Complex & Intelligent Systems (2023) 9:6891–6903
https://doi.org/10.1007/s40747-023-01110-7

ORIGINAL ARTICLE

A federated learning algorithm using parallel-ensemble method on non-IID datasets

Haoran Yu[1] · Chang Wu[1] · Haixin Yu[1] · Xuelin Wei[1] · Siyan Liu[1] · Ying Zhang[1]

Received: 6 December 2022 / Accepted: 16 May 2023 / Published online: 6 June 2023

Abstract

Traditional federated learning algorithms suffer from considerable performance reduction with non-identically and independently distributed datasets. This paper proposes a federated learning algorithm based on parallel-ensemble learning, which improves performance for image classification on these datasets. The training process of this algorithm includes basic federation learning and meta federation learning. First, several basic models are trained through cross-validation of federated learning, and then the meta-model is trained using the prediction results of the validation sets. In the training process, the training of different basic models is parallel. In prediction, meta-model is used to aggregate the output of the basic models to get the final prediction results. Our algorithm can achieve higher accuracy than traditional federated learning when using non-independent identically distributed datasets for image classification. Our algorithm aggregates different models through federated learning based on parallel-ensemble method, and improves the image classification performance of federated learning on non-independent identically distributed datasets.

Keywords Federated learning · Ensemble learning · Non-IID · Image classification

Introduction

With the proliferation of smartphones, sensors, and other edge devices, the demand for distributed machine learning is increasing. As an important technology in distributed machine learning, Federated learning is a setting for centralized model training on data distributed to clients [1]. In federated learning, the client will not send the original data directly to the server but will upload the intermediate results of the training, which the server will aggregate. Combined with encryption technology, data privacy can be well protected. Aggregation algorithms, device heterogeneity, privacy protection, communication optimization and semi-supervised federated learning are all research directions of federated learning.

However, the heterogeneity of edge devices brings challenges to federated learning. The challenges faced by federated learning are mainly divided into device heterogeneity, data heterogeneity, and model heterogeneity [2]. Data heterogeneity means that the clients' data are not independent and identically distributed. In the traditional setting, data are stored centrally, and the central model may access all data information. However, in federated learning, data are only stored locally, resulting in inconsistencies in data distribution, including differences in feature distributions, label distributions, and concepts [3]. The non-IID (non-independent identically distributed) between data will greatly impact the performance of federated learning [4]. For example, if the same experimental parameters are applied in an

Haoran Yu and Chang Wu are co-first authors of this paper.

✉ Chang Wu
changwu@uestc.edu.cn

Haoran Yu
202022010314@std.uestc.edu.cn

Haixin Yu
974095304@qq.com

Xuelin Wei
202021010312@std.uestc.edu.cn

Siyan Liu
laurel64zhu@126.com

Ying Zhang
15847693106@163.com

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Xiyuan Avenue, Chengdu 611731, Sichuan, People's Republic of China

image classification task, the non-IID dataset may lower prediction accuracy by 40% or more.

In addition, the relevant work in other fields is also of great help to our research. The idea of joint iteration of distributed devices such as federated learning is similar to iterative learning control in linear systems. Zhuang et al. propose an optimal ILC algorithm for linear time-invariant multiple-input-multiple-output (MIMO) systems with nonuniform trial lengths under input constraints [5]. Djordjevic et al. consider control problem of the HSA with unknown dynamics, based on adaptive dynamic programming via output feedback [6]. Zhou et al. propose a framework of point-to-point ILC extended within discrete linear time-invariant (LTI) system [7].

This paper proposes a new algorithm to improve the performance of federated learning on the non-IID datasets. Based on traditional federated learning, this algorithm introduces Stacking [8] in ensemble learning and employs meta-model to combine multiple basic deep learning models. It shows that multiple basic models are trained by cross-training and then used to generate a new dataset for training the meta-model. To ensure the generalization of the model, it takes use of federated Learning's properties, including cross-validating datasets from various clients and trains numerous models in parallel. When predicting, the data are first input into the model generated by cross-training, and the outputs are averaged to obtain the prediction results of each basic model. Then, the prediction results of basic models are used as the input of meta-model, which generates the final result. This paper uses ResNet [9], VGG [10], and DenseNet [11] as basic models and Linear Regression (LR) as the meta-model in the experiment. After that, the training sets in the MNIST [12] and CIFAR-10 [13] datasets are divided into five non-IID federated learning datasets to be the training set for each client. The test sets in these datasets are directly used as the test set for the experiment. Finally, the classification accuracy is significantly improved compared with using basic models directly for federated learning.

The main contributions of this work are illustrated as follows:

- Our algorithm uses an ensemble learning method to combine multiple deep learning models to improve the performance of federated learning in non-IID datasets.
- Our algorithm uses parallel computing in the training process, which can train multiple independent deep learning models at the same time to reduce training time.
- The main contributions of this work are illustrated as follows: we compare our algorithm with FedAvg [14] and FedProx [15] under a variety of datasets and experimental parameters, and discuss the improvement and feasibility of our algorithm.

The rest of this paper is organized as follows. In Sect. "Background", we introduced relevant background knowledge. Section "Algorithm" proposes our algorithm in details. Furthermore, Sect. "Experiment" analyzes the performance of our algorithm in the case of different datasets and parameters. Finally, Sect. "Summarize" gives a summary.

Background

This section first presents the development process and research direction of federated learning, then describes the characteristics of the non-IID dataset and its impact on federated learning. Finally, it briefly introduces ensemble learning and its combination with federated learning.

Federated learning

The term 'federated learning' was first coined in 2016 by Jakub et al. from Google [1]. Federated learning is a machine learning setting where the goal is to train a high-quality centralized model, while the training data exist in the client in a distributed form. The original intention of federated learning was to solve the problem of Android mobile phone users updating the model locally. Later studies have also designed federated learning algorithms for sensor-wearing human activity recognition (HAR) [16]. In a broad sense, federated learning, as a distributed machine learning paradigm, can effectively solve the problem of data silos. It enables participants to model together without sharing data, breaking the data silos, and realizing collaborative training. For example, Zhang et al. proposed a federated learning method for mechanical fault diagnosis to improve learning performance while maintaining privacy [17]. The types of federated learning include horizontal, vertical, and hybrid federated learning [18].

Federated learning consists of multiple rounds of training. In each round, each client independently calculates the update of the current global model based on its local data and sends it to the central server. The server will aggregate the updates from the clients to generate a new global model. The aggregation algorithm is one of the research focuses of federated learning. FedAvg [14] is first proposed to obtain the global model by taking the weighted average of all local models engaged in learning. FedProx [15] tries to solve systems heterogeneity and statistical heterogeneity by adding a proximal term to the local model's objective function of FedAvg. FedDC [19] introduces lightweight modifications in the local training phase to utilize this learned local drift variable to bridge the gap, i.e., conducting consistency in parameter-level. In addition, algorithms such as FedPD [20] and FedBN [21] attempt to solve various challenges of federated learning such as non-convex objective functions and

feature shift. In the following experiments, we selected the FedAvg and FedProx, which solve similar problems with our algorithm, as comparison.

Non-IID

In traditional machine learning, the data are stored in the same machine. It is assumed that the data are independently sampled from the same distribution, i.e., Independently identical distribution. However, in distributed computing, data distribution among devices typically varies greatly, implying that data are non-independent and identically distributed. Non-independence refers to existing associations between different data instead of independent sampling; different distribution means that the distribution of the sample space of different data sampling is different.

In federated learning, we normally assume that different edge devices collect data independently, so only different distribution is considered. The distribution in the training data can be expressed as $P(x, y) = P(y|x)P(x) = P(x|y)P(y)$, so it can be divided into different feature distribution $p(x)$, different label distribution $p(y)$ and different concepts $p(y|x)$ and $p(x|y)$. Zhao et al. examined the performance of FedAvg on the non-IID dataset and pointed out that Weight Divergence is the main reason for the performance degradation of FedAvg on the non-IID dataset [4].

For non-IID dataset, many studies have improved model updating and model aggregation [3] to enhance the robustness of the global model. Other studies use personalized federated learning to optimize the global model for different users [22]. Ma et al. propose Layer-wised Personalized Federated learning (pFedLA) to optimize the personalized model aggregation for clients with heterogeneous data [23]. Of course, it is also an effective method to enhance the non-IID data at the data layer [24]. The algorithm proposed in this paper combines federated learning and ensemble learning to improve the performance of FedAvg on the non-IID dataset.

Ensemble learning

Ensemble learning executes learning tasks by building and combining multiple learners. Generally speaking, ensemble learning will first generate a set of individual learners and then combine them with a certain strategy, and the individual learners are usually selected from existing learning algorithms. Ensemble learning may take various forms. Boosting is a family of algorithms that can advance weak learners to strong learners, and one of the most famous representative is AdaBoost [25]. In addition, Bagging is the prominent representative of the parallel-ensemble learning, which extends the Random Forest [26]. When the training dataset is large, a effective strategy is to combine multiple learners by the Learning Method. Stacking [8] is a typical representative of the Learning Method.

Many studies have used ensemble learning to solve some problems of federated learning. Lin et al. proposed a robust federated meta-learning method [27] and verified its convergence. FedBoost [28] combined with Boosting reduces the communication cost of federated learning. Federated Forest [29] proposes a federated forest framework based on the CART tree and bagging in terms of longitudinal federated learning. Li et al. studied how to train GBDT in the context of federated learning with a focus on horizontal federated learning [30]. The algorithm proposed in this paper combines federated learning and stacking to improve the performance of FedAvg in the scenario of non-IID image classification.

Algorithm

This algorithm combines ensemble learning with the traditional federated learning algorithm. The overall structure is shown in Fig. 1. The algorithm consists of two major steps: basic federated learning and meta federated learning. During training, client-side cross-validation is used to train multiple basic models in parallel and use the prediction results of basic models to train the meta-model. When making the prediction, the test data are first input into basic models, and then the meta-model is used to aggregate the output of the basic models to receive the final prediction result. This section describes the algorithm in detail.

Federated learning

FedAvg is used as the aggregation algorithm in this work for federated learning. Here is a brief introduction to the objective function and training process of the FedAvg algorithm.

Suppose m clients are participating in model training. P_i represents the index collection of the dataset of client i. $n_i = |P_i|$, which is the size of the dataset. The objective function of the entire federated learning training can be expressed as

$$\min_{\omega \in \mathbb{R}} f(\omega) \quad where \quad f(\omega) = \sum_{i=1}^{m} \frac{n_i}{n} F_i(\omega) \tag{1}$$

$F_i(\omega)$ represents the objective function of client i, namely

$$F_i(\omega) = \frac{1}{n_i} \sum_{j \in P_i} f_j(\omega) \tag{2}$$

The training process of federated learning is divided into multiple rounds. Considering differences in computing power and network conditions, not all m clients are required to participate in each round of federated learning. The clients

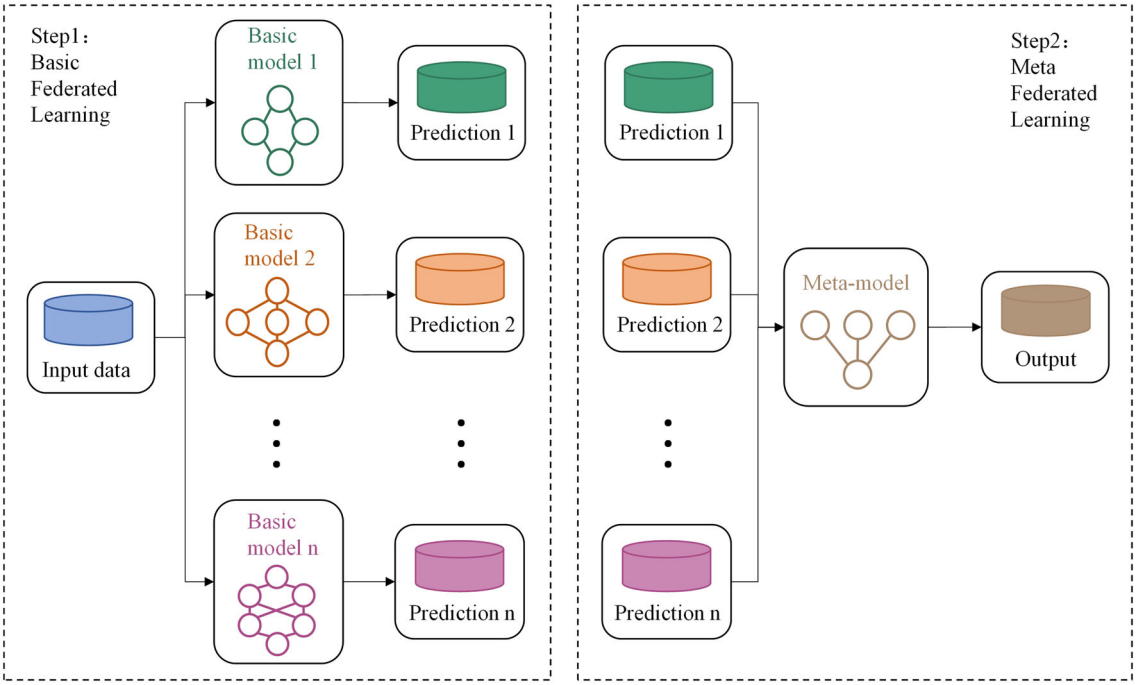

Fig. 1 Algorithm structure. Basic federated learning trains basic models, the predictions of which are used as the input of the meta-model. Meta-federated learning trains meta-model to obtain the output

Algorithm 1 Client's local update.

Require: Epochs, E; Batchs, B; Learning rate, η; Old global state, ω; Optimizer, $Adam$;
Ensure: new global state: ω;
function UPDATE(E, B, η, ω)
  for $i = 1 \rightarrow E$ **do**
    for $b \in B$ **do**
      $\omega \leftarrow$ ADAM(ω, b, η)
    end for
  end for
  return ω
end function

need to negotiate the total number of communication rounds R, the number of local training times E for each round, and the size of the dataset B for each training in advance. Each round, the clients participating in the training download the current global model from the server and then use the local data for training. After training, the clients upload the updated local model to the server. The server gathers the models and takes the weighted average according to the data magnitude as the global model for the next round. Algorithm 1 shows this process.

Basic federated Learning

The first step is basic federal learning. In this step, the basic models is trained by cross-validation, and the prediction results of the validation set are combined as the training data for the next step. This step will be elaborated next.

Basic model training

First, select some models as basic models. Suppose that n different basic models are adopted, and their network structures are different from each other. Each basic model can complete the tasks of training and prediction independently. For example, ResNet, VGG and other CNN networks are all advanced models, and each of them can complete the whole task of image classification. However, traditional deep learning models often perform unsatisfactorily in federated learning with non-IID datasets. Therefore, this algorithm utilizes them as basic models and uses a meta-model to aggregate them.

It is assumed that m clients are taking part in the training. The dataset owned by the client i is CD_i. The dataset CD_i is contained in the training model means that the client i participates in the training model. When the client i participates in training a federated learning model, it uses its local dataset CD_i to train the local model and uploads the model update to the server. The server uses the FedAvg algorithm mentioned in the previous section for aggregation. The process of updating the local model by the client in federated learning is shown in Algorithm 1.

If all clients participate in the training simultaneously, the generalization of the meta-model will be greatly reduced.

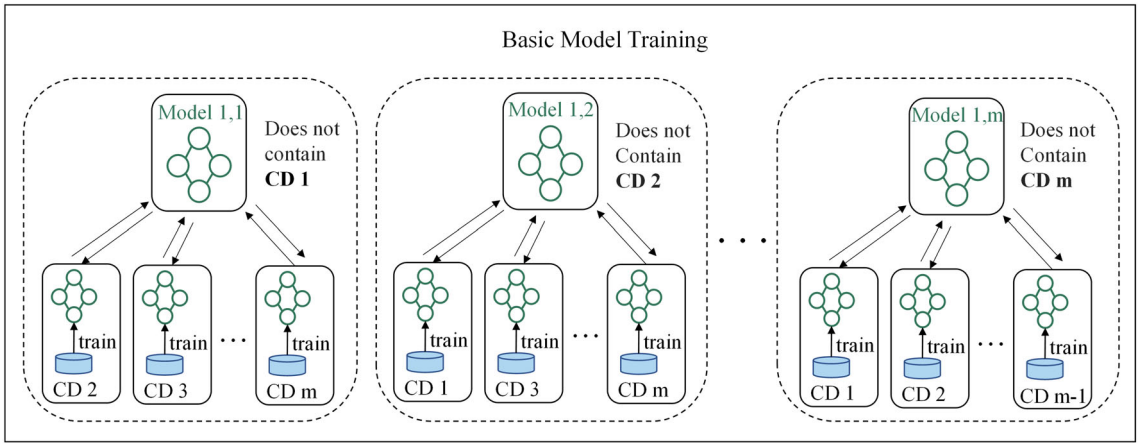

Fig. 2 Basic model training. Each basic model needs to train m sub models, and m is the number of clients. For each sub model, $m-1$ clients participate in federated learning, and CD_i represents the dataset of client i participates in training

Federated learning provides a natural dataset division. Therefore, when training basic models, this algorithm adopts the method of cross-validation. Each basic model must be trained to generate m sub models with the same structure but different parameters. The only difference in training these models is the datasets contained in training. Figure 2 shows the training method of one of the basic models. To achieve cross-validation, each basic model needs to train m sub models. For example, the sub models of basic model 1 are $model_{1,1}$, $model_{1,2}$, ... $model_{1,m}$, respectively. Each basic model is trained by federated learning participated by $m-1$ clients. $Model_{i,j}$ indicates that the structure of this model is the same as the basic model i, and CD_j does not be contained in training. In the end, $n \times m$ basic models can be generated.

During the basic model training, each client needs to train $n \times (m-1)$ models. When training a single model, each client needs to wait for other clients to upload local updates. The client cannot download the new global model and start the next round of training until the server completes the aggregation. In order to reduce the waiting time, this algorithm allows the client to sustain multiple models simultaneously while training other models during the wait time. Among these models, some have the same structure, and some have overlapping training sets. However, they are independent of each other. That is, these models are not interdependent or ordered. Therefore, the client can train these models in parallel. After each round of updating all models, it will wait for the aggregated results of the server. The details are shown in Algorithm 2.

Algorithm 2 Parallel training.

Require: Communication rounds, R; Basic models number, n; Client number, m; Current client, k; Epochs, E; Batchs, B; Learning rate, η; Local data, X;
Ensure: Basic model, $Model_{i,j}$, $1 \le i \le n$, $1 \le j \le m$; Meta model, $Model_{meta}$;
function TRAINING($R, n, m, k, E, B, \eta, X$)
  TRAINBASICMODEL(R, n, m, k, E, B, η)
  PREDICTBASICMODEL(X, k)
  TRAINMETAMODEL(R, E, B, η)
end function
function TRAINBASICMODEL(R, n, m, k, E, B, η)
  for $r = 1 \rightarrow R$ **do**
    for $i = 1 \rightarrow n$ **do**
      for $j = 1 \rightarrow m$ **do**
        if $j \neq k$ **then**
          Get $\omega_{i,j}$ from server
          $\omega_{i,j} \leftarrow$ UPDATE($E, B, \eta, \omega_{i,j}$)
          Upload $\omega_{i,j}$ to server
        end if
      end for
    end for
  end for
end function
function PREDICTBASICMODEL(X, k)
  for $i = 1 to n$ **do**
    Get $Model_{i,k}$ from server
    $Prediction_{i,k} = Model_{i,k}(X)$
    Upload $Prediction_{i,k}$ to server
  end for
end function
function TRAINMETAMODEL(R, E, B, η)
  for $r = 1 \rightarrow R$ **do**
    Get ω_{meta} from server
    $\omega_{meta} \leftarrow$ UPDATE($E, B, \eta, \omega_{meta}$)
    Upload ω_{meta} to server
  end for
end function

Basic model prediction

After training the sub models of each basic model, we need to use these sub models to predict the verification set. These predictions will be used as the training set of the meta-model. Figure 3 shows the prediction of the basic model and the combination of results.

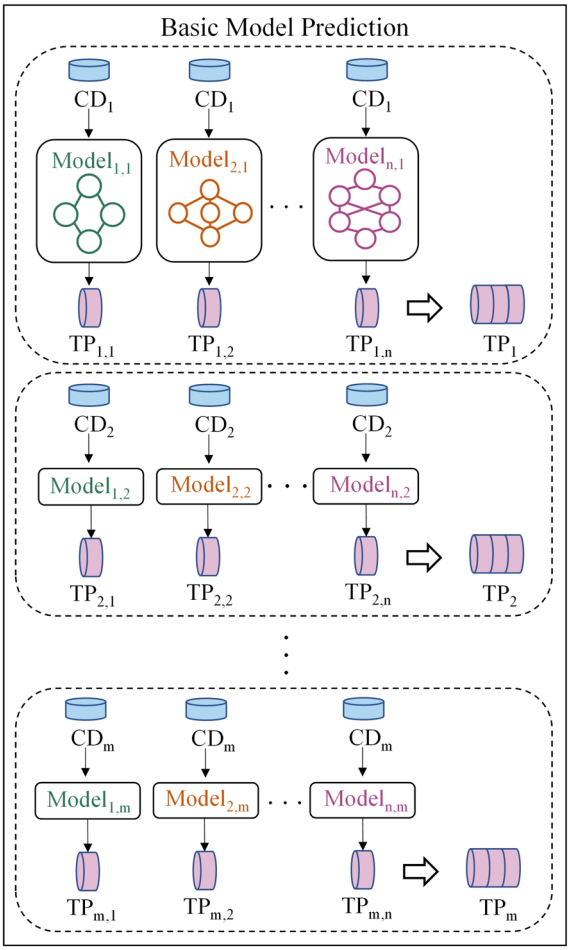

Fig. 3 Basic model prediction. Each client inputs data into trained sub models. The data CD_j of client j are input into $Model_{i,j}$ to obtain the prediction $TP_{i,j}$ for training. Each client combines its own predictions as the training data for meta federated learning (refer Sect. "Meta-federated learning")

A total of $n \times m$ basic models was generated in the previous training process, and each model has one client that does not participate in the training. Therefore, the algorithm uses the client's dataset not contained in training to make predictions. Assuming that the current basic model is $Model_{i,j}$, then the dataset CD_j of client j does not participate in training. Therefore, use CD_j for prediction and get the prediction $TP_{i,j}$ for training. This process is executed locally on the client j, and $TP_{i,j}$ is also stored locally.

At this stage, each client uses n models to make predictions on its own dataset. The client j inputs its dataset CD_j to $Model_{1,j}$, $Model_{2,j}$, ..., $Model_{n,j}$ to predict, which will get the prediction result $TP_{1,j}, TP_{2,j}, \ldots, TP_{n,j}$. The client j combines these predictions locally to form a new dataset TP_j, where each prediction is used as a feature. Finally, m clients have their own new training sets, $TP_1, TP_2, \ldots, TP_m$.

It is worth noting that to preserve the information of the prediction results output by basic models to a greater generalization, the probability of each category is used as the prediction result of basic models in this algorithm, instead of the final classification result. For example, the target of a basic model is ten classifications, and then its prediction result is a 10-dimensional vector. Therefore, the outputs of the n basic models are combined into a $10 \times n$-dimensional vector as a row in the dataset.

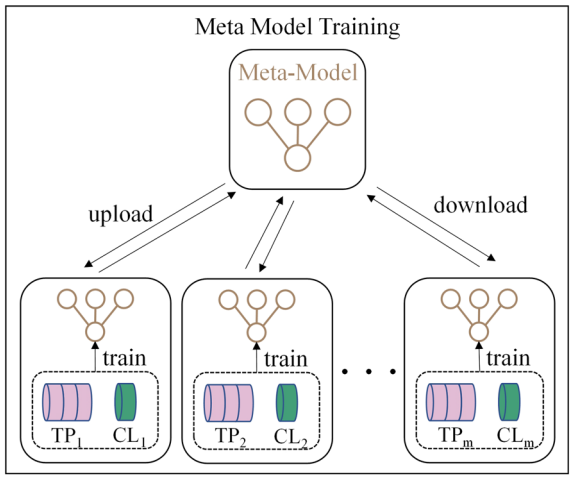

Fig. 4 Meta model training. Predictions $TP_1, TP_2, \ldots, TP_m$ for training (refer Sect. "Basic model prediction") and corresponding labels $CL_1, CL_2, \ldots, CL_m$ are combined as the training set of meta-model training

Meta-federated learning

Meta-federated learning is to train a meta-model using the training set prepared before. Figure 4 shows the process of meta-training. In the previous section, each client generated a new dataset TP_j, which was integrated by the outputs of basic model $Model_{1,j}, Model_{2,j} \ldots Model_{n,j}$. The client only needs to label the new dataset with the original label CL_j in CD_j, and it can be used as a complete dataset for training the meta-model.

The meta-model usually adopts a simple linear regression model. The input of the meta-model is a $10 \times n$-dimensional vector, that is, the output of n basic models. The output is the final classification category. The training process is still defined as federated learning. Each client trains independently with the locally generated dataset to update the local model and then uploads it to the server for aggregation. Finally, a trained meta-model is obtained.

Prediction

When new data are available, it is necessary to use the $n \times m$ basic models and meta-models obtained during training to predict the classification results. The prediction process

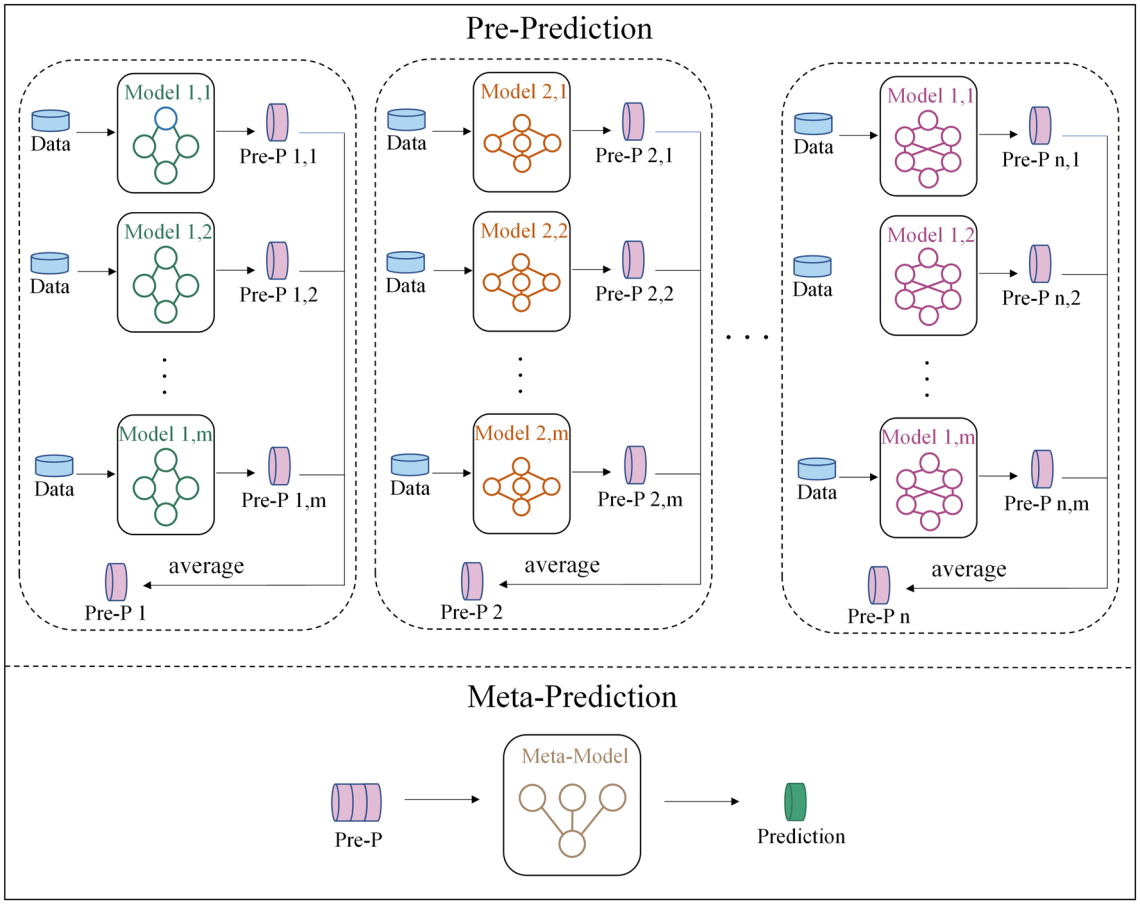

Fig. 5 Prediction includes two steps: pre-prediction and meta-prediction. Predict $Pre\text{-}P_{i,j}$ obtained from $Model_{i,j}$. The predict of the same basic model are combined as a dimension of the meta-prediction input

is shown in Fig. 5, which is divided into two steps: pre-prediction and meta-prediction.

For pre-prediction, first, input the data into the basic model separately to get the pre-prediction result. For example, $Model_{i,j}$ gets prediction result $Pre\text{-}P_{i,j}$. It can be seen from the settings during training that $Model_{i,1}$, $Model_{i,2}$, $\ldots$, $Model_{i,m}$ all have the same structure, but the training data are different, so pre-prediction average their prediction results to get $Pre\text{-}P_i$. Finally, combine these averages $Pre\text{-}P_1$, $Pre\text{-}P_2$, $\ldots$, $Pre\text{-}P_n$, each of which as a one-dimensional feature, to generate $Pre\text{-}P$.

For meta-prediction, input the $Pre\text{-}P$ generated by the pre-prediction into the meta-model to get the final prediction result.

Experiment

This section will first introduce the preparations for the experiment, including datasets, partitioning methods, and models. The experimental results will then be presented and analyzed in detail.

Preparation

In the experiment, different datasets are selected, and different datasets are divided in different ways. At the same time, several different models are selected for aggregation and comparison. Table 1 shows the algorithm, datasets, parameters and evaluation indicators in relevant papers. This section details the preparation of the experiment according to the datasets, parameters and evaluation indicators in these papers.

Datasets

The algorithm in this paper is more suitable for more complex classification models, so this paper selects two image classification datasets, including MNIST [12] and CIFAR-10 [13]. The MNIST is a large database of handwritten digits that is commonly used for training various image processing

Table 1 The algorithm, datasets, parameters and evaluation indicators in relevant papers

Algorithm	Datasets	Parameters	Evaluation indicators
FedAvg [14]	MNIST, CIFAR-10 Shakespeare	Adjustable: epoch, batch size, learning rate, client fraction	Accuracy, loss
FedProx [15]	MNIST, FEMNIST, Shakespeare, Sent140	Adjustable: μ, stragglers Fixed: epoch, batch size, learning rate	Loss
FedPD [20]	FEMNIST	Adjustable: R, p Fixed: epoch, batch size, learning rate	Gradient
FedBN [21]	SVHN, USPS, SynthDigits, MNIST-M, MNIST	Adjustable: epoch Fixed: batch size, learning rate	Accuracy, loss

systems., which contains 60000 training images and 10000 test images. All images in MNIST are 28 * 28 gray images. The CIFAR-10 dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. CIFAR-10 contains 10 different categories of 32 * 32 RGB images, each of which has 6000 for training and 1000 for testing.

Division methods

Federated learning needs to divide datasets into different clients. This paper mainly consider the division method of label shift, that is, the label distribution of different clients is different. In order to realize the dependent identically distributed dataset of different clients, this paper selects the traditional image classification dataset, and controls the degree of non-IID dataset through different dataset division methods. In the experiment, two different methods were selected, one is based on category and the other is based on Dirichlet distribution.

This paper uses category-based classification for MNIST dataset. This division allows different clients to have all data of different categories. For example, there are 10 images with different numbers in the MNIST dataset. In the experiment, five clients were asked to select all images with two different numbers, so that the local data of each client are different. Such division brings great challenges to federal learning.

The other way is based on Dirichlet distribution. In the experiment, random sampling is conducted in the Dirichlet distribution to determine the proportion of each category divided into different clients. In the case of five clients, the proportion of a category divided into different clients is $\boldsymbol{X} = (x_1, x_2, x_3, x_4, x_5)$. Let X obey the Dirichlet distribution, that is $\boldsymbol{X} \sim Dir(\alpha)$. The parameter α determines the degree of non-IID. The larger the α, the closer to uniform distribution. In the experiment, CIFAR-10 is divided based on Dirichlet distribution. Different α is selected for CIFAR-10 for experiment.

Models

In fact, the method proposed in this paper can be used for various depth learning models, but several common CNN models are selected as the basic models in the experiment to verify the effectiveness of our method. In the experiment, ResNet and VGG are mainly used as the basic models. By adjusting different datasets, partition methods and super parameters, it is compared with federated learning that only uses a single model. In addition, DenseNet was added to the experiment to test the selection of three basic models in detail.

Parameters and evaluation indicators

In the federal learning involved in the next experiment, adam is used as the optimizer and cross entropy as the loss function. In the comparative experiment, the learning rate is optimized, respectively. Refer to the parameters used in other papers in Table 1, and remove the parameters specific to other methods, this experiment discuss several situations where the batch size is 128 or 256 and the local training epoch is 1 or 3. In the experiment of adding DenseNet, the batch size is set to 128, and the local training epoch is set to 20.

The evaluation indicators of federated learning algorithm include learning performance, communication efficiency, and incentive mechanism. Our algorithm improves the performance of federated learning in non-IID datasets, so we mainly consider the indicators related to learning performance. Table 1 shows the indicators about learning performance commonly used in federated learning algorithms.

On the basis of these indicators, this paper adds some indicators to show the characteristics of our algorithm. The accuracy is the main indicator for us to measure the performance of the model, including the accuracy in the training set and the test set. For classification problems, F1-Score combined with accuracy and recall is also an important criterion. In order to explain how our algorithm makes up for the defects of different models and combines their advantages, we also use the confusion matrix to clearly show the classification results of different models.

Results

This section shows the experimental results based on the above settings. The experiment includes two parts: basic federal learning and meta federal learning. This section first analyzes the detailed result of training based on the use of three basic models, and then uses different training parameters in different datasets to compare with traditional federated learning.

Detailed explanation

This section explored the performance of the method proposed in this paper when selecting three basic models in detail. ResNet, VGG, and DenseNet are chosen as basic models in the experiment. In basic model training, each model is trained by four clients, and finally, 15 basic models are generated. The basic models trained by ResNet, VGG and DenseNet are $Model_{1,j}$, $Model_{2,j}$ and $Model_{3,j}$, respectively.

The global test set is used for each trained model for testing. The results are shown in Table 2. Fold-j indicates that client j did not participate in the training model in this table. It can be seen that the non-IID division method makes the datasets between different clients vary greatly. Therefore, different models that different clients do not participate in training, have completely different performance. For example, ResNet's Fold-1 model has an accuracy of 76.86%, which is even higher than 75.13% of all clients. This shows that the data of client 1 have a negative impact on the ResNet-based federated learning. Therefore, our method adopts cross-validation to reduce this negative impact.

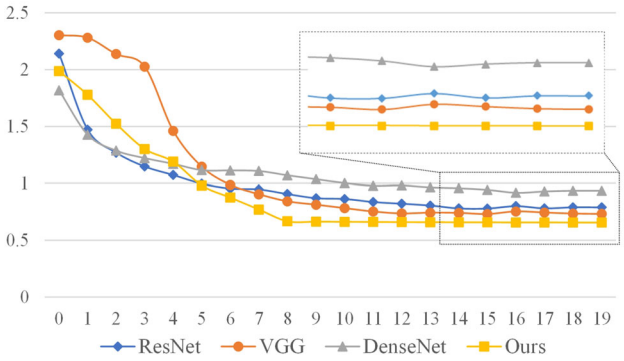

Fig. 6 The Comparison of training loss. ResNet, VGG and DenseNet use the loss of the global model in federated learning, and our method uses the loss of the global model in meta federated learning

Each client uses its local data to make predictions for the models that have not participated in the training and then uses the prediction results as the input of the meta-model training set. Finally, it combines the training set with original data labels to train the meta-model. Figure 6 shows the training loss comparison curves of different methods. It can be seen that our method can eventually converge to a smaller loss.

Finally, basic models and the meta-model obtained by training are integrated as a whole model. The training set and test set in CIFAR-10 are used to test the model's classification accuracy, and the weighted F1-Score is calculated in the test set. The final result is shown in Table 3. ResNet, VGG, and DenseNet are the three basic models used in this experiment. The classification accuracy of the algorithm in this paper on the training and test set reaches 86.12% and 78.06%, respectively. Compared with the traditional federated learning using the three basic models of ResNet, VGG, and DenseNet alone, the classification accuracy of this algorithm on the training set is increased by 5.37%, 3.05%, and 4.93%, respectively, and is improved to 10.99%, 8.39% and 9.45% on the test set. In terms of F1-Score, this algorithm achieves 85.96%, which is 11.79%, 11.17%, and 11.23% higher than the three basic models, respectively. It can be seen that after using this algorithm, the performance of the model is significantly improved compared to the traditional federated learning using the three basic models alone, and it also has better generalization.

Table 2 Accuracy of 5-fold cross-validation

Model	Fold-1[a] (%)	Fold-2 (%)	Fold-3 (%)	Fold-4 (%)	Fold-5 (%)	Non-Fold[b] (%)
ResNet	76.86	71.20	70.48	73.00	69.96	75.13
VGG	81.48	76.06	80.05	75.89	73.00	77.73
DenseNet	70.00	61.73	67.91	65.39	69.61	76.67

[a]Fold-j indicates that client j did not participate in the training model
[b]Non-Fold indicates that all clients participate in federated learning

Table 3 Performance comparison of different methods

Model	Train/acc (%)	Test/acc (%)	Test/F1-score (%)
Ours	**78.06**	**86.12**	**85.96**
ResNet	72.69	75.13	74.17
VGG	75.01	77.73	74.79
DenseNet	73.13	76.67	75.73

The items to be compared include the classification accuracy of the test set and training set, and the F1-score of the test set
Bold values emphasize the performance achieved by our algorithm

Figure 7 shows the confusion matrices of the prediction results of the three basic models and the algorithm in this paper on the test set, respectively. It can be seen that this algorithm uses the meta-model to integrate the advantages of the three basic models and achieves better prediction performance. For example, although VGG has high overall accuracy, most cats are classified as dogs, and the other two models make up for this shortcoming. Therefore, our algorithm can achieve better performance in cat classification. The same applies to other categories.

Experiment of different training parameters

Tables 4 and 5 show the accuracy of our algorithm on different datasets when five clients participate in training, compared with FedAvg and FedProx. FedAvg and FedProx uses ResNet and VGG models for testing, and the method proposed in this paper uses both models as the basic models. In the experiment, different batch sizes and epochs of local training were selected. For MNIST, the experiment adopts the classification based method. For CIFAR-10, the experiment adopts the division method based on Dirichlet distribution, and adjusts different parameters.

It can be seen from Tables 4 and 5 that for different datasets and different division methods, the accuracy of the method proposed in this paper is improved compared with FedAvg and FedProx. Different training parameters were selected in the experiment. It is worth noting that the batch and epoch in the parameters refer to the batch size and epoch when the client is training locally. When datasets and partitions are different, the training parameters for obtaining the highest accuracy will also be different. When using MNIST dataset, the model with batch = 256 and epoch = 1 has the highest accuracy. When using the CIFAR-10 dataset, the model with the highest accuracy depends on the partition parameter α. When $\alpha = 0.5$, the model with batch=256 and epoch=3 performs better, and When $\alpha = 1.0$ or $\alpha = 5.0$, the model with batch = 128 and epoch = 3 has better performance.

The above experiments verify that the federated learning algorithm based on parallel-ensemble learning described in this paper outperforms the traditional federated learning algorithm without parallel-ensemble learning on the image classification task.

Table 4 The accuracy of our algorithm and other methods on MNIST

Batch[a]	Epoch[b]	Method	Model	Accuracy (%)
128	1	FedAvg	ResNet	85.56
			VGG	83.41
		FedProx	ResNet	85.92
			VGG	89.51
		Ours	ResNet&VGG	**97.46**
	3	FedAvg	ResNet	81.96
			VGG	68.99
		FedProx	ResNet	83.62
			VGG	88.42
		Ours	ResNet&VGG	**96.88**
256	1	FedAvg	ResNet	83.92
			VGG	83.06
		FedProx	ResNet	86.18
			VGG	95.06
		Ours	ResNet&VGG	**97.51**
	3	FedAvg	ResNet	83.35
			VGG	62.18
		FedProx	ResNet	89.17
			VGG	89.05
		Ours	ResNet&VGG	**95.85**

Bold values emphasize the performance achieved by our algorithm
[a]Batch refers to the batch size of the client during local training
[b]Epoch refers to the epoch of the client during local training

Feasibility analysis

Practically, the data between different devices shows different distribution. When using these non-IID datasets for federated learning of a single model, it often fails to achieve good performance. However, our algorithm uses meta-model to fuse multiple basic models, combining the advantages of different models, and can perform better in non-IID datasets.

Generally, models that can achieve the same abilities but have different structures are selected as the basic models. These models can be used independently, but in our algorithm, they can be trained at the same time to make up for the defects between different structures. The meta-model usually uses a simple linear regression model to reduce over-fitting.

In addition, our algorithm uses parallel training to reduce the time cost of training multiple models. At the same time, when training each specific model, our algorithm adopts the federated learning structure, which can protect privacy and data security in a limited communication time.

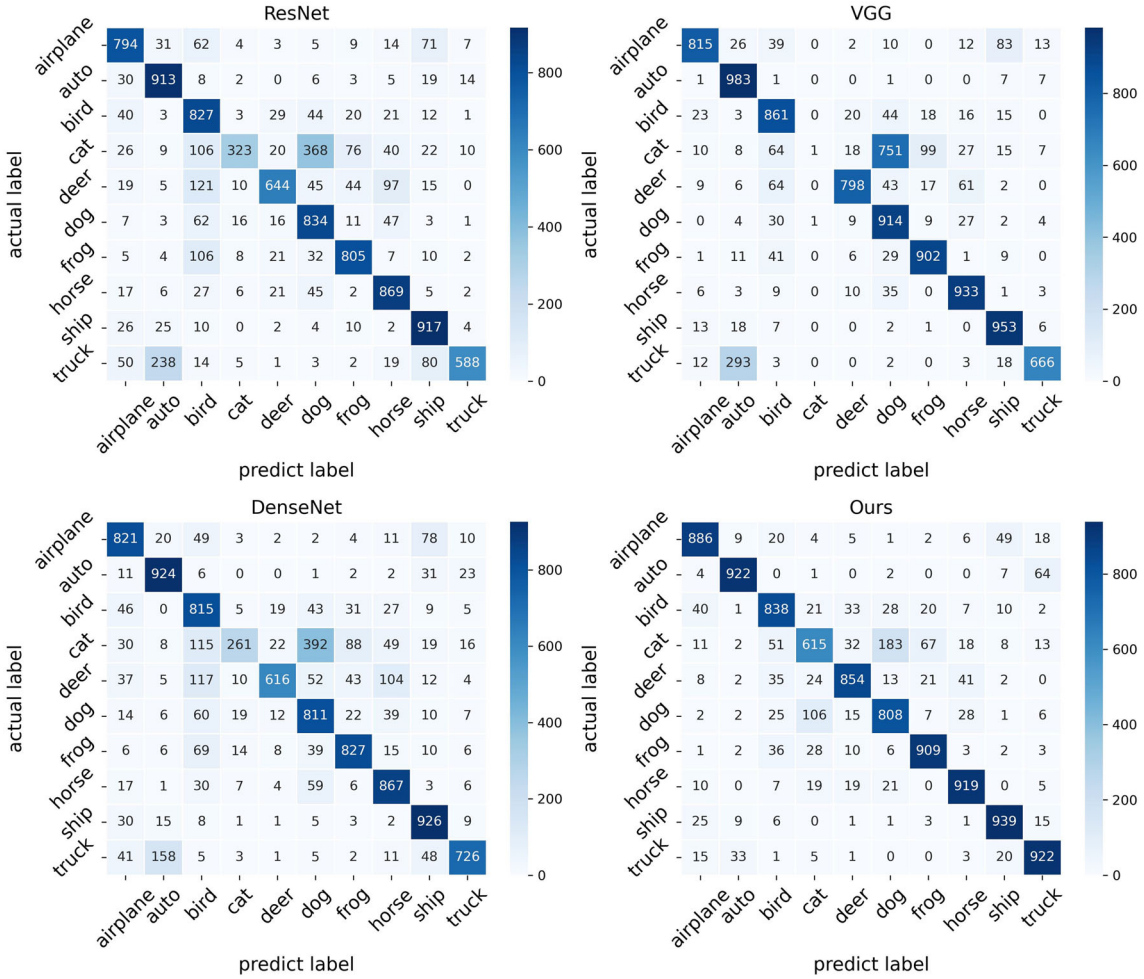

Fig. 7 The confusion matrix of Resnet, VGG, DenseNet and our methods on test set

Summarize

This section will summarize the whole article, state the conclusions and look forward to the future.

Conclusion

As a distributed machine learning technology, Federated learning can perform federated learning while protecting privacy. However, different edge devices have different datasets and often follow different distributions. Compared with traditional machine learning, the performance of the federated learning aggregation algorithm under the non-IID dataset is not ideal. In the experiments of this paper, the three deep learning models of ResNet, VGG, and DenseNet perform far worse than the performance in the IID dataset.

The algorithm proposed in this paper combines federated learning and ensemble learning and uses a meta-model to combine the prediction results of several basic models to achieve the effect of learning from each other's strengths. At the same time, this algorithm utilizes the characteristics of federated learning to implement a parallel cross-validation algorithm. Each client maintains multiple cross-trained models simultaneously, reducing the waiting time for each round in federated learning. In the experiments of this paper, the algorithm proposed in this paper has a significant improvement in classification accuracy and F1-Score compared to using the deep learning model alone for federated learning. At the same time, it can be seen from the confusion matrix that this algorithm overcomes the defect that the datasets of different clients are not independent and identical to a certain extent and makes up for the weakness of a single deep learning model in the classification on the non-IID dataset.

Future work

In addition to image classification, federated learning is used in a wide range of fields. The problem of dataset non-IID appears in several federated learning settings. In the future, it is hoped that the algorithm proposed in this paper can be

Table 5 The accuracy of our algorithm and other methods on CIFAR-10

Batch[a]	Epoch[b]	Method	Model	Accuracy (0.5)[c] (%)	Accuracy (1.0) (%)	Accuracy (5.0) (%)
128	1	FedAvg	ResNet	74.75	63.91	66.28
			VGG	81.38	67.96	73.09
		FedProx	ResNet	75.45	73.75	74.29
			VGG	84.46	74.32	77.62
		Ours	ResNet&VGG	**85.36**	**74.74**	**79.17**
	3	FedAvg	ResNet	72.96	61.81	65.51
			VGG	80.91	67.13	75.18
		FedProx	ResNet	76.24	73.41	75.36
			VGG	83.13	73.93	79.94
		Ours	ResNet&VGG	**86.25**	**75.39**	**80.59**
256	1	FedAvg	ResNet	72.97	60.11	65.67
			VGG	81.44	64.44	73.51
		FedProx	ResNet	73.91	74.36	74.23
			VGG	84.37	72.11	75.17
		Ours	ResNet&VGG	**84.59**	**71.74**	**78.71**
	3	FedAvg	ResNet	72.23	59.12	65.78
			VGG	81.55	67.55	75.93
		FedProx	ResNet	75.67	71.86	76.35
			VGG	76.17	73.25	75.01
		Ours	ResNet&VGG	**86.55**	**75.87**	**78.43**

Bold values emphasize the performance achieved by our algorithm
[a]Batch refers to the batch size of the client during local training
[b]Epoch refers to the epoch of the client during local training
[c]The number in parentheses after "Accuracy" indicates the α used when using the division method based on Dirichlet distribution

applied to more federated learning scenarios other than image classification to solve more practical problems and reduce the impact of imbalanced data distribution. Furthermore, it is envisaged that this algorithm is suitable for deep learning models and can be applied to traditional machine learning models to improve their performance.

Compared with the traditional federated learning algorithm, the algorithm proposed in this paper needs to train more models. Although the parallel-ensemble learning method is used in the algorithm, the training efficiency is still lower than the traditional federated learning. Therefore, it is hoped that the efficiency of training can be further improved while ensuring the performance.

Data Availability The data that support the findings of this study are available from the corresponding author, Chang Wu, upon reasonable request.

References

1. Konečnỳ J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492
2. Wu Q, He K, Chen X (2020) Personalized federated learning for intelligent IoT applications: a cloud-edge based framework. IEEE Open J Comput Soc 1:35–44
3. Li Q, Diao Y, Chen Q, He B (2021) Federated learning on non-iid data silos: an experimental study. arXiv preprint arXiv:2102.02079
4. Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V (2018) Federated learning with non-iid data. arXiv preprint arXiv:1806.00582
5. Zhuang Z, Tao H, Chen Y, Stojanovic V, Paszke W (2022) An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints. IEEE Trans Syst Man Cybern Syst
6. Djordjevic V, Stojanovic V, Tao H, Song X, He S, Gao W (2022) Data-driven control of hydraulic servo actuator based on adaptive dynamic programming. Discrete Contin Dyn Syst Ser S 15(7)
7. Zhou C, Tao H, Chen Y, Stojanovic V, Paszke W (2022) Robust point-to-point iterative learning control for constrained systems:

a minimum energy approach. Int J Robust Nonlinear Control 32(18):10139–10161
8. Wolpert DH (1992) Stacked generalization. Neural Netw 5(2):241–259
9. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
10. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
11. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
12. Deng L (2012) The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Mag 29(6):141–142
13. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Handb Syst Autoimmune Dis 1(4)
14. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR, pp 1273–1282
15. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V (2020) Federated optimization in heterogeneous networks. Proc Mach Learn Syst 2:429–450
16. Xiao Z, Xu X, Xing H, Song F, Wang X, Zhao B (2021) A federated learning system with enhanced feature extraction for human activity recognition. Knowl Based Syst 229:107338
17. Zhang W, Li X, Ma H, Luo Z, Li X (2021) Federated learning for machinery fault diagnosis with dynamic validation and self-supervision. Knowl Based Syst 213:106679
18. Zhu H, Zhang H, Jin Y (2021) From federated learning to federated neural architecture search: a survey. Complex Intell Syst 7:639–657
19. Gao L, Fu H, Li L, Chen Y, Xu M, Xu C-Z (2022) Feddc: federated learning with non-iid data via local drift decoupling and correction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10112–10121
20. Zhang X, Hong M, Dhople S, Yin W, Liu Y (2020) Fedpd: a federated learning framework with optimal rates and adaptivity to non-iid data. arXiv preprint arXiv:2005.11418
21. Li X, Jiang M, Zhang X, Kamp M, Dou Q (2021) Fedbn: federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623
22. Kulkarni V, Kulkarni M, Pant A (2020) Survey of personalization techniques for federated learning. In: 2020 fourth world conference on smart trends in systems, security and sustainability (WorldS4). IEEE, pp 794–797
23. Ma X, Zhang J, Guo S, Xu W (2022) Layer-wised model aggregation for personalized federated learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10092–10101
24. Shin M, Hwang C, Kim J, Park J, Bennis M, Kim S-L (2020) Xor mixup: privacy-preserving data augmentation for one-shot federated learning. arXiv preprint arXiv:2006.05148
25. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
26. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
27. Lin S, Yang G, Zhang J (2020) A collaborative learning framework via federated meta-learning. In: 2020 IEEE 40th international conference on distributed computing systems (ICDCS). IEEE, pp 289–299
28. Hamer J, Mohri M, Suresh AT (2020) Fedboost: a communication-efficient algorithm for federated learning. In: International conference on machine learning. PMLR, pp 3973–3983
29. Liu Y, Liu Y, Liu Z, Liang Y, Meng C, Zhang J, Zheng Y (2020) Federated forest. IEEE Trans Big Data
30. Li Q, Wen Z, He B (2020) Practical federated gradient boosting decision trees. Proc AAAI Conf Artif Intell 34:4642–4649