IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 23 August 2023, accepted 4 September 2023, date of publication 7 September 2023, date of current version 14 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313001

RESEARCH ARTICLE

Heterogeneous Defect Prediction Based on Federated Prototype Learning

AILI WANG[1], **(Member, IEEE), LINLIN YANG**[1], **HAIBIN WU**[1], **AND YUJI IWAHORI**[2], **(Member, IEEE)**
[1]Heilongjiang Province Key Laboratory of Laser Spectroscopy Technology and Application, Harbin University of Science and Technology, Harbin 150080, China
[2]Department of Computer Science, Chubu University, Kasugai, Aichi 487-8501, Japan

Corresponding author: Haibin Wu (woo@hrbust.edu.cn)

This work was supported in part by the Reserved Leaders of Heilongjiang Provincial Leading Talent Echelon of 2021, in part by the High and Foreign Expert's Introduction Program under Grant G2022012010L, and in part by the Key Research and Development Program Guidance Project under Grant GZ20220123.

ABSTRACT Software defect prediction is used to identify modules in software projects that may have defects. Heterogeneous Defect Prediction (HDP) establishes a cross project defect prediction model based on different software defect datasets. However, due to the heterogeneity of multi-source data, the model performance is usually not ideal. In addition, the project data holder is unwilling to disclose the data due to privacy regulations and other reasons, resulting in data islands. This paper presents a federal prototype learning based on prototype averaging (FPLPA), which combines federated learning (FL) with prototype learning for heterogeneous defect prediction. Firstly, the client used one-sided selection (OSS) algorithm to remove noise from local training data, and applied Chi-Squares Test algorithm to select the optimal subset of features. Secondly, the client constructed the convolution prototype network (CPN) to generate their own local prototypes. CPN are more robust to heterogeneous data than convolutional neural networks (CNN), while avoiding the deviation effect of class imbalances in software data. The prototype is used as the communication subject between the clients and the server. Because the local prototype is generated in an irreversible way, it can play a role of privacy protection in the communication process. Finally, the local CPN network is updated with the loss of local prototype and global prototype as regularization. We have verified on 10 projects in three public data sets (AEEEM, NASA and Relink), and the experimental results show that FPLPA is superior to other HDP solutions.

INDEX TERMS Heterogeneous defect prediction, federated learning, prototype learning, data islands.

I. INTRODUCTION

Software systems have been widely used in various areas of life. A software defect is commonly defined as a deviation between the programmed results and requirements, which can lead to serious consequences [1], [2], [3], [4]. In the development life cycle of a software project, the later the internal defects are detected, the higher cost of repairing the defects. Therefore, the project leader hopes to detect as many inherent defects as possible before software deployment by means of software quality assurance such as software testing or code review. But if we focus on all the program modules, it will consume a lot of manpower and material resources. The software quality assurance department hopes to identify program modules with potential defects as early as possible, and then allocate sufficient testing resources to them.

Software defect prediction is one of the feasible methods. According to the historical software development data and the defects found, the potential defective program modules in the software project can be predicted by means of machine learning and other methods. Modern software development systems include many programming languages and measurement element methods, which are vast and complex. Common development languages such as C, C++ and JAVA

The associate editor coordinating the review of this manuscript and approving it for publication was Chengpeng Hao.

are used. Metric elements include Line of Code (LOC), McCabe, Halstead and so on [5], [6], which can view software components from multiple perspectives.

Current software defect prediction can be divided into Within-Project Defect Prediction (WPDP) and Cross-Project Defect Prediction (CPDP) [7], [8], [9], [10], [11]. Software defect prediction within a project refers to building a model and forecasting on the limited data of the same project.

Many metric modes make it very complex to build software defect prediction models. Choosing a quantum set of metrics to improve the performance of software defect prediction methods is a challenging problem. On the other hand, a large amount of training data is necessary to establish a good defect prediction model [12], [13]. In reality, it is difficult to collect enough training data, and most historical software defect data sets are unbalanced [14], [15], [16]. The trained model will over fit, resulting in lack of generalization performance in other projects. Therefore, the CPDP method has been proposed to build a software defect prediction model based on the historical data of other projects (i.e. source projects), and then conduct defect prediction on the current project (i.e. target projects) [17], [18], [19].

Some studies show that building software defect prediction models across projects can indeed improve the generalization performance of models, including using subspace learning and other methods. But the data distribution between the source project and the target project is very different in most cases. Heterogeneous defect prediction of data heterogeneity exists between the source project and the target project [20], [21], [22], [23], [24]. This leads to the prediction model built on the source project is often difficult to have a satisfactory prediction effect on the target project. Due to industry competition, business privacy and other reasons, software enterprises are usually reluctant to share local data [25]. The training method of centralizing multiple project data in a single third-party node is difficult to achieve. The "data island" arises from this reason. Therefore, on the premise of meeting the increasingly strict provisions of the privacy regulatory authority, narrowing the data distribution difference between the source project and the target project and solving the data island problem in the software defect prediction field are the urgent problems that researchers are facing in designing CPDP prediction methods.

FL [26] is a training method that uses data sets distributed among multiple clients to build a global model cooperatively. FL integrates multi-party data information through homomorphic encryption, differential privacy and other privacy protection technologies. The client can exchange model parameters, model structure, and parameter gradients during the model training process. Data can be plaintext, data encryption, adding noise, etc. But the local training data will not leave the local participants. This exchange method will not expose local user data, reducing the risk of data leakage. The trained FL model can be shared and deployed among all data clients. Prototype learning [27] is a strategy that can eliminate data redundancy, discover the internal structure of data, and improve data quality. By finding a prototype set to reduce the data in the source sample space, prototype learning can enhance the data availability and improve the execution efficiency of the machine learning algorithm.

This paper proposes a federated prototype learning based on prototype averaging (FPLPA). In order to make the model have better generalization performance for heterogeneous data, we introduce prototype learning into FL. Firstly, FPLPA uses OSS to remove the noise of clients' local training data, then uses Chi-Squares Test to select the optimal feature (metric) subset, and finally generates class prototype through CPN. CPN retains and optimizes local data while learning heterogeneous data. FPLPA uses the class prototype generated by CPN as the communication agent of FL, rather than the gradient used in most FL architectures. Therefore, CPN can avoid the bias effect of unbalanced data on traditional convolutional networks. Prototypes can play a privacy protection role as communication entities, as the prototype data generated by CPN is irreversible. The server collects and averages local prototypes from clients to get global prototypes, and then sends the global prototype back to all clients. The client uses the global prototype to regularize the training of the local personalized model.

The main contributions of this paper can be summarized as follows:

1) We propose FPLPA by introducing federated prototype learning into the field of HDP which allows multiple clients to train personalized local models without disclosing local data, to solve the problem of data islands in the field of CPDP.
2) Local data is preprocessed using OSS which helps class prototypes better cover instances of the same class by reducing noise data and Chi-Squares Test is used to search for an optimal subset of features.
3) FPLPA can protect the privacy of local data without introducing noise, which prevents malicious attacks from obtaining sensitive information from the communication process.
4) We tested FPLPA performance on three public datasets, NASA, AEEEM and Relink. Experimental results show that FPLPA performs better than other advanced software defect methods.

The rest of this paper is organized as follows. Section II describes the work related to software defect prediction. Section III introduces the implementation principle of our proposed FPLPA algorithm in detail. Section IV gives the details of the experimental design and discusses our experimental results. The last section summarizes the conclusions and future work.

II. RELATED WORKS

A. SOFTWARE DEFECT PREDICTION METHODS

As mentioned earlier, software defect prediction is divided into WPDP and CPDP. The training data adopted by WPDP is the data information of software modules in the historical version of the project, The distribution of training data and test data is basically the same. To solve the metrics subset selection problem, Huda et al. [28] proposed two novel hybrid models which embed the metric selection and training processes as a single process. Different wrapper approaches are combined with support vector machines and artificial neural network to identify a subset of significant metrics.

Subsequently, Huda et al. [29] proposed a software defect prediction ensemble model. which used a combination of random oversampling, Majority Weighted Minority Oversampling Technique, and Fuzzy-Based Feature-Instance Recovery to build an ensemble classifier for minimizing the effect of imbalance distribution of classes in the training data. Zhao et al. [30] proposed Siamese dense neural networks, which integrates similarity feature learning and distance metric learning into a unified approach. Siamese networks are powerful for learning a few instances and have been perfectly used in other fields.

However, when it is necessary to predict the defects of a new project or a project lacking historical data, The method based on WPDP is no longer feasible. Researchers propose to use the historical data of other projects to build a CPDP classifier and predict new projects. Jing et al. introduce canonical correlation analysis (CCA) into CPDP to make the distributions of source and target datasets similar, and then proposed CCA+ [31] cross-project heterogeneous defect prediction method. The unified metric representation (UMR) is formed to alleviate the data heterogeneity between source and target projects. Gong et al. [32] mapped the data of source and target projects into a UMR and validated the effectiveness of this method on 18 common projects across 4 datasets. Nam et al. proposed TCA+ [33] which is an improvement of transfer component analysis (TCA) [34]. This method assumes that the source project and target project have the same feature set. The prediction performance of cross project model can be enhanced to some extent by improving the feature distribution similarity between source project and target project. Sun et al. [35] proposed a cross-project semi-supervised defect prediction method using a generative adversarial networks [36] called as Discriminating Opponent Feature Learning (DAFL). Ma et al. [37] proposed a method called as Kernel Canonical Correlation Analysis plus (KCCA+). Combining the kernel method and transfer learning techniques, this method improves the performance of the predictor with more adaptive ability.

The above research has made certain achievements in the selection of metrics set and the treatment of class imbalance, especially the pioneering cross-project research. However, more robust methods are still needed in scenarios with highly heterogeneous data.

B. FEDERATED LEARNING

The core concept of FL is data availability and invisibility which ensures that all clients can build a training model cooperatively without leaving the local client.

Federated averaging (FedAvg) [38] is one of the first federate learning frameworks proposed. This method obtains the global model by averaging model parameters from different clients. The recent research work based on FL mainly focuses on data heterogeneity, privacy protection and communication efficiency between clients and servers. FedProx [39] can be seen as the generalization and reconstruction of FedAvg. In the case of heterogeneity, FedProx shows more stable and accurate convergence than FedAvg. Li et al. [40] proposed using local batch normalization before averaging model parameters to reduce feature offset and solve the problem of relying on identically distributed features. Its performance is superior to classic FedAvg and FedProx.

In the FL framework, each client has its own sensitive data and model parameters. Once these parameters are used by the attacker, Privacy information will be exposed. At present, differential privacy and homomorphic encryption are the mainstream methods for privacy protection in FL. Madi et al. [41] proposed a secure framework relying on Homomorphic Encryption and Verifiable Computation, and conducted experiments on the FEMNIST dataset. Wang et al. [42] proposed a federated reinforcement learning method based on gradient clustering (FRLGC). FRLGC adds Gaussian noise to the data of local clients to achieve data level privacy protection.

However, the privacy protection methods based on homomorphic encryption or differential privacy will introduce noise. The noise of homomorphic multiplication will increase explosively as the number of calculations increases. This means that when the noise increases to a certain extent, ciphertext cannot be decrypted. Differential privacy directly adds noise to the gradient. It will inevitably reduce the upper performance limit and convergence speed of the model.

Some researchers believe that personalized local models can better adapt to the data distribution of each client. Therefore, many personalized FL frameworks based on adaptive technology have been proposed. Li et al. [43] proposed a novel self-adaptive federated learning framework in heterogeneous systems that adaptively selects clients with larger local training loss in each training round to accelerate the convergence of the global model. Chai et al. [44] proposed a tier-based FL system that groups clients into tiers based on training performance. This algorithm alleviates performance issues caused by data heterogeneity by optimizing accuracy and training time.

In terms of improving the communication efficiency between clients and servers, Model distillation can accelerate the convergence speed of local model and achieve more efficient update speed by reducing the parameters. Wang et al. [45] proposed a federated peer-to-peer network architecture, which called heterogeneous defect prediction based on federated transfer learning via knowledge distillation (FTLKD). Leveraging unlabeled auxiliary data (FEDAUX) [46] is an extended version of federated distillation. This method proposes to conduct

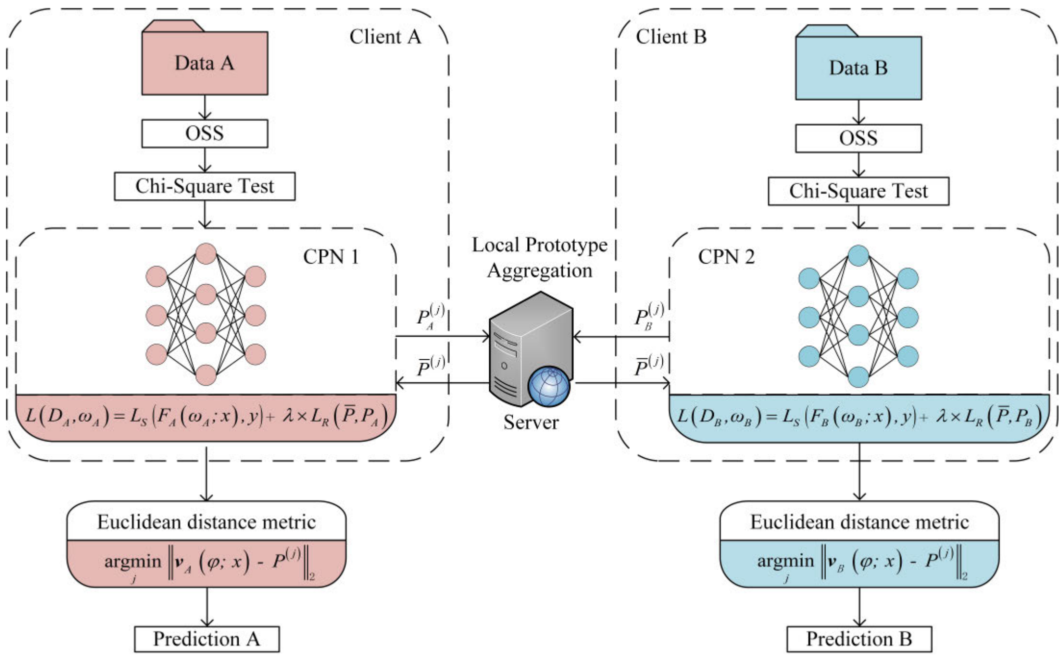
Client A
Data A
OSS
Chi-Square Test
CPN 1
$L(D_A,\omega_A)=L_S(F_A(\omega_A;x),y)+\lambda\times L_R(\bar{P},P_A)$
Client B
Data B
OSS
Chi-Square Test
CPN 2
$L(D_B,\omega_B)=L_S(F_B(\omega_B;x),y)+\lambda\times L_R(\bar{P},P_B)$
Local Prototype Aggregation
$P_A^{(j)}$
$\bar{P}^{(j)}$
$P_B^{(j)}$
$\bar{P}^{(j)}$
Server
Euclidean distance metric
$\operatorname{argmin}_j \|v_A(\varphi;x)-P^{(j)}\|_2$
Euclidean distance metric
$\operatorname{argmin}_j \|v_B(\varphi;x)-P^{(j)}\|_2$
Prediction A
Prediction B

FIGURE 1. **Overall framework of FPLPA. which is carried out under the framework of FL. It mainly includes data preprocessing, local model training, local prototype aggregation and personalized local model testing.**

unsupervised pre-training on auxiliary data to find an initialization model. FEDAUX shows excellent performance on convolutional neural networks and transformer models.

C. PROTOTYPE LEARNING

The CPDP method was originally proposed to solve the problem of insufficient effective data faced by WPDP. However, cross-project also brings challenges to heterogeneous data. Since prototype learning can generate new data with commonalities in the source domain. It maximizes the computational efficiency of FL and protects sensitive information of each client.

Prototype learning is often referred to as subset selection or core set construction. Suppose a source set $S = \{s_1, s_2, \cdots, s_m\}$ and a target set $T = \{t_1, t_2, \cdots, t_n\}$ contain m and n instances respectively. Prototype learning aims to find a prototype set $\Omega \subseteq S$ from the source set S, so that Ω can retain the information contained in the target set T to the greatest extent, and all instances in Ω have the least overlapping information.

Prototype learning can be regarded as a fine-grained clustering problem. From a local perspective, each element in the prototype set can contain public information to the greatest extent. It can replace target set to describe complex events. This is very beneficial for the communication phase of FL. Prototypes can be obtained in two ways. One way is to select representative data from the source set to form a prototype set, which is called the selection method. Another way is to regenerate a set of representative points in the source set, called the generation method.

The earliest prototype learning method is K-nearest-neighbor (K-NN) [47]. To reduce the storage space burden and computing requirements of K-NN, Kohonen proposed learning vector quantization [48], which is another nearest neighbor prototype classifier. It can be regarded as a two-layer neural network model, and each node in the output layer has a weight vector. By gradually converging the boundaries between weight vectors to the boundaries of Bayesian classification, the most representative prototypes can be found.

The representativeness of the prototype can be significantly improved by CNN [49]. Yang et al. [50] proposed robust classification with convolutional prototype learning, which is the first attempt to use a deep convolution network to generate prototypes. Elhamifar et al. [51] also proposed a supervised prototype selection framework. It combines deep representation learning with prototype selection, and effectively applies it to any video summarization task. Pang et al. proposed a unified framework for (DisP+V) [52] face recognition. Experiments on real world face data sets show that DisP+V model performs well. Zarei Sabzevar's builds a highly interpretable prototype [53]. The neuron's functionality is interpreted as calculating the distance difference between positive and negative prototypes, which successfully explains the function of the deep neural network.

Prototypes can help models learn better domain invariant features. Therefore, prototypes are also widely used in the field of multi-source heterogeneous domain adaptation. Tan et al. [54] applied prototype learning to FL for the first time. Zhou et al. [55] proposed a new prototype-based multisource domain adaptation. Through the inherent class prototype and domain prototype, the corresponding class prototype is constructed, and the semantic class information of the source domain is transformed into the target domain with no label. Li et al. proposed a multi-instance alignment model based on global class prototype [56]. The consistency of predictions is improved by minimizing the cross-entropy loss of class distributions calculated based on global class prototypes.

III. PROPOSED METHOD

Fig.1 shows the design framework of FPLPA, which is carried out under the framework of FL, and mainly includes data pre-processing, local model training, local prototype aggregation and personalized local model testing.

In the data preprocessing phase, the clients use the OSS method of unilateral selection to remove abnormal instances from metric element (feature) data. Secondly, the Chi-Squares Test method is used to filter the feature data to obtain the optimal feature subset. In the personalized local model training phase, the clients input the optimal feature subset into the CPN to obtain the prediction tag and generate the local prototype. CPN is the fusion of CNN and prototype learning. The convolution layer of CPN is used to extract deep features, and the end uses the assigned prototype to represent classes. It can better handle the continuous heterogeneous data in the incremental learning task and can retain or integrate most of the previously learned knowledge while learning new knowledge.

The local prototype is obtained by averaging the embedded vectors of similar instances. All clients send the local prototype to the server. In the local prototype aggregation stage, the server again averages the local prototype to obtain the global prototype, and sends the global prototype back to the clients. Clients use the global prototype to regularize the local model parameters and obtain personalized local models. When the client's local model converges or reaches the maximum communication round, the local model update stops. In the personalized local model testing phase, the shortest Euclidean distance between the embedded vector of the test instance and the global prototype is used to obtain the classification results. By observing the Euclidean distance between the embedding vector of the test sample and two global prototypes (positive and negative), which global prototype has the smallest distance from the embedding vector, the sample is determined as this class.

A. ONE-SIDED SELECTION FOR REMOVING NOISE DATA

From the perspective of defect dataset distribution, the defect data set has many negative noises and most of them are distributed in the boundary area as shown in Fig.2 The existence of noise in supervised learning makes the prediction in actual data more complex, which easily leads to over-fitting of the classifier on the training set. We use OSS [57] to delete negative noise instances and negative instances in the Tomek links to obtain a clean dataset, which can effectively suppress the interference of abnormal instances on

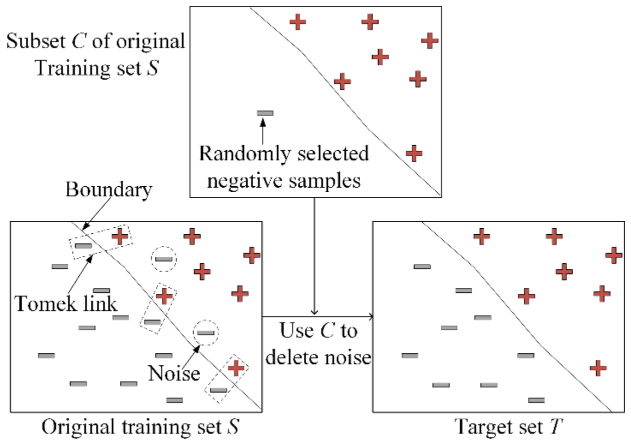
Subset C of original Training set S
Randomly selected negative samples
Boundary
Tomek link
Noise
Use C to delete noise
Original training set S
Target set T

FIGURE 2. OSS algorithm theory, OSS algorithm theory, which is used to remove noisy instances on classification boundaries.

model training. Tomek links are paired on the boundary area but do not belong to the same class. Compared with other noise filtering technologies, OSS uses different noise handling strategies for instances located in different areas. It can delete noise instances more conservatively, effectively avoid over-eliminating noise instances, and help protect the diversity of instances in the dataset.

The specific steps of the OSS algorithm are as follows:

1) Assume S as the original training set.
2) Set C is defined as all positive instances and one randomly selected from negative instances.
3) Use all the instances in the set C to reclassify S through 1-NN classification principle, and compare the assigned labels with the original labels. Move all misclassified instances into the set C, which became smaller than set S.
4) Delete all negative instances participating in Tomek-Link from C. In this way, the negative instances that are considered as boundary and noises are removed, while all the positive instances will be retained and the final set T is obtained.

The Euclidean distance formula is as follows:

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{u} (p_i - q_i)^2} \tag{1}$$

where, d is the distance between two instances, u is the feature number of instances, p_i is the dimensional feature value of instance x_1, q_i is the dimensional feature value of x_1. In this way, negative instances that are considered as edges or noises are removed, and a software defect training dataset with less noise is obtained.

B. CHI-SQUARE TEST FOR SELECTING OPTIMAL FEATURE SUBSET

The software system is quite large and complex, including many related metrics. It is very important to select a degree quantum set that can improve the performance of

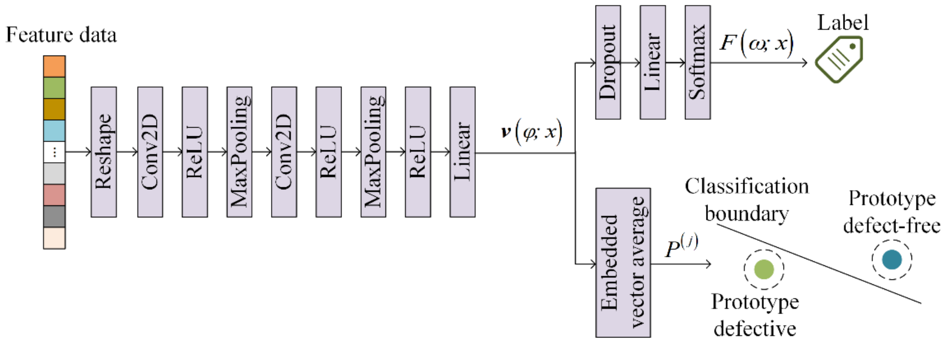

FIGURE 3. The structure of CPN, which is a combination of convolutional neural networks and prototype theory. Its output results include classification labels and local prototype vectors.

software defect prediction methods. Different historical software project in the dataset uses different metric elements, for example, some of them use 61 metric elements to measure software modules, while others use 26 metric elements.

However, these historical software project data are high-dimensional data. At the same time, when building a CPDP model using FL, the number of features of each client should be aligned to keep the local model structure of each client the same. We use the Chi-Squares Test to select the optimal feature subset of historical software project data and can be extended to high-dimensional data sets of software modules. Chi-Squares Test can be used to test the correlation between the two groups. For software defect prediction, first calculate the Chi-Squares value of each feature. The Chi-Squares value of feature f and class c is calculated as follows:

$$chi\,(f, c) = \sum_{i=1}^{n} \frac{(f-c)^2}{c} \tag{2}$$

where n is the number of instances, $chi\,(f, c)$ indicates the correlation between the feature and the class. Filtering feature selection method can reduce the difficulty of model learning while removing redundant and useless features, which can speed up the training of local models under the FL framework and can effectively avoid over fitting.

C. CPN AND FEDERATED COMMUNICATIONS

In the FL paradigm, each client has its own independent local model. However, in the increasingly complex heterogeneous data scenario, the optimal model parameters of each client are different, which means that gradient-based communication cannot effectively provide each client with information that can alleviate the heterogeneity of data. Therefore, establishing a robust FL framework will be a challenge. In our FPLPA, each client has a CPN as their local model, which is the combination of CNN and prototype learning. The convolution layer of CPN is used to extract the depth feature of defect data. The model output result will specify a prototype to represent a class. Because the same label space allows different clients to share the same embedded space, CPN processes unknown heterogeneous data in the learning task in a regularized way, while preserving or integrating previously learned knowledge. Therefore, class prototypes can efficiently exchange information between heterogeneous clients.

D. CPN AND FEDERATED COMMUNICATIONS

In the FL paradigm, each client has its own independent local model. However, in the increasingly complex heterogeneous data scenario, the optimal model parameters of each client are different, which means that gradient-based communication cannot effectively provide each client with information that can alleviate the heterogeneity of data. Therefore, it is a challenge to establish a strong FL framework. Each client has a CPN as their local model in FPLPA, which is a combination of CNN and prototype learning. The convolution layer of CPN is used to extract the depth feature of defect data. The model output result will specify a prototype to represent a class. CPN processes unknown heterogeneous data in learning tasks in a rule-based manner while preserving or integrating previously learned knowledge, as the same label space allows different clients to share the same embedding space. Therefore, class prototypes can efficiently exchange information between heterogeneous clients.

For the software defect data of a given client, the feature data is put into the CPN after data preprocessing. CPN uses vectors in low dimensional subspaces to approximate the most representative prototype. When the Softmax layer is discarded, the convolutional prototype network is more robust than the traditional CNN network in terms of data heterogeneity.

Heterogeneous data will lead to model parameter ω_i with different forms and sizes for any client. When the total number of clients is m and data heterogeneity exists, the training goal is to minimize the loss of client i. At present, most FL frameworks optimize global model parameters by simply averaging model parameters ω_i. Specifically:

$$\arg\min_{\omega_1,\omega_2,\cdots,\omega_m} \sum_{i=1}^{m} \frac{|D_i|}{N} L_S\left(F_i\left(\omega_i; x\right), y\right) \tag{3}$$

where L_S is defined as a loss function of supervised learning. $F_i\,(\omega_i; x)$ is the prediction label for local model i, x is the model input, and y is the real label. However, this brutal approach does not really solve the heterogeneity problem and cannot create a personalized model suitable for each local data distribution. Therefore, the communication and aggregation process of FPLPA is proceeded by prototype manner.

CPN is responsible for extracting the deep features of software defect framework to generate class prototypes and output prediction labels. For each class, the CPN network generates only one class prototype, which can well contain the common information of all similar instances. But the prototype does not belong to any of the original instances. Fig.3 shows the structure of convolution neural network in CPN. First, the feature is reshaped as input to CPN, which contains two convolution layers, two maximum pooling layers, and two linear layers. Then, the output embedded vector $\mathbf{v}(\varphi; x)$ of the first Linear layer will pass through two branches. $\mathbf{v}(\varphi; x)$ is mapped from the feature space to the label space through the Dropout layer, Linear layer, and Softmax classifier of the first branch to get the prediction label $F(\omega; x)$. The second branch averages the embedded vector $\mathbf{v}(\varphi; x)$ to get prototypes of different classes, and this operation is irreversible. For software defect prediction that belongs to the two-classifier task, CPN network will output one prototypes for each class.

The prototype is defined as $P^{(j)}$ to represent the class j in class set C. For client i, the prototype is the average of the embedded vector $\mathbf{v}_i(\varphi_i; x)$ of the feature data in class j, and $P_i^{(j)}$ represents the prototype of class j from client i, Because $P_i^{(j)}$ is obtained by averaging low dimensional embedding vectors which is mathematically irreversible and no information related to training data can be obtained.

$$P_i^{(j)} = \frac{1}{|D_{i,j}|} \sum_{(x,y)\in D_{i,j}} \mathbf{v}_i(\varphi_i; x) \tag{4}$$

$D_{i,j}$ is a subset of the local dataset D_i and consists of training instances belonging to class j. The server aggregates local prototypes from multiple clients to generate a global prototype, specifically:

$$\bar{P}^{(j)} = \frac{1}{|\mathcal{N}_j|} \sum_{i\in N_j} \frac{|D_{i,j}|}{N_j} P_i^{(j)} \tag{5}$$

$\mathcal{N}_j$ is a collection of clients with class j examples. In the optimization objective of FPLPA, minimize the total loss sum of all clients' local learning tasks. $F_i(\omega_i; x)$ is the prediction label of the model. N is the total number of instances on all clients. L_S and OPT_S are defined as the optimization objective of local and global supervised learning, respectively. Specifically:

$$OPT_S = \underset{\{\bar{P}^{(j)}\}_{j=1}^{|P|}}{\text{argmin}} \sum_{i=1}^{m} \frac{|D_i|}{N} L_S(F_i(\omega_i; x), y) \tag{6}$$

The mean square error (MSE) loss function is used to calculate the loss of supervised learning for each client. y_k is the real label of instance k. $F_{i,k}(\omega_i; x)$ is the prediction label. n is the total number of instances for the client i. The MSE loss function is expressed as:

$$MSE\left(y_k, F_{i,k}(\omega_i; x)\right) = \frac{\sum_{k=1}^{n}\left(y_k - F_{i,k}(\omega_i; x)\right)^2}{n} \tag{7}$$

Secondly, the same MSE loss function is used to minimize the loss of each client's local prototype and global prototype L_R, which will be used as the regularization item of supervised learning. OPT_R is the global optimization objective of prototype learning. N_j is the number of instances belonging to the class j on all clients. The specific implementation formula is written as follows:

$$\text{OPT}_\text{R} = \sum_{j=1}^{|p|}\sum_{i=1}^{m} \frac{|D_{i,j}|}{N_j} L_\text{R}\left(\bar{P}_i^{(j)}, P_i^{(j)}\right) \tag{8}$$

Then, the global optimization objective OPT of FPLPA can be expressed as:

$$OPT = \underset{\{\bar{P}^{(j)}\}_{j=1}^{|P|}}{\text{argmin}} \sum_{i=1}^{m} \frac{|D_i|}{N} L_S(F_i(\omega_i; x), y)$$

Algorithm 1 FPLPA

```
Initialize: $\omega_i, i = 1, \cdots, m$ for each client,
Input: $D_i, \omega_i$, Number of users $m$, Number of communication rounds $T$, Local update rounds $E$, Local learning rate $\mu$, Local dataset $C_i$
Server executes:
1: Initialize global prototype matrices $\{\bar{P}^{(j)}\}$ for all classes.
2: for each communication round $t = 1, \ldots, T$ do
3: for each client $i \in \{1, 2, \ldots, m\}$ in paralled do
4:         $P_i \leftarrow$ Local Update $\{i, \bar{P}_i\}$
5:     end for
6:     Server selects a local prototype subset.
7:         $\{P_s^{(j)}\} \leftarrow Random\{P_m^{(j)}\}$
8:     Selected clients update global prototype.
9:             $\bar{P}^{(j)} \leftarrow \{P_s^{(j)}\}$
10:    Clients update local prototype $P_i$ with prototype in $\{\bar{P}^{(j)}\}$
11: end for
Local Update $(i, \bar{P}_i)$:
1: for each local epoch $e = 1, 2, \ldots, E$ do
2:     for each $(x_i, y_i) \in D_i$ do
3:         Client $i$ collects local embedded vectors
4:         $F_i(\omega_i; x), v_i(\varphi_i; x) \leftarrow CPN_i$
5: Client $i$ generates the prototype for class $j$.
6:             $P_i^{(j)} \leftarrow v_i(\varphi_i; x)$
7: Client $i$ computes loss by using local prototypes.
8:         $L(D_i, \omega_i) \leftarrow L_S(F_i(\omega_i; x), y) + \lambda \cdot L_R(\bar{P}_i, P_i)$
9: Client updates local model according to the loss $L(D_i, \omega_i)$.
10:     end for
11: end for
12: return$P_i$
```

TABLE 1. **Statistics of items used in the experiment.**

Database	Project	Description of Project	Metrics	Instance	Defects	Defective (%)
NASA	PC3	Flight Software of Each Orbiting Satellite	37	1077	134	12.44
	PC4	Flight Software of Each Orbiting Satellite	37	1458	178	12.21
	MW1	A Zero Gravity Experiment on Combustion	37	253	27	10.67
AEEEM	EQ	OSGi Framework	61	324	129	39.81
	JDT	IDE Development	61	997	206	20.66
	LC	Text Search Engine Library	61	691	64	9.26
	ML	Task management	61	1862	245	13.16
	PDE	Development	61	1497	209	13.96
Relink	Apache	Open-Source software system	26	194	98	50.52
	Safe	Open Intents Library	26	56	22	39.29

$$+\lambda \cdot \sum_{j=1}^{|P|} \sum_{i=1}^{m} \frac{|D_{i,j}|}{N_j} L_R \left(\bar{P}^{(j)}, P_i^{(j)}\right) \tag{9}$$

λ is the regularization weighting factor. When the client updates the local model, to generate consistent class prototypes between clients, the regularization item is added to the local loss function. The local prototype $P_i^{(j)}$ can approach the global prototype $\bar{P}_i^{(j)}$ while minimizing the classification error. The loss function is defined as follows:

$$L(D_i, \omega_i) = L_S(F_i(\omega_i; x), y) + \lambda \cdot L_R(\bar{P}_i, P_i) \tag{10}$$

When the local optimization goal of the client converges, the local model will enter the test stage. In the implementation, the MSE loss function uses L2 as the distance between the embedded vector obtained from the instance and the local class prototype. The output of the test set is calculated as:

$$\hat{y} = \underset{j}{\operatorname{argmin}} \left\| v_i(\varphi_i; x) - P^{(j)} \right\|_2 \tag{11}$$

where $\mathbf{v}_i(\varphi_i; x)$ is the embedded vector obtained from feature data of the test instance x. In Algorithm 1, we show the principle and process of FPLPA specific to each client.

IV. EXPERIMENTS AND ANALYSIS

A. EXPERIMENTAL SETTINGS

1) DATASETS DESCRIPTION

We used NASA, AEEEM [58] and Relink [12] datasets widely used in software defect prediction to evaluate the performance of FPLPA algorithm. NASA datasets come from 13 actual software projects. Each project contains several modules from different software projects, ranging from 125 modules to 17186 modules. It is composed of LOC, McCabe, Halstead and other types of metrics and defective class labels.

The AEEEM data set was compiled by D'Ambros et al. Each item in AEEEM contains 61 features, of which 17 are related to source code, 5 are related to previous prediction, 5 are related to code change entropy, 17 are related to source code entropy and 17 are related to source code decay.

The Relink dataset was collected and collated by Wu et al., and the defect information in the dataset was confirmed manually. They use the Understand tool to analyze three projects (such as Apache, Safe and ZXing), and extract important software feature indicators from the source code. Relink dataset has 26 complexity features, which are mainly based on code complexity and abstract syntax tree and can be generally divided into two classes: features based on program complexity and features based on quantity. Table 1 shows the statistics of the items used in the experiment.

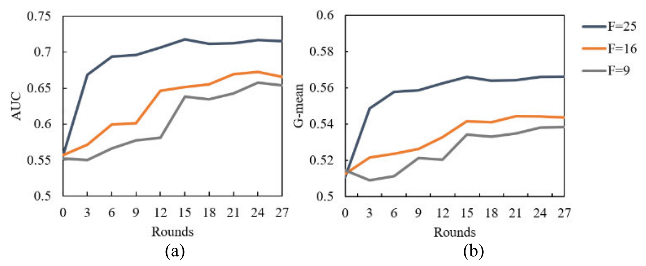

FIGURE 4. **Impact of the number of filtering features on prediction performance. (a) AUC. (b) G-mean.**

TABLE 2. **Confusion matrix.**

	Predict as Defective	Predict as defect-free
Defective	True positive (TP)	False negative (FN)
Defect-free	False positive (FP)	True negative (TN)

2) EXPERIMENTAL ENVIRONMENT AND EVALUATION METRICS

In this paper, the programme software used in the experiments is Pycharm 2018 and Pytorch 1.4.1 to build the FPLPA framework, and all experiments are carried out on NVIDIA GTX GeForce 1650 Ti.

In software defect prediction, defective instances are usually called positive class, and defect-free instances are called negative class, and the model evaluation index is Area Under Curve (AUC), which is defined as the area enclosed by receiver operating characteristic curve and coordinate axis, which can make a more reasonable evaluation of the classifier.

The G-mean index is a geometric mean, which can consider the classification performance of the software defect

TABLE 3. Randomly select local prototype subsets from 5 clients for aggregation.

Project \ number	Index	2	3	4	5
PC3	AUC	0.5000	0.7792	0.7701	0.7802
	G-mean	0.5000	0.5930	0.5896	0.5933
MW1	AUC	0.6434	0.6581	0.6507	0.5000
	G-mean	0.5404	0.5446	0.5425	0.5000
EQ	AUC	0.7064	0.7575	0.6271	0.7229
	G-mean	0.5688	0.5858	0.5279	0.5732
JDT	AUC	0.5000	0.5000	0.5000	0.5000
	G-mean	0.5000	0.5000	0.5000	0.5000
LC	AUC	0.6998	0.7786	0.7286	0.5178
	G-mean	0.5616	0.5920	0.5740	0.4988
Average	AUC	0.6099	**0.6947**	0.6553	0.6042
	G-mean	0.5342	**0.5631**	0.5469	0.5330
	Time(s)	104.5989	104.2988	104.7092	104.8400

TABLE 4. Randomly select local prototype subsets from 6 clients for aggregation.

Project \ number	Index	2	3	4	5	6
PC3	AUC	0.7652	0.7885	0.7929	0.7752	0.7971
	G-mean	0.5873	0.5961	0.5976	0.5916	0.5991
PC4	AUC	0.6052	0.8241	0.8377	0.8360	0.8165
	G-mean	0.5317	0.6080	0.6123	0.6118	0.6054
MW1	AUC	0.6331	0.4382	0.6360	0.6360	0.6044
	G-mean	0.5404	0.4754	0.5384	0.5384	0.5216
EQ	AUC	0.4059	0.4957	0.4872	0.4872	0.4885
	G-mean	0.4503	0.4723	0.4684	0.4684	0.4716
JDT	AUC	0.8104	0.7506	0.7956	0.7882	0.7652
	G-mean	0.6028	0.5812	0.5984	0.5954	0.5872
LC	AUC	0.5606	0.6836	0.7350	0.7279	0.7458
	G-mean	0.5047	0.5531	0.5760	0.5738	0.5793
Average	AUC	0.6337	0.6634	**0.7141**	0.7084	0.7029
	G-mean	0.5362	0.5477	**0.5652**	0.5632	0.5607
	time(s)	223.2827	223.7503	223.5860	223.3990	223.1628

prediction models for positive and negative classes, defined as Eq.12.

$$G-\text{mean} = \sqrt{\frac{TP}{FN+TP} * \frac{TN}{TN+FP}} \tag{12}$$

G-mean can be calculated by the confusion matrix as shown in Table 2, if the classifier is biased toward one of the classes, the correctness of the other will be affected. The larger the value of the G-mean index, the higher the classification accuracy of positive and negative classes, and the better the performance of the model.

AUC and G-mean have the following advantages compared to Accuracy. AUC has strong robustness to class imbalance problems. Accuracy may experience biases when dealing with datasets with imbalanced classes, while AUC is not sensitive to changes in class distribution. It provides a comprehensive measure that can provide a more comprehensive and fair model evaluation in cases of imbalanced classes.

G-mean performs equally well in dealing with class imbalance issues. It can comprehensively evaluate the performance of the model on different classes by considering the geometric mean of classification accuracy and recall for each class. G-mean focuses more on the Accuracy of classification for smaller classes, which is very useful for important but small sample classes.

B. FEATURE SELECTION ANALYSIS

Different defect datasets use different metrics, which reflects the heterogeneity of software defect data. For example, AEEEM uses 61 metric elements to analyze the characteristics of software modules. NASA uses 37 metric elements. Relink uses the least 26 metric elements. These datasets are high-dimensional data. FL requires that the number of features between different clients should be aligned to keep the local model structure of each client the same. Chi-square algorithm can maintain the fidelity of training data after processing the mixed attribute data of defect data. It is a reliable method for digital discretization and feature selection, which can eliminate irrelevant or redundant features of defect data and improve model accuracy.

Therefore, we used Chi-Squares Test to verify the effect of the number of features on federated training, as shown in Fig.4 where NASA and Relink datasets are used as clients. F is the number of features.

TABLE 5. Randomly select local prototype subsets from 8 clients for aggregation.

Project \ number	Index	2	3	4	5	6	7	8
PC3	AUC	0.7584	0.7542	0.5959	0.6738	0.7954	0.6008	0.7936
	G-mean	0.5859	0.5844	0.5259	0.5466	0.5985	0.5235	0.5979
PC4	AUC	0.5000	0.7539	0.8326	0.7690	0.8120	0.7926	0.7903
	G-mean	0.4944	0.5812	0.6107	0.5874	0.6029	0.5963	0.5954
MW1	AUC	0.5147	0.4442	0.6507	0.5753	0.5992	0.6096	0.5798
	G-mean	0.4904	0.4619	0.5425	0.5088	0.5202	0.5230	0.5158
EQ	AUC	0.3062	0.3070	0.4872	0.5958	0.5000	0.4872	0.4872
	G-mean	0.4323	0.4326	0.4684	0.5291	0.5000	0.4684	0.4684
JDT	AUC	0.7672	0.7679	0.6226	0.4896	0.7603	0.7623	0.7711
	G-mean	0.5870	0.5875	0.5320	0.4697	0.5854	0.5861	0.5893
LC	AUC	0.6483	0.6259	0.7450	0.7485	0.7493	0.7477	0.7495
	G-mean	0.5398	0.5303	0.5791	0.5801	0.5803	0.5799	0.5804
ML	AUC	0.6475	0.6481	0.6485	0.5115	0.6766	0.6528	0.6611
	G-mean	0.5464	0.5470	0.5483	0.4895	0.5574	0.5485	0.5523
PDE	AUC	0.6489	0.5904	0.5549	0.6670	0.6678	0.6602	0.6626
	G-mean	0.5476	0.5240	0.5063	0.5516	0.5526	0.5478	0.5501
Average	AUC	0.5989	0.6114	0.6422	0.6288	**0.6951**	0.6642	0.6869
	G-mean	0.5280	0.5311	0.5391	0.5328	**0.5622**	0.5467	0.5562
	time(s)	369.4015	367.5804	369.5575	369.9910	369.7329	369.7488	367.2336

TABLE 6. Comparison results with other methods when NASA and AEEEM are used as datasets.

Project	Index	CCA+	KCCA+	FedAvg	FTLKD	FRLGC	FPLPA
PC3	AUC	0.4703	0.6029	0.5361	0.5417	0.5753	**0.7673**
	G-mean	0.4830	0.5232	0.5117	0.5092	0.5071	**0.5889**
PC4	AUC	0.4618	0.4861	0.5222	0.5530	0.6148	**0.8098**
	G-mean	0.4859	0.4859	0.5071	0.5014	0.5238	**0.6030**
MW1	AUC	0.5322	0.5317	0.5734	0.7091	0.6083	**0.6360**
	G-mean	0.5048	0.5180	0.5242	**0.5685**	0.5208	0.5384
JDT	AUC	0.4920	0.4933	0.5336	0.5119	0.5174	**0.7964**
	G-mean	0.4972	0.4917	0.5079	0.5013	0.4808	**0.5978**
LC	AUC	0.5367	0.4279	0.5755	0.4750	0.5597	**0.7761**
	G-mean	0.5367	0.4774	0.5243	0.4655	0.5007	**0.5901**
Average	AUC	0.4986	0.5084	0.5482	0.5581	0.5751	**0.7571**
	G-mean	0.5015	0.4992	0.5150	0.5092	0.5066	**0.5836**

After calculating the Chi-square values of all features and arranging them in descending order, the size of the best feature subset is selected as the hyperparameters.

We set the parameters for feature selection to 9, 16, and 25 to construct tensor inputs for the convolutional network. When selecting 25 optimal features as feature subsets, the convergence time is shorter and the convergence curve is smoother compared to 9 and 16. This shows that when 9 and 16 features are selected as feature subset, some useful features have been eliminated, which will significantly reduce the performance of the model. When the number of features is 25, the defect information of local data from different clients can be retained to the greatest extent.

In the feature selection process with NASA and Relink as data sets, the models perform well when each client uses 25 features as the 5×5 input of the CPN.

C. AGGREGATION ANALYSIS OF RANDOMLY SELECTED CLIENTS

The FL system assumes that all client model information participates in aggregation. However, the data from different clients is heterogeneous. The training performance of local models is also uneven. The aggregation of local models directly on the server side has problems such as reduced accuracy of aggregation models. The prototype considers the difference of clustering structure between clients due to the difference of data distribution, and avoids introducing greater heterogeneity through prototype subset in each round of update. The exchange of information between prototypes achieves the purpose of mutual learning, allowing clients with heterogeneous data to benefit from each other, while minimizing harmful interference.

To explore the impact of different local prototype subsets on local model optimization in FPLPA when aggregating global prototypes, we set up a random sampling in the prototype set to aggregate global prototypes. It is important to note that this subset selection strategy does not permanently freeze unselected local prototypes. Members of the prototype subset are refreshed at each global update, which means that all local prototypes have an opportunity to be aggregated objects.

Five different projects in NASA and AEEEM are used as client sources for HDP and experimental results of the

TABLE 7. Comparison results with other methods when NASA and relink are used as datasets.

Project	Index	CCA+	KCCA+	FedAvg	FTLKD	FRLGC	FPLPA
PC3	AUC	0.5156	0.6294	0.5179	0.5190	0.4988	**0.7578**
	G-mean	0.5040	0.5293	0.5059	0.5024	0.4711	**0.5856**
PC4	AUC	0.6000	0.4806	0.5377	0.4788	0.5561	**0.7952**
	G-mean	0.5177	0.4940	0.5116	0.467	0.4979	**0.5981**
MW1	AUC	0.4579	0.5155	0.5051	0.7463	0.5000	**0.6434**
	G-mean	0.4873	0.5068	0.5000	0.5827	0.5000	**0.5404**
Apache	AUC	0.5373	0.5275	0.5109	0.5602	0.5718	**0.6782**
	G-mean	0.5151	0.5127	0.5032	0.5102	0.5172	**0.5594**
Safe	AUC	0.5537	0.5940	0.5535	0.5000	0.7500	**0.8500**
	G-mean	0.5205	0.5397	0.5171	0.5000	0.5774	**0.6146**
Average	AUC	0.5329	0.5494	0.5250	0.5609	0.5753	**0.7449**
	G-mean	0.5089	0.5165	0.5076	0.5125	0.5127	**0.5796**

TABLE 8. Comparison results with other methods when aeeem and relink are used as datasets.

Project	Index	CCA+	KCCA+	FedAvg	FTLKD	FRLGC	FPLPA
EQ	AUC	0.5171	0.5715	0.5120	0.6500	0.6389	**0.7099**
	G-mean	0.5060	0.5314	0.5038	0.5477	0.5307	**0.5671**
JDT	AUC	0.5010	0.5009	0.5199	0.5119	0.6857	**0.7844**
	G-mean	0.5026	0.4980	0.5066	0.4907	0.5367	**0.5934**
LC	AUC	0.5961	0.5147	0.5336	0.7000	0.5718	**0.7314**
	G-mean	0.5323	0.5009	0.5106	0.5578	0.5543	**0.5731**
Apache	AUC	0.5281	0.5331	0.5104	0.6429	**0.7045**	0.6971
	G-mean	0.5094	0.5154	0.5035	0.5429	0.5172	**0.5640**
Safe	AUC	0.5084	0.5596	0.5655	0.4944	0.5833	**0.6429**
	G-mean	0.5018	0.5256	0.5205	0.4898	**0.5641**	0.5345
Average	AUC	0.5301	0.5360	0.5283	0.5998	0.6368	**0.7131**
	G-mean	0.5104	0.5143	0.5090	0.5258	0.5406	**0.5664**

proposed FPLPA are shown in Table 3. As the size of the local prototype subset increases, the local model performs best when 3 of 5 local prototypes are randomly selected. When the average AUC and G-mean reach 0.6947 and 0.5631 respectively, the model is in the best state.

In each round of training, when a large subset of local prototypes participates in the aggregation, greater data heterogeneity will be introduced, and the generated global prototypes are unfavorable to the regularization of local models. At the same time, the global prototype obtained by aggregating a small subset of local prototypes cannot better describe the global distribution of the same class of samples, and the regular items will be more likely to deviate to a single client. Therefore, when the total number of clients is 5, the best selection parameter is 3.

In addition, as the size of the local prototype subset changes, the time for training is comparable, about 104 seconds. The FPLPA algorithm is aggregated in the form of local prototype subsets. When the total number of clients is fixed, the change in the number of the local prototype subset will not affect the final training time.

To explore the generalization of the randomly chosen prototype subset method with different total number of clients, we set the total number of clients to 6 and 8 as shown in Table 4 and 5. After the total number of clients increased from 5 to 6 and 8, the training time also increased for the number of the local clients has increased. When the number of clients changed from 5 to 6, 1458 instances were added for training and testing, which increased the training time from 104s to 223s. Similarly, when the number of clients increased from 6 to 8, the 3359 additional instances for training and testing increased the training time from 223s to 369s.

It is worth noting that when the total number of clients is fixed to 6 and 8, and different local prototype subset sizes are used for aggregation, the training time is still comparable. In Table 4 and Table 5, the best performance of model is to select 4 and 6 local prototypes when there are 6 and 8 clients. Changing the size of a local prototype subset will not change the final training time.

Considering the training time and performance of the model together, it is best to set the total number of clients and the local prototype subset to 5 and 3 with less training time and good prediction performance. Therefore, our next experiment will be carried out according to this setting.

These results show that in heterogeneous CPDP scenarios, with the increase of the total number of clients, the stray problem will become more serious, and the training time will also increase. However, aggregation can be achieved by randomly selecting a subset of local prototypes, which can mitigate the stray caused by data heterogeneity. However, the deviation caused by data heterogeneity can be mitigated by selecting a subset of local prototypes of appropriate size.

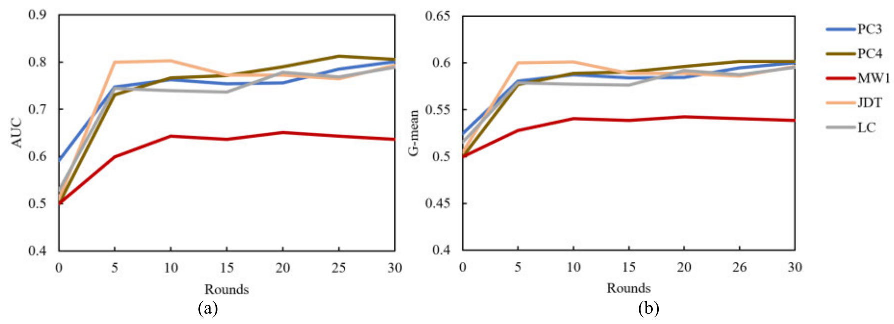

FIGURE 5. Convergence trend of different projects in NASA and AEEEM with the increase of communication rounds. (a) AUC. (b) G-mean.

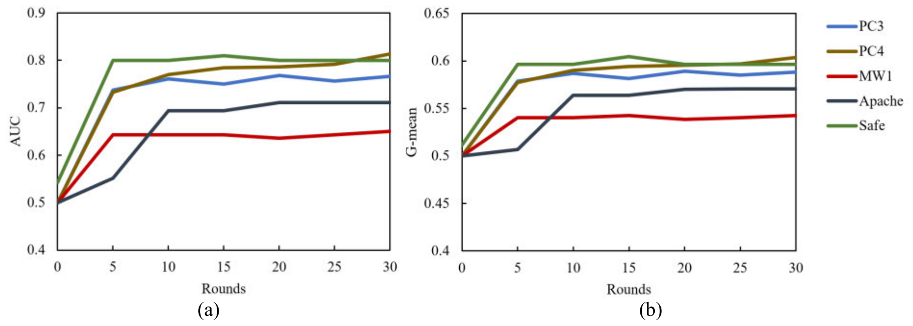

FIGURE 6. Convergence trend of different projects in NASA and Relink with the increase of communication rounds. (a) AUC. (b) G-mean.

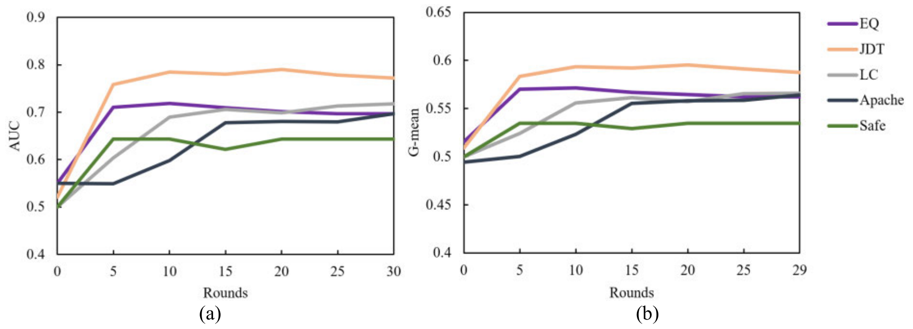

FIGURE 7. Convergence trend of different projects in AEEEM and Relink with the increase of communication rounds. (a) AUC. (b) G-mean.

D. COMPARISON WITH THE ADVANCED SOFTWARE DEFECT PREDICTION METHODS

To verify the advantages of FPLPA over other methods in predicting heterogeneous software defects, we used CCA+, KCCA+, FedAvg, FTLKD and FRLGC as comparative methods.

Table 6 shows the experimental results of five projects in NASA dataset and AEEEM dataset with different clients, namely PC3, PC4, MW1, JDT and LC. The AUC of all clients in the FPLPA method is 0.6360∼0.8098, the G-mean is 0.5384∼0.6030, and the average values are 0.7571 and 0.5836 respectively. Taking the average value as the comparison object, it is higher than CCA+, KCCA+, FedAvg, FTLKD and FRLGC (0.2585, 0.0821), (0.2487, 0.0844), (0.2089, 0.0686), (0.1990, 0.0744) and (0.1820, 0.0770) respectively.

In this group of experiments, except that the G-mean of project MW1 is lower than the comparison experiment, the prediction performance of other projects is superior to other methods. MW1 has the same number of features compared to PC3 and PC4. But the dataset collection is based on different project than PC3 and PC4. We believe that the reason why MW1 is not as high as the comparative experiment is because the local prototype of MW1 has a large intra cluster spacing compared to other client prototypes, which is very unfavorable for MW1.The experimental results show that when FPLPA processes complex heterogeneous data, CPN can well handle the classification of multi-source heterogeneous data, thus improving the robustness of FPLPA.

Table 7 shows the experimental results of five projects in NASA dataset and Relink dataset with different clients, namely PC3, PC4, MW1, Apache and Safe. AUC of all clients in FPLPA is 0.6434∼0.8500, G-mean is 0.5404∼0.6146, and the average values are 0.7449 and 0.5796 respectively. Compared with CCA+, KCCA+, FedAvg, FTLKD and FRLGC, they were higher (0.2120, 0.0707), (0.1955, 0.0631), (0.2199, 0.0720), (0.1840, 0.0671) and (0.1696, 0.0669) respectively.

In this group of experiments, AUC and G-mean of all items were higher than those of other methods. The experimental results also show that CPN is updated in a regular way, which avoids the bias effect on traditional convolutional network when learning new knowledge from other clients.

Table 8 shows the experimental results of five projects in the AEEEM dataset and Relink dataset as different clients, respectively EQ, JDT, LC, Apache and Safe. AUC of all clients in FPLPA is 0.6429∼0.7644, and G-mean is 0.5345∼0.5860. The average values were 0.6945 and 0.5589, respectively. Taking the average value as the comparison object, it is higher than CCA+, KCCA+, FedAvg, FTLKD and FRLGC (0.1644, 0.0485), (0.1585, 0.0446), (0.1662, 0.0499), (0.0947, 0.0331) and (0.0677, 0.0183) respectively. In this group of experiments, except that AUC and G-mean of LC project are lower than other methods, Apache's AUC and Safe's G-mean are also lower than other methods. The prediction performance of other projects is higher than that of other methods. FPLPA reduces the difference in local data distribution among multiple clients by averaging local class prototypes. The global class prototype helps the feature extraction network to obtain more accurate features. The experimental results show that the prototype can solve the heterogeneity of multi-party software defect data. Compared with other software defect methods, FPLPA has obvious advantages.

E. THE CONVERGENCE ANALYSIS

To explore the change trend of prediction performance of each client as the number of communication rounds increases, Fig. 5, Fig. 6 and Fig. 7 take NASA and AEEEM, NASA and Relink, AEEEM and Relink as the data sets of clients, respectively, to observe the change of AUC and G-mean evaluation indicators as the number of communication rounds increases. Obviously, with the increase of rounds, most clients show stable convergence.

In Fig. 5, AUC and G-mean increase by 0.2386 and 0.0774 on average after the model converges compared with that without federated communication. In $0<$ Rounds<5, PC3, PC4, JDT and LC showed a stable growth trend and reached convergence. In $0<$ Rounds<10, MW1 has achieved convergence despite slow growth. It can be seen from the curve on the figure that 4 of the 5 clients have a relatively compact prototype distribution, resulting in a rapid upward trend. The prototype distribution of MW1 is more remote from the other four.

In Fig. 6, AUC and G-mean increased by 0.2399 and 0.0780 on average after the model converges compared with that without federated communication. In $0<$ Rounds<5, PC3, PC4, MW1 and Safe show a stable growth trend and reach convergence. Apache reaches convergence when Rounds $=$ 10. The upward trend of these 5 curves is very rapid, which indicates that the prototype distribution between these 5 clients is more compact than that of the 5 in Fig. 6.

In Fig. 7, AUC and G-mean increased by 0.1812 and 0.0592 on average after the model converges compared with that without federated communication. In $0<$ Rounds<5, EQ, JDT, and Safe grow rapidly and converge, while LC and Apache converge when Rounds $=$ 10 and 15 respectively.

In general, we propose that FPLPA has better prediction performance than CCA+, KCCA+, FedAvg, FTLKD and FRLGC. FPLPA combines the advantages of FL and prototype learning, that is, FL realizes cross project construction of software defect prediction, and prototype ensures consistency between heterogeneous clients. FPLPA allows to use a small amount of local software defect data to build a more effective prediction model, which is more successful in balancing generalization and personalization.

V. CONCLUSION

This paper proposes FPLPA for heterogeneous defect prediction. FPLPA uses multiple nodes with limited data sets to build an efficient personalized software defect prediction model. First, clients use OSS technology to remove noise instances from private data, and use Chi-Squares Test technology to filter out the optimal feature subset of local datasets. The CPN built by prototype learning and CNN is used to generate local prototypes, which are obtained from depth features. The prototype is used as the communication subject between the clients and the server, replacing the model parameters used in the traditional FL architecture. Because the local prototype is generated in an irreversible way, the privacy information of the local data will not be exposed during the communication process. Secondly, the server aggregates subsets of local prototypes to generate global prototypes, which can avoid the stray problem caused by heterogeneity and better characterize the global distribution of same class data. Finally, the local CPN network is updated with the loss of local prototype and global prototype as regularization, which can ensure the integrity of the original knowledge while continuously learning other heterogeneous data, thus establishing a local model that is robust to new data. In the test phase, CPN abandoned CNN's Softmax and adopted the Euclidean distance between the low-dimensional embedded vector and the local prototype as the classification criterion.

We have verified on 10 projects in three public data sets (AEEEM, NASA and Relink). Compared with other methods, AUC and G-mean are up to 0.2585 and 0.0844 higher. The experimental results show that FPLPA is superior to other HDP solutions.

REFERENCES

[1] M. B. R. Pandit and N. Varma, "A deep introduction to AI based software defect prediction (SDP) and its current challenges," in *Proc. IEEE Region Conf. (TENCON)*, Kochi, India, Oct. 2019, pp. 284–290, doi: 10.1109/TENCON.2019.8929661.

[2] L. Qiao, G. Li, D. Yu, and H. Liu, "Deep feature learning to quantitative prediction of software defects," in *Proc. IEEE 45th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Madrid, Spain, Jul. 2021, pp. 1401–1402, doi: 10.1109/COMPSAC51774.2021.00204.

[3] J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk, "On the value of static analysis for fault detection in software," *IEEE Trans. Softw. Eng.*, vol. 32, no. 4, pp. 240–253, Apr. 2006, doi: 10.1109/TSE.2006.38.

[4] X. Cai, Y. Niu, S. Geng, J. Zhang, Z. Cui, J. Li, and J. Chen, "An undersampled software defect prediction method based on hybrid multi-objective cuckoo search," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 5, Mar. 2020, doi: 10.1002/cpe.5478.

[5] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empirical Softw. Eng.*, vol. 17, nos. 4–5, pp. 531–577, Aug. 2012, doi: 10.1007/s10664-011-9173-9.

[6] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007, doi: 10.1109/TSE.2007.256941.

[7] Q. Yu, J. Qian, S. Jiang, Z. Wu, and G. Zhang, "An empirical study on the effectiveness of feature selection for cross-project defect prediction," *IEEE Access*, vol. 7, pp. 35710–35718, 2019, doi: 10.1109/ACCESS.2019.2895614.

[8] W. Wen, B. Zhang, X. Gu, and X. Ju, "An empirical study on combining source selection and transfer learning for cross-project defect prediction," in *Proc. IEEE 1st Int. Workshop Intell. Bug Fixing (IBF)*, Hangzhou, China, Feb. 2019, pp. 29–38, doi: 10.1109/IBF.2019.8665492.

[9] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 811–833, Sep. 2018, doi: 10.1109/TSE.2017.2724538.

[10] F. Wu, X.-Y. Jing, Y. Sun, J. Sun, L. Huang, F. Cui, and Y. Sun, "Cross-project and within-project semisupervised software defect prediction: A unified approach," *IEEE Trans. Rel.*, vol. 67, no. 2, pp. 581–597, Jun. 2018, doi: 10.1109/TR.2018.2804922.

[11] T. Jiang, L. Tan, and S. Kim, "Personalized defect prediction," in *Proc. 28th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Silicon Valley, CA, USA, Nov. 2013, pp. 279–289, doi: 10.1109/ASE.2013.6693087.

[12] Y. Huang and X. Xu, "Two-stage cost-sensitive local models for heterogeneous cross-project defect prediction," in *Proc. IEEE 46th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Los Alamitos, CA, USA, Jun. 2022, pp. 819–828, doi: 10.1109/COMPSAC54236.2022.00132.

[13] E. Kim, J. Baik, and D. Ryu, "Heterogeneous defect prediction through correlation-based selection of multiple source projects and ensemble learning," in *Proc. IEEE QRS*, Hainan, China, Dec. 2021, pp. 503–513, doi: 10.1109/QRS54544.2021.00061.

[14] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, Jun. 2013, doi: 10.1109/TR.2013.2259203.

[15] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," in *Proc. 40th Int. Conf. Softw. Eng. (ICSE)*, Gothenburg, Sweden, May 2018, p. 699, doi: 10.1145/3180155.3182520.

[16] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online defect prediction for imbalanced data," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, Florence, Italy, vol. 2, May 2015, pp. 99–108, doi: 10.1109/ICSE.2015.139.

[17] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Inf. Softw. Technol.*, vol. 54, no. 3, pp. 248–256, Mar. 2012, doi: 10.1016/j.infsof.2011.09.007.

[18] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 2, pp. 111–147, Feb. 2019, doi: 10.1109/TSE.2017.2770124.

[19] X.-Y. Jing, F. Wu, X. Dong, and B. Xu, "An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems," *IEEE Trans. Softw. Eng.*, vol. 43, no. 4, pp. 321–339, Apr. 2017, doi: 10.1109/TSE.2016.2597849.

[20] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 874–896, Sep. 2018, doi: 10.1109/TSE.2017.2720603.

[21] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in *Proc. ICST*, Luxembourg, 2013, pp. 252–261, doi: 10.1109/ICST.2013.38.

[22] Z. Li, X.-Y. Jing, X. Zhu, and H. Zhang, "Heterogeneous defect prediction through multiple kernel learning and ensemble learning," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Shanghai, China, Sep. 2017, pp. 91–102, doi: 10.1109/ICSME.2017.19.

[23] J. Chen, Y. Yang, K. Hu, Q. Xuan, Y. Liu, and C. Yang, "Multiview transfer learning for software defect prediction," *IEEE Access*, vol. 7, pp. 8901–8916, 2019, doi: 10.1109/ACCESS.2018.2890733.

[24] H. Chen, X.-Y. Jing, and B. Xu, "Heterogeneous defect prediction through joint metric selection and matching," in *Proc. IEEE 21st Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Hainan, China, Dec. 2021, pp. 367–377, doi: 10.1109/QRS54544.2021.00048.

[25] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "On the multiple sources and privacy preservation issues for heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 4, pp. 391–411, Apr. 2019, doi: 10.1109/TSE.2017.2780222.

[26] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[27] P. Zhu, W. Hu, C. Yuan, and L. Li, "Prototype learning using metric learning based behavior recognition," in *Proc. 20th Int. Conf. Pattern Recognit.*, Istanbul, Turkey, Aug. 2010, pp. 2604–2607, doi: 10.1109/ICPR.2010.638.

[28] S. Huda, S. Alyahya, M. M. Ali, S. Ahmad, J. Abawajy, H. Al-Dossari, and J. Yearwood, "A framework for software defect prediction and metric selection," *IEEE Access*, vol. 6, pp. 2844–2858, 2018, doi: 10.1109/ACCESS.2017.2785445.

[29] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, "An ensemble oversampling model for class imbalance problem in software defect prediction," *IEEE Access*, vol. 6, pp. 24184–24195, 2018, doi: 10.1109/ACCESS.2018.2817572.

[30] L. Zhao, Z. Shang, L. Zhao, A. Qin, and Y. Y. Tang, "Siamese dense neural network for software defect prediction with small data," *IEEE Access*, vol. 7, pp. 7663–7677, 2019, doi: 10.1109/ACCESS.2018.2889061.

[31] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, Bergamo, Italy, Aug. 2015, pp. 496–507.

[32] L. Gong, S. Jiang, Q. Yu, and L. Jiang, "Unsupervised deep domain adaptation for heterogeneous defect prediction," *IEICE Trans. Inf. Syst.*, vol. E102.D, no. 3, pp. 537–549, Mar. 2019, doi: 10.1587/transinf.2018EDP7289.

[33] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, San Francisco, CA, USA, May 2013, pp. 382–391.

[34] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011, doi: 10.1109/TNN.2010.2091281.

[35] Y. Sun, X.-Y. Jing, F. Wu, J. Li, D. Xing, H. Chen, and Y. Sun, "Adversarial learning for cross-project semi-supervised defect prediction," *IEEE Access*, vol. 8, pp. 32674–32687, Feb. 2020, doi: 10.1109/ACCESS.2020.2974527.

[36] J. Liu, C. Wang, H. Su, B. Du, and D. Tao, "Multistage GAN for fabric defect detection," *IEEE Trans. Image Process.*, vol. 29, pp. 3388–3400, 2020, doi: 10.1109/TIP.2019.2959741.

[37] Y. Ma, S. Zhu, Y. Chen, and J. Li, "Kernel CCA based transfer learning for software defect prediction," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 8, pp. 1903–1906, 2017, doi: 10.1587/transinf.2016EDL8238.

[38] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2016, pp. 1273–1282.

[39] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[40] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," 2021, *arXiv:2102.07623*.

[41] A. Madi, O. Stan, A. Mayoue, A. Grivet-Sébert, C. Gouy-Pailler, and R. Sirdey, "A secure federated learning framework using homomorphic encryption and verifiable computing," in *Proc. RDAAPS*, Hamilton, ON, Canada, 2021, pp. 1–8, doi: 10.1109/RDAAPS48126.2021.9452005.

[42] A. Wang, Y. Zhao, G. Li, J. Zhang, H. Wu, and Y. Iwahori, "Heterogeneous defect prediction based on federated reinforcement learning via gradient clustering," *IEEE Access*, vol. 10, pp. 87832–87843, 2022, doi: 10.1109/ACCESS.2022.3195039.

[43] L. Li, M. Duan, D. Liu, Y. Zhang, A. Ren, X. Chen, Y. Tan, and C. Wang, "FedSAE: A novel self-adaptive federated learning framework in heterogeneous systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Shenzhen, China, Jul. 2021, pp. 1–10, doi: 10.1109/IJCNN52387.2021.9533876.

[44] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "TiFL: A tier-based federated learning system," in *Proc. ACM HPDC*, Stockholm, Sweden, 2020, pp. 125–136.

[45] A. Wang, Y. Zhang, and Y. Yan, "Heterogeneous defect prediction based on federated transfer learning via knowledge distillation," *IEEE Access*, vol. 9, pp. 29530–29540, Feb. 2021, doi: 10.1109/ACCESS.2021.3058886.

[46] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, "FedAUX: Leveraging unlabeled auxiliary data in federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 5531–5543, Sep. 2023, doi: 10.1109/TNNLS.2021.3129371.

[47] S. M. Weiss, "Small sample error rate estimation for k-NN classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 3, pp. 285–289, Mar. 1991, doi: 10.1109/34.75516.

[48] K. J. Devi, G. B. Moulika, K. Sravanthi, and K. M. Kumar, "Prediction of medicines using LVQ methodology," in *Proc. Int. Conf. Energy, Commun., Data Anal. Soft Comput. (ICECDS)*, Aug. 2017, pp. 388–391, doi: 10.1109/ICECDS.2017.8390162.

[49] Y. Li, W. Zhao, and J. Pan, "Deformable patterned fabric defect detection with Fisher criterion-based deep learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1256–1264, Apr. 2017, doi: 10.1109/TASE.2016.2520955.

[50] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Robust classification with convolutional prototype learning," 2018, *arXiv:1805.03438*.

[51] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1600–1607, doi: 10.1109/CVPR.2012.6247852.

[52] M. Pang, B. Wang, M. Ye, Y.-M. Cheung, Y. Chen, and B. Wen, "DisP+V: A unified framework for disentangling prototype and variation from single sample per person," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 867–881, Feb. 2023, doi: 10.1109/TNNLS.2021.3103194.

[53] R. Zarei-Sabzevar, K. Ghiasi-Shirazi, and A. Harati, "Prototype-based interpretation of the functionality of neurons in winner-take-all neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 11, 2022, doi: 10.1109/TNNLS.2022.3155174.

[54] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "FedProto: Federated prototype learning across heterogeneous clients," in *Proc. AAAI*, Vancouver, BC, Canada, 2022, pp. 8432–8440.

[55] L. Zhou, M. Ye, D. Zhang, C. Zhu, and L. Ji, "Prototype-based multisource domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5308–5320, Oct. 2022, doi: 10.1109/TNNLS.2021.3070085.

[56] A. Li, P. Yuan, and Z. Li, "Semi-supervised object test via multi-instance alignment with global class prototypes," in *Proc. CVPR*, Jun. 2022, pp. 9809–9818, doi: 10.1109/CVPR52688.2022.00958.

[57] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.

[58] Z. Xu, J. Liu, Z. Yang, G. An, and X. Jia, "The impact of feature selection on defect prediction performance: An empirical comparison," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Ottawa, ON, Canada, Oct. 2016, pp. 309–320, doi: 10.1109/ISSRE.2016.13.

AILI WANG (Member, IEEE) was born in Tianjin, China, in 1979. She received the B.S., M.S., and Ph.D. degrees in information and signal processing from the Harbin Institute of Technology, Harbin, China, in 2002, 2004, and 2008, respectively. She joined the Harbin University of Science and Technology as an Assistant, in 2004 and she became an Associate Professor and a Master's Tutor with the Department of Communication Engineering, in 2010. She has been a Visiting Professor to do search of 3D polyp reconstruction with the Computer Science Laboratory, Chubu University, Japan, in 2014. Her research interests include image super resolution, image fusion, object tracking, software engineering, and reinforcement learning and federated learning.

LINLIN YANG received the bachelor's degree from the School of Electronic and Information Engineering, Henan Institute of Technology, in 2021. He is currently pursuing the master's degree with the School of Measurement and Control Technology and Communication Engineering, Harbin University of Science and Technology. His research interests include software engineering, prototype learning, and federated learning.

HAIBIN WU was born in Harbin, China, in 1977. He received the B.S. and M.S. degrees from the Harbin Institute of Technology, Harbin, in 2000 and 2002, respectively, and the Ph.D. degree in measuring and testing technologies and instruments from the Harbin University of Science and Technology, Harbin, in 2008. From 2009 to 2012, he held a post-doctoral position with the Key Laboratory of Underwater Robot, Harbin Engineering University. From 2014 to 2015, he was a Visiting Scholar with the Robot Perception and Action Laboratory, University of South Florida. His research interests include robotic vision, visual measuring and image processing, medical virtual reality, and photoelectric testing.

YUJI IWAHORI (Member, IEEE) was born in Japan, in 1959. He received the B.S. degree from the Nagoya Institute of Technology, in 1983, and the M.S. and Ph.D. degrees from the Tokyo Institute of Technology, in 1985 and 1988, respectively. He joined the Nagoya Institute of Technology, in 1988. He was a Professor with the Nagoya Institute of Technology, in 2002. He has also been a Research Collaborator with IIT Guwahati, since 2010. Since 2010, he has been engaged in research of endoscopic polyps with Aichi Medical University. His research interests include computer vision, image recognition, deep learning and neural networks, pattern recognition and pattern classification, bioinformatics, biomedical imaging and artificial intelligence applications, and research of medical images. He serves as a Peer-Reviewer for KES Journals, such as the IEEE Transactions on Neural Networks and Learning Systems, *Multimedia Tools and Applications* (MTAP) (Springer), and IEEJ.