needs the clients exchange their class embeddings with the server compared with FedUV [27]. This will be what we hope to improve next.

Acknowledgements This work was supported in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDA27040300, Jiangsu Key Research and Development Plan (No. BE2021012-2), and NSFC 61906195, 61876182.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G.V., Krpalkova, L., Riordan, D., Walsh, J.: Deep learning vs. traditional computer vision. In: Science and Information Conference, pp. 128–144 (2019). Springer
2. Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E.: Deep learning for computer vision: a brief review. Comput. Intell. Neurosci. **2018** (2018)
3. Ioannidou, A., Chatzilari, E., Nikolopoulos, S., Kompatsiaris, I.: Deep learning advances in computer vision with 3d data: a survey. ACM Comput. Surv. (CSUR) **50**(2), 1–38 (2017)
4. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
6. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition (2015)
7. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
8. Sun, Y., Wang, X., Tang, X.: Deep learning face representation from predicting 10,000 classes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1891–1898 (2014)

9. Sun, Y., Chen, Y., Wang, X., Tang, X.: Deep learning face representation by joint identification-verification. In: NIPS, pp. 1988–1996 (2014). http://papers.nips.cc/paper/5416-deep-learning-face-representation-by-joint-identification-verification
10. Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2892–2900 (2015)
11. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
12. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: European Conference on Computer Vision, pp. 499–515 (2016). Springer
13. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
14. 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 (CVPR) (2018)
15. 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 (CVPR) (2019)
16. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R.G.L., Eichner, H., Rouayheb, S.E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P.B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., sTICH, S.U., Sun, Z., Suresh, A.T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F.X., Yu, H., Zhao, S.: Advances and Open Problems in Federated Learning (2021)
17. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
18. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282 (2017). PMLR. https://proceedings.mlr.press/v54/mcmahan17a.html
19. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020). https://doi.org/10.1109/MSP.2020.2975749
20. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, H.B., Overveldt, T.V., Petrou, D., Ramage, D., Roselander, J.: Towards Federated Learning at Scale: System Design (2019)
21. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency (2017)
22. Hsu, T.-M.H., Qi, H., Brown, M.: Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification (2019)
23. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143 (2020). PMLR
24. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated Optimization in Heterogeneous Networks (2020)
25. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization (2020)
26. Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, H.B.: Adaptive Federated Optimization (2021)
27. Hosseini, H., Park, H., Yun, S., Louizos, C., Soriaga, J., Welling, M.: Federated Learning of User Verification Models Without Sharing Embeddings (2021)
28. Bojanowski, P., Joulin, A.: Unsupervised learning by predicting noise. In: International Conference on Machine Learning, pp. 517–526 (2017). PMLR
29. Yu, F., Rawat, A.S., Menon, A., Kumar, S.: Federated learning with only positive labels. In: International Conference on Machine Learning, pp. 10946–10956 (2020). PMLR. https://proceedings.mlr.press/v119/yu20f.html
30. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. In: Learned-Miller, E., Ferencz, A., Jurie, F. (eds) Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition, Marseille, France (2008). https://hal.inria.fr/inria-00321923
31. Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., Zafeiriou, S.: Agedb: The first manually collected, in-the-wild age database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
32. Sengupta, S., Chen, J.-C., Castillo, C., Patel, V.M., Chellappa, R., Jacobs, D.W.: Frontal to profile face verification in the wild. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9 (2016). https://doi.org/10.1109/WACV.2016.7477558
33. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. In: ICML, vol. 2, p. 7 (2016)
34. Wang, F., Xiang, X., Cheng, J., Yuille, A.L.: Normface: L2 hypersphere embedding for face verification. In: Proceedings of the 25th ACM International Conference on Multimedia. MM '17, pp. 1041–1049. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3123266.3123359
35. Wang, F., Cheng, J., Liu, W., Liu, H.: Additive margin softmax for face verification. IEEE Signal Process. Lett. **25**(7), 926–930 (2018). https://doi.org/10.1109/LSP.2018.2822810
36. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D.: Federated Learning for Mobile Keyboard Prediction (2019)
37. Duong, C.N., Truong, T.-D., Luu, K., Quach, K.G., Bui, H., Roy, K.: Vec2face: Unveil human faces from their blackbox features in face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6132–6141 (2020)
38. Yi, D., Lei, Z., Liao, S., Li, S.Z.: Learning Face Representation from Scratch (2014)
39. Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)