Research on Few-Shot Defect Detection Algorithm Based on Federated Learning

Yufeng Xiong*

Department of Electronic Information, Ningbo University

Abstract: The algorithm based on deep learning has been widely used in defect detection in all walks of life, but the performance of the deep learning model depends mainly on rich annotation data. However, in the actual scene, obtaining large-scale, high-quality data to ensure users' privacy and safety is challenging, which limits its further promotion in specific application fields. To solve this problem, we propose a federated few-shot defect detection framework, which uses the privacy protection of the federated framework to jointly train independent few-shot tasks distributed on different clients to obtain a few-shot model that can quickly adapt to new tasks with limited data. We have done many experiments to evaluate our framework's effectiveness, and the results show that our framework is superior to the baseline and achieves the same performance as the model trained with a lot of data.

Keywords: Defect detection, Data privacy and safety, Federated framework, Few-shot learning.

1. INTRODUCTION

Surface defect detection plays an important role in the manufacturing process [1-5]. Undetected defects will affect the normal use of products and even lead to serious safety problems. Therefore, it is very important to realize an efficient and accurate defect detection system. However, it is challenging to accurately detect the surface defects of objects. As shown in Figure **1**, some defects are blurred and hard to be distinguished from the background. In addition, the data volume of the defect detection dataset in the industry is very limited. For many extreme cases, it is almost impossible to collect enough data. The scarcity of data leads to the over-fitting of the model and poor performance. This problem hinders the practical application of defect detection to some extent [6].

To solve the problem of insufficient labeled data, many researchers are committed to making the model have excellent generalization ability through the few-shot learning technique [7]. The core idea of the few-shot learning technique is to classify and locate the targets in the image through training a small number of labeled samples and to obtain a detection model with certain generalization ability by designing reasonable training methods, model structure, and loss function, to realize effective detection of targets in complex environment [8-11].

The success of deep learning is inseparable from a large number of marked data [12], but the process of obtaining data also brings a series of data security risks. In the era of big data, data scattered on various devices will be collected and trained in a centralized machine learning model, which may lead to the disclosure of users' privacy and cause concerns about the security of private information from all walks of life [13]. In this regard, researchers are thinking about how to ensure that data owners can combine the data of multiple users for model training without revealing their data privacy to provide them with more efficient, accurate, and safe models. Federated learning [14] came into being under this background. Different from the traditional machine learning model training, federated learning does not require all data to be centralized but only sends the model to each data source client, learns its own private data locally at the participants, and then aggregates the learning results of all parties to get the final global model.

Federated few-shot learning [15] refers to the joint training of independent few-shot tasks built on multiple clients by using the few-shot learning method in the federated learning scenario so as to absorb the knowledge from multiple clients and build a few-shot model that can quickly adapt to local tasks. Among them, federated learning can train the model without leaving the local user data, which fully guarantees data privacy; few-shot learning solves the problem of the weak generalization ability of the model caused by a very small number of samples. The effectiveness of the combination of the two has been verified by a large number of studies, but there are still the following challenges in this direction. Firstly, the heterogeneity of data leads to the difficulty of convergence of the federal model [16]. In the federated learning scenario, due to the different data sources of each participant, the local data distribution of each participant is quite different from the global data distribution, that is, the problem of federal data heterogeneity. This problem makes the local optimization goal and the global optimization goal of each participant deviate during the training of the federated model, thus affecting the convergence speed of the model and the final prediction performance of the model. Secondly, the few-shot task in the federal scene is more difficult. The difficulty of the few-shot

*Address correspondence to this author at the Department of Electronic Information, Ningbo University; E-mail: xyf_0321@163.com

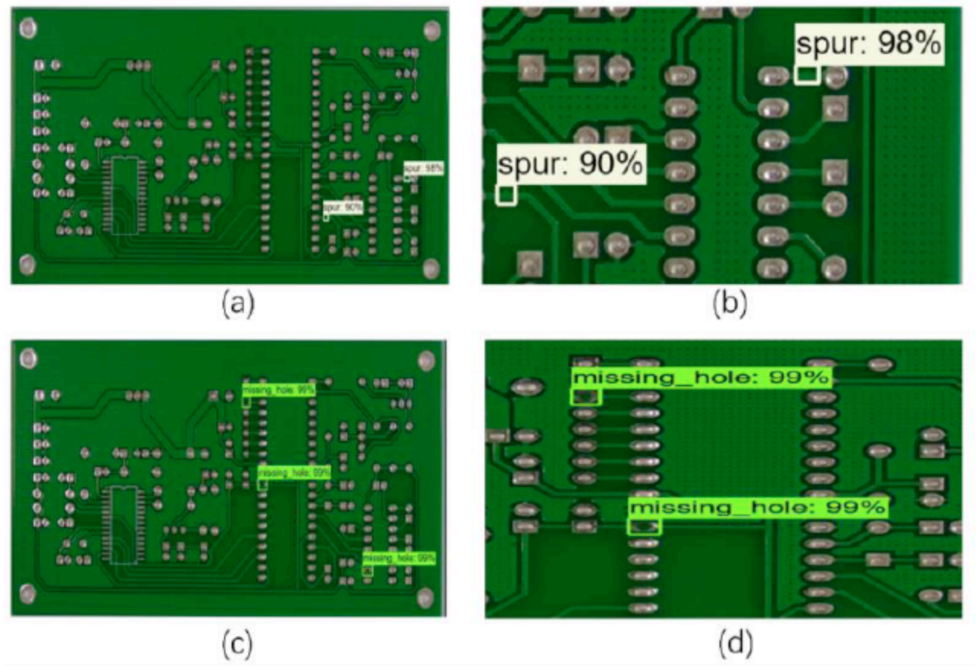

Figure 1: Examples of PCB surface defects. (**a**) and (**c**) show two types of defects—spur and missing-hole—while (**b**) and (**d**) are the corresponding close-up views. As shown in the figure, these defects are subtle and challenging to detect against their background.

classification task depends on the difference between different categories of samples. When the differences between different categories are large, the model is easier to distinguish, and vice versa. However, in the federal scenario, the data of a single participant is collected or generated from the same data source, which often leads to the sample categories of a single participant being closer to the global sample label space, making constructing the few-shot task more difficult. Difficult few-shot tasks will increase the difficulty of model learning and affect the generalization effect of the model.

To solve the above problems, we propose a federal few-shot learning framework: the server uses reinforcement learning [22] technology to build a new clustering algorithm [17-20] to aggregate client models with similar tasks, thus solving the problem of client data heterogeneity; In the client local model, the contrast learning technology [21] is used to improve the intra-class compactness and inter-class differences at the instance level, thus alleviating the problem of misclassification and further improving the accuracy of recognition. We have done a lot of experiments to evaluate the effectiveness of our framework. The results show that our framework is superior to the contrasted baseline and achieves the same performance as the model trained with a large number of data.

To sum up, our main contributions are summarized as follows.

(1) In this work, we propose a clustering algorithm based on reinforcement learning. In federated learning, data heterogeneity leads to the difficulty of global model convergence. The problem of data heterogeneity was solved by aggregating only client models with similar tasks. Using reinforcement learning to build a clustering algorithm to divide clients with similar tasks into the same cluster avoids the problems of selecting the initial cluster center of traditional clustering, determining the number of clusters, and determining the high computational complexity of the clustering algorithm.

(2) In the local model training, we found that many redundant candidate frames in the Faster-RCNN algorithm were extracted by the RPN (Region Proposal Network) network and input into the classification network in the second stage, which made the model training process slow and prone to misclassification. We use contrastive learning technology to improve the intra-class compactness of the same category and the inter-class differences of different categories and further find candidate boxes similar to labels, thus alleviating the impact of these issues and improving the accuracy of recognition. Through a large number of experimental tests, the effectiveness of our method is verified.

2. RELATED WORK

2.1. Defect Detection

The purpose of defect detection is to classify and locate related defects in images. Early methods are mainly based on hand-made features, such as directional gradient histogram [25] and accelerated robust features [26], and usually use some traditional machine learning algorithms, such as support vector machine [27-29], to distinguish defects from backgrounds. However, these methods are weak in extracting representative features in complex scenes and are often not suitable for practical projects. In recent years, deep learning algorithms have been widely used in defect detection because they can automatically learn a very discriminating representation by training models with marked data. A lot of work is based on some classic deep learning object detection frameworks, such as single shot detector (SSD) [30] and Faster R-CNN [23]. For example, Zeng *et al*. [31] used SSD to locate defective areas. Jin [32] used Faster R-CNN to identify related targets. Some researchers have proposed some effective modules and combined them with the classical detection framework. For example, Hao [33] introduced a feature fusion strategy in Faster R-CNN. Cheng and Yu [2] designed a new channel attention module, which improved the embedding of extraction. They designed a multi-layer feature fusion network to further improve the detection accuracy. These methods have achieved good results.

However, all the above methods need a lot of annotated training data. In practical defect detection applications, this requirement is usually difficult to meet. To solve this problem, we combine federated learning with few-shot learning to realize a federated few-shot defect detection framework in this paper. In the case of insufficient training data, the framework can still achieve competitive performance while ensuring the security of user data.

2.2. Few-Shot Detection

The few-shot target detection task aims to classify and accurately locate the targets in the image by training a small number of labeled samples so as to obtain a detection model with good generalization ability. The task can be described as: given datasets and, representing base class datasets, each category has enough labeled training samples to represent new class datasets, and each category has only a few labeled samples. The categories in the base class and the new class do not overlap, that is, $\mathrm{Cbase} \cap \mathrm{Cnovel} = \emptyset$. Given a test image, the category and position of N targets in this image are predicted. The goal of few-shot target detection is to predict the targets in the test image with the help of the prior knowledge learned in the annotated base class and a small number of new class training samples. Most work achieves this goal by designing strategies that transfer knowledge from similar task datasets to target task datasets. For example, Kang [8] proposed a re-weighting module to reflect the importance of various features. Fan [10] constructed a multi-relationship detector to take advantage of the similarity between different classes of samples. To sum up, the above methods are all based on meta-learning, which occupies a mainstream position in few-shot detection. However, these methods usually occupy a lot of memory to save pre-training data, so it is not convenient to apply them to practical projects.

After experimental analysis, Tian [34] thinks that the paradigm based on meta-learning may not be the best solution. The federal few-shot defect detection framework we proposed is a few-shot defect detection framework based on fine-tuning. The client uses local data to fine-tune the basic model sent by the server and does not need to store a large amount of training data locally. This framework is more convenient to implement.

2.3. Federated Learning

In order to solve the two major problems of data islands and privacy protection in the era of big data, Federated Learning [14] has gradually stepped onto the historical stage with the purpose of shifting the focus of artificial intelligence to algorithm architecture with privacy protection. In 2016, Google put forward the original definition of federated learning [35], that is, a machine learning model can be trained in a distributed manner without centralized data. In the federated learning scenario, we divide the training data into horizontal federated learning, vertical federated learning, and federated transfer learning according to the data feature distribution and sample ID distribution among different participants [36]. Horizontal federal learning is mainly suitable for federal learning. Participants' data have overlapping data characteristics. The data characteristics overlap among participants, but the data samples owned by participants are different. However, longitudinal federated learning is suitable for data samples with overlapping training data of federated learning participants. However, when the datasets owned by participants in federated learning are small in the intersection of samples and features, all participants can try to use federated transfer learning to train a better machine learning model cooperatively. This paper will mainly focus on horizontal federal learning.

Although federated learning can safely use the data information of all parties for model joint training in a distributed way, due to its training mode and setting, the data of all parties are usually non-independent and identically distributed; that is, there is data heterogeneity. Previous studies have shown that in heterogeneous data scenarios, local updates may lead to the performance degradation of the global model. As

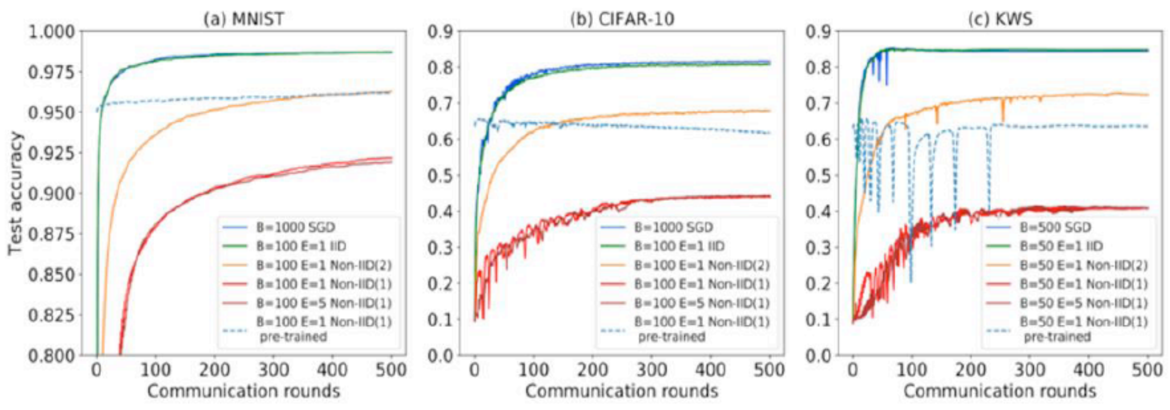

Figure 2: Experimental effect of FedAvg under IID and Non-IID data distribution [37].

shown in Figure **2**, Zhao [37] proved that the Non-IID data distribution has a great influence on the model accuracy of the FedAvg algorithm through related experiments. Under the Non-IID data distribution, the data distribution of each participant is quite different from the global data distribution, which leads to the inconsistency between the local optimization objectives and the global optimization objectives of each participant.

To solve this problem, we propose a clustering algorithm based on reinforcement learning, which divides clients with similar tasks into the same cluster. After the algorithm is completed, the federated aggregation algorithm FedAvg [35] is used to aggregate the model and send it to the clients so that clients with similar tasks can aggregate data with each other. After a large number of experiments, the algorithm significantly reduces the impact of Non-IID data on the global model and improves the generalization of the model.

3. METHODOLOGY

In this section, we explain the proposed federal few-shot defect detection framework and specific construction details. Specifically, section 3.1 introduces the whole process and learning plan. Then, sections 3.2 and 3.3 describe the structure of the adopted basic model in detail.

3.1. Overall Pipeline

In this section, we outline the proposed federal few-shot defect detection framework, as shown in the

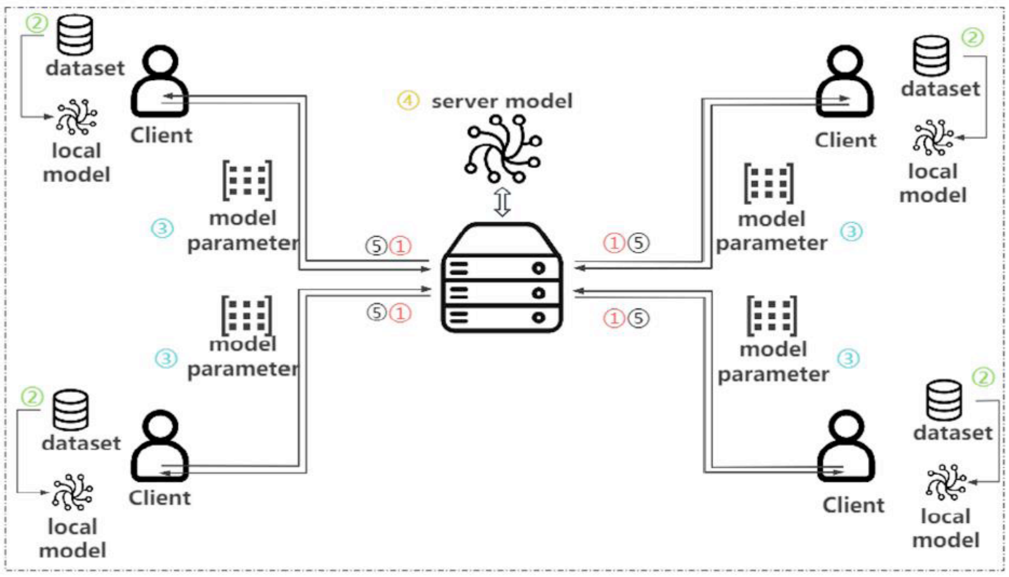

Figure 3: The overall training process of the model. The entire algorithm model proposed consists of 5 steps. Step 1: The server issues the basic model. The server trains the model using a public dataset and sends it to the client. Step 2: Client training. The client uses local sample data for model training. Step 3: The client uploads the model. The client uses the trained sample data as a local model representation and uploads it to the server. Step 4: Server aggregates client model parameters. The server uses the constructed clustering algorithm to aggregate clients with similar representation information. Step 5: Update. The client receives a model update from the server to update the local model. Repeat step four until the number of iterations or model convergence is reached.

Algorithm 1: The Training Process of the Federated Few-Shot Learning Model

```
Step 1: SERVERISSUEBASICMODEL()
//Server trains the basic model on public dataset and sends it to clients
Step 2: CLIENTTRAINING()
//Each client trains the model using local data
Step 3: CLIENTUPLOADMODEL()
//Clients upload locally trained models to the server
while maximum iterations not reached and model not converged do
    Step 4: SERVERAGGREGATEMODELS()
    //Server aggregates model parameters from clients
    Step 5: CLIENTUPDATEMODEL()
    //Server sends the updated model to clients for local updates
    if convergence criteria are met then
        Break
    else
        Continue to the next iteration
    end if
end while
```

figure. According to some previous work [1] and [33], we use Faster R-CNN [23] as our local model and replace the traditional method of region proposal [24] with RPN(Region Proposal Network) to realize complete end-to-end learning, thus speeding up the algorithm. FedAvg [35] aggregation algorithm is used in federated learning. FedAvg is a commonly used federated learning algorithm that aggregates model parameters through weighted averages. Its basic idea is to upload the parameters of the local model to the server, and the server calculates the average value of all model parameters and then broadcasts this average value back to all local devices. This process can be iterated many times until it converges, and it has the characteristics of low communication overhead and strong generalization. The entire framework process is shown in Figure **3** and the entire algorithm process is described in Algorithm **1**.

3.2. Optimization of Local Model

We use Faster R-CNN as the client's basic model. We find that in few-shot learning because there is only limited local data, the object is usually located accurately. However, in difficult few-shot learning, it is easy to be wrongly classified into other categories that are easily confused. Secondly, in the Faster-RCNN algorithm, many redundant candidate frames are extracted by the RPN network and input into the classification network in the second stage, which causes the model training process to slow down and lead to misclassification. Our goal is to reduce the instance-level similarity between similar objects with different categories of labels.

Specific methods. To learn contrastive object proposal encodings, we introduce a contrastive branch to guide the RPN features to learn contrastive-aware proposal embeddings. Further, proposals similar to the labeled label are found, which reduces the redundant candidate boxes to be input into the classification network in the second stage and increases the recognition accuracy. As shown in Figure **4**.

Similarity calculation. We use a bounding box classifier based on cosine similarity, in which the predicted i_{th} instance is the j_{th} class, which is calculated by cosine similarity between the feature vector z_i and the class weight vector z_j.

$$logit_{\{i,j\}} = \frac{z_i^T z_j}{||z_i||\cdot||z_j||} \tag{1}$$

Loss function. Our loss function is defined as follows. For a batch of feature maps

$\{z^i, y^i\}_{i=1}^{N}$, z_i where is the i_{th} feature map feature of the regional scheme, y_i is the label of foreground or background, y_i and τ is the nonparametric temperature, as shown in InfoNCE [38]. z_i and measure the cosine similarity between the i_{th} proposal and the j_{th} proposal in the projected hypersphere.

$$L = \frac{-1}{N}\sum_{j=1,j\neq i}^{N} \mathbb{I}\{y_i\}\cdot\log\frac{exp(\tilde{z_i}\cdot\tilde{z_j}/\tau)}{\sum_{K=1}^{N}\mathbb{I}_{k\neq i}\cdot exp(\tilde{z_i}\cdot\tilde{z_k}/\tau)} \tag{2}$$

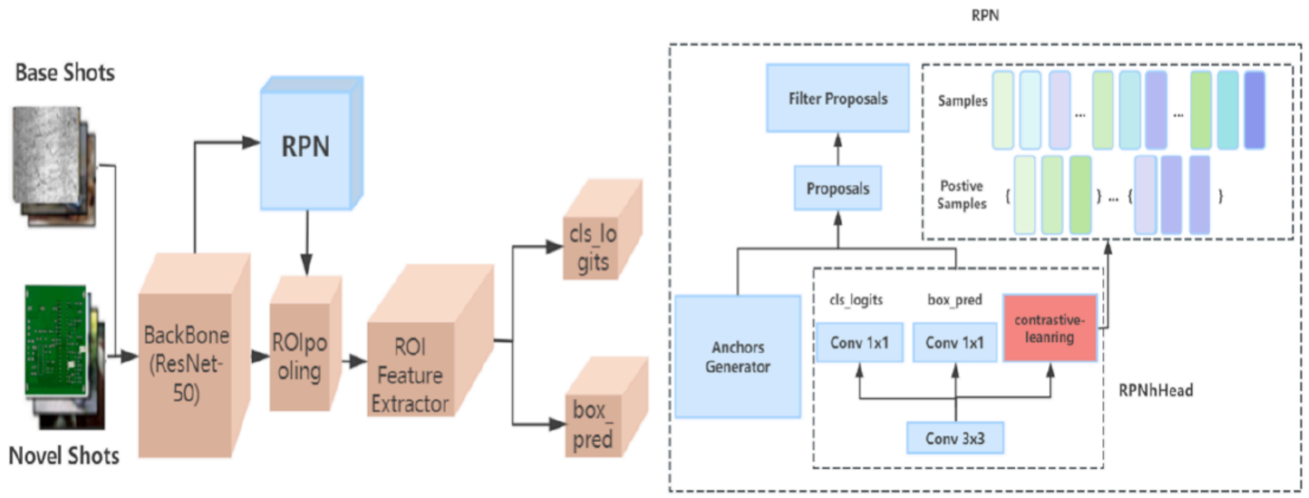

Figure 4: The overall structure of the local model. We introduced a contrastive learning branch in RPN to guide feature learning. We have designed a comparison objective to maximize intra-class consistency and cross-class inconsistency.

3.3. Reinforcement Clustering Algorithm

It is not the best choice to use the classical client-server federated learning architecture to train a globally shared model for scenarios where there are specific groups between clients, or the data distribution is obviously different. In contrast, it is more reasonable and effective to divide clients with similar data distribution into the same group and then train a federated learning model for each group, that is, to train a federated learning model for each client group with similar data distribution.

We combine the Q-learning algorithm and partition-based clustering algorithm in reinforcement learning and propose a reinforcement-based clustering algorithm (RCA). Each iteration of the algorithm is regarded as a state of the RCA algorithm; each client is each Agent. Think of the combination of different clients as different actions. When selecting actions, the mechanisms of "exploration" and "utilization" are applied, and a new design method of greed coefficient is put forward to explore more possibilities and prevent falling into local optimum. The environment takes the changing trend of intra-class distance as the basis for sending reinforcement signals to the Agent. After receiving the reinforcement signals, the Agent updates the cumulative reward value of each behavior according to the types of reinforcement signals. When the cumulative reward value of each behavior in the Q-table converges or reaches the maximum number of iterations, the algorithm ends, and the clustering results are output for federated aggregation.

Q-table construction. During the RCA algorithm operation, assuming that the number of input clients is n and the number of clusters is m, the Q-table with the size of $N * M$ will be established in the initialization stage of the algorithm to store the cumulative reward value continuously updated by each Agent during the algorithm iteration. Each client is associated with an Agent in the algorithm, and each Agent is associated with a row of Q values in the Q-table. The iterative process of the algorithm is the process of constantly updating the Q-table. The structure of the Q-table is shown in Table **1**.

Construction of reinforcement signal. The reinforcement signal is the feedback signal given to the Agent by the environment in the strong learning task. There are generally two kinds of reinforcement signals, namely, rewarding reinforcement signals and punitive

Table 1: The Structure of the Q-Table

Client (Agent)	$Action_1$	$Action_2$	…	$Action_m$
$Client_1$	R_{11}	R_{12}	…	R_{1m}
$Client_2$	R_{21}	R_{22}	…	R_{2m}
$Client_3$	R_{31}	R_{31}	…	R_{3m}
…	…	…	…	…
$Client_n$	R_{n1}	R_{n2}	…	R_{nm}

reinforcement signals. Intra-class distance refers to the mean square distance between sample points of the same class, which reflects the compactness of a cluster in the sample space. The smaller the intra-class distance, the more compact the samples in the cluster are, and the higher the similarity is. The greater the intra-class distance means that the samples in the cluster are more dispersed, the greater the span of each attribute value in the sample, and the lower the similarity between the samples. During the operation of the algorithm, when the intra-class distance becomes smaller, +1 represents the reward signal and -1 represents the punishment signal. The calculation formula of intra-class distance is as follows. $D_{(c_j)}$ represents the average within-class distance of cluster j_{th}, $|c_j|$ represents the number of samples in cluster j_{th}, M represents the dimension of input data, and x_{im} represents the m_{th} attribute value of sample i_{th}, c_{jm} represents the m_{th} attribute of cluster center j_{th}.

$$D_{c_j} = \frac{1}{|c_j|}\sum_{i=1}^{|c_j|}\sqrt{\sum_{m=1}^{M}(x_{i_m} - c_{j_m})^2} \tag{3}$$

Selection of greed coefficient. Greedy strategy is a method based on probability to compromise the process of "exploration" and "utilization" in reinforcement learning when the Agent chooses behavior. In traditional federated learning, the greedy coefficient is a fixed value, which does not take into account the accumulation of "knowledge" by the Agent in the process of "exploration". In the process of "exploration", the Agent will store the accumulated "knowledge" in the Q-table in the form of cumulative reward value. The cumulative reward value of the Agent for each behavior can reflect the similarity between the sample and each cluster to some extent. It can be roughly considered that the similarity between the sample and the cluster with the largest cumulative reward value is the highest, and the similarity with the cluster with the smallest cumulative reward value is the lowest. To make greater use of the knowledge stored in the Q-table, the greedy coefficient is set to a dynamic value that increases with the number of iterations. The method of greedy coefficient changing with the number of iterations is as follows. N_c represents the number of behaviors in the Q-table that have not been selected, and N_m represents the total number of behaviors in the Q-table.

$$\varepsilon = \frac{N_c}{N_m} \tag{4}$$

Operation process of the algorithm. Firstly, the required initialization parameters are input before the algorithm is operated. In the operation stage of the algorithm, all agents randomly select a behavior in the discrete behavior set during the first iteration, and the agents (clients) who choose the same behavior are grouped into a cluster; at the same time, the intra-class distance of each cluster is calculated and recorded. Starting from the second iteration, each Agent selects behaviors according to the ε-greedy strategy and calculates a new intra-class distance, which is compared with the intra-class distance obtained in the previous iteration. If the intra-class distance decreases, a reward signal is given to all agents in the cluster. If the intra-class distance increases, all Agents in the cluster are given punishment signals, and each Agent updates the cumulative reward of each behavior in the Q-table according to the type of reinforcement signal obtained until the change of the intra-class distance is less than the threshold or the algorithm stops iterating. Finally, the clients are outputted under each behavior (category). The entire algorithm process is described in Algorithm **2**.

4. EXPERIMENTS

In the part of algorithm design in the previous section, this section mainly verifies the algorithm on the federated few-shot learning task of each step in the defect detection task is verified by experiments and compared with related algorithms.

4.1. Data Preparation

The dataset used in this chapter is taken from the public PCB dataset commonly used in the field of defect detection, which is widely used in defect detection research in the manufacturing industry.

PCB defect dataset. It is a public synthetic PCB dataset published by Peking University, which contains 1386 images and 6 kinds of defects (missing holes, mouse bites, open circuits, short circuits, stray, and fake copper) for detection, classification, and registration tasks. In this paper, we selected 693 images that are suitable for the detection task, randomly selected 593 images as the training set, and 100 images as the verification set.

In addition to the above datasets, to verify the effectiveness of the improvement, this experiment also used the Deep-PCB defect dataset for the ablation experiment, and all the images in this dataset were obtained from linear scanning CCD. The dataset is divided into a training set and a verification set, in which the training set contains 700 pictures and the verification set contains 250 pictures. Each image has a resolution of 640×640 and contains 8 to 15 defect detection targets.

Algorithm 2: Reinforcement Learning Clustering Algorithm

```
Input:
The number of clients N
The number of clusters M
Maximum number of iterations Max_iter
Intra-class distance change threshold T
Result: Select the client label for the same behavior
--------------------------------------------------
Initialization:
Intra-class distance D = 0
Iterations iter = 0;
while $iter \leq Max\_iter$ and $D \leq T$ do
    // Random selection behavior;
    if $iter == 1$ then
        $action_{iter}$ = RANDOMCHOOSEACTION() ;
    end
    // Using greedy strategy to choose behavior;
    if $iter! = 1$ then
        $action_{iter}$ = CHOOSEACTION($\varepsilon$) ;
    end
    // Enhanced signal occurs;
    SENDSINGNAL($action_{iter}$);
end
```

4.2. Experimental Setup

To simulate the heterogeneous scene of the data domain in federated learning, we follow the popular strategy [39, 40], distribute samples to all clients according to Dirichlet distribution, and set the concentration parameter to 1.0. The experimental setup conditions used are shown in Table **2**.

Table 2: Experimental Condition Setting

Condition Type	Experimental Setup
Batch size	4
Training epochs	100
optimizer	Adam
Learning rate	0.0015
Weight decay	0.05
Input size	640×640
Confidence threshold	0.5

4.3. Evaluating Indicator

To evaluate the effectiveness of the improved network structure, this paper uses the following indicators to evaluate the model: mean average accuracy rate (AP), accuracy rate (Precision), and Recall rate. The mean average accuracy rate comprehensively considers the average accuracy rate of different categories, which can comprehensively evaluate the performance of multi-category target detection tasks. Accuracy (P) indicates the proportion of samples with positive prediction, while recall (R) indicates the proportion of positive samples with correct prediction. In the multi-class target detection task, a comprehensive performance evaluation index mAP is obtained by calculating the average accuracy rate (AP) of each class and taking the average value. The specific formula is as follows:

$$AP = \int_0^1 P(R)dR \quad (5)$$

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{K} \quad (8)$$

Among them, TP stands for True Positive, that is, the number of samples correctly predicted by the model is positive; FN represents False Negative, that is, the number of samples that the model failed to correctly predict as positive; FP stands for false positive, that is, the number of samples with positive model error prediction; mAP is the average accuracy of all categories; K is the number of categories. These indicators can comprehensively evaluate the performance of the model in the target detection task.

4.4. Ablation Experiment

In this part, we conducted ablation studies on the proposed federal few-shot model framework to validate the effectiveness of key designs in the framework. We use 20% of the data as the training set, and refer to Table **2** for other experimental settings. First, we get rid of the strategy of optimizing the local model by contrast learning, and we call this variant F1. Secondly, we remove the federated learning module so that the client model can't use the global knowledge in the server model to learn similar knowledge. We call this variant F2. The results of overall ablation research are shown in Table **3** and Table **4**. From the results, we observe that this method is superior to all variants, which verifies the effectiveness of the design in this method.

Table 3: Ablation Experiment Based on Dataset PCB

Model	mAP@0.5	Precision/%	Recall/%
this method	**0.79**	**0.74**	**0.72**
F1	0.62	0.60	0.59
F2	0.72	0.70	0.66

Table 4: Ablation Experiment Based on Deep-PCB Dataset

Model	mAP@0.5	Precision/%	Recall/%
this method	**0.79**	**0.75**	**0.78**
F1	0.61	0.61	0.64
F2	0.72	0.70	0.72

4.5. Contrast Experiment

We train the model on the constructed PCB training set and evaluate its performance on the corresponding validation set. Simultaneously, we introduce several single-stage and two-stage object detection algorithms, including SPP-Net [41], Faster-RCNN, SSD, and YOLO-V5 [42], to compare their performance on few-shot tasks. Importantly, all comparison models are initialized with random weights, while the methods proposed in this chapter leverage weights pre-trained on ImageNet, a widely-used image dataset. The specific results are shown in Table **5**. With a full training set, our method exhibits a slight advantage over all other target detection algorithms in terms of mAP. As the number of training samples decreases, the performance of all models declines, emphasizing that fewer samples result in each sample contributing more

Table 5: Contrast Experiment Based on PCB Dataset

Proportion	Performance index	Detection algorithm				
		SPP-Net	Faster-RCNN	SSD	YOLO-V5	Ours
100%	mAP@0.5	0.62	0.87	0.77	0.88	**0.90**
	Precision/%	0.67	0.88	0.79	0.91	**0.89**
	Recall/%	0.66	0.85	0.77	0.89	**0.90**
	FPS	11	15	45	55	**14**
20%	mAP@0.5	0.47	0.63	0.60	0.72	**0.79**
	Precision/%	0.60	0.62	0.61	0.68	**0.74**
	Recall/%	0.61	0.65	0.65	0.67	**0.68**
	FPS	11	15	45	55	**14**
5%	mAP@0.5	0.19	0.33	0.47	0.42	**0.53**
	Precision/%	0.40	0.56	0.52	0.50	**0.56**
	Recall/%	0.42	0.51	0.50	0.51	**0.51**
	FPS	11	15	45	55	**14**

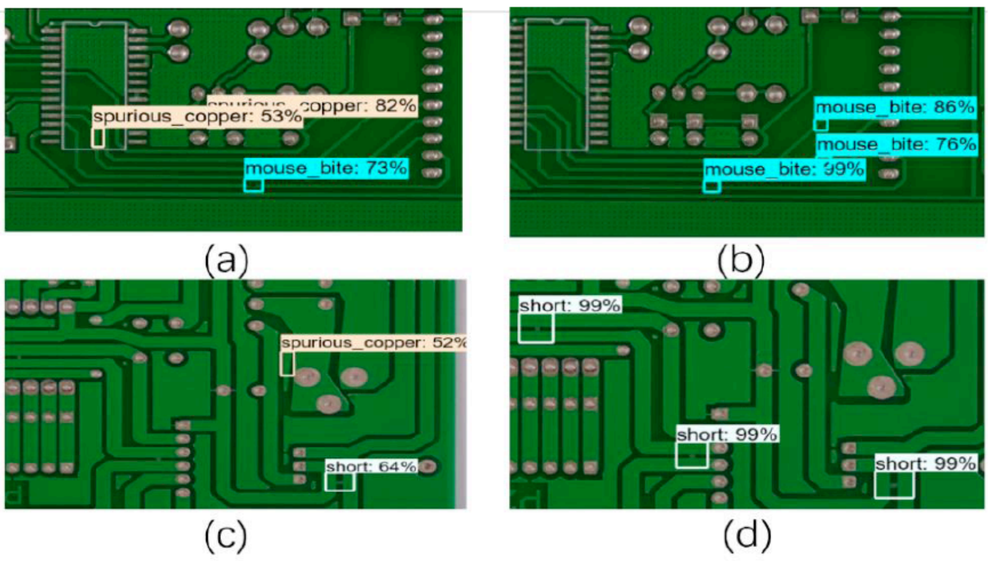

Figure 5: Improved defect detection results.

significantly to the model's performance, thereby causing a more considerable performance drop. Under few-shot conditions, our algorithm shows distinct advantages, particularly with 20% and 5% of the training samples. Furthermore, compared to other detection algorithms, our method demonstrates the smallest performance decline as the number of training samples reduces, highlighting its superior capability in handling few-shot learning tasks.

Figure **5** shows the performance of the improved algorithm in dealing with different defects, such as mouse bite and short circuit. The results show that the improved algorithm can more accurately capture the missed targets detected by the original algorithm in the detection process and significantly improve the problem of missing detection of small target defects by the original algorithm. In addition, for the targets that can be detected by both models, the improved algorithm also shows a higher level of confidence, which shows that the improved algorithm has significantly improved the detection accuracy of small targets in the same complex background.

Generally speaking, through this visual analysis, it is confirmed that the algorithm has obviously improved the accuracy of PCB defect identification and detection in complex backgrounds, which further verifies the effectiveness and substantial improvement of the improved algorithm.

5. CONCLUSION

In this paper, the problem of federated few-shot learning is studied, and its purpose is to learn a federated few-shot model that can obtain satisfactory performance in new tasks under the condition of limited labeled samples. However, due to the challenges of global data differences and insufficient local data, federated few-shot learning is still difficult to achieve. To address these challenges, we propose a new federated few-shot learning framework. Especially, we use reinforcement learning-based clustering algorithms to aggregate clients with similar tasks and then perform model aggregation on models belonging to the same cluster, greatly avoiding the problems caused by client data heterogeneity. Then, we utilized contrastive learning to identify candidate boxes that belong to similar categories to the labels further, alleviating the problem of misclassification. We conducted extensive experiments on few-shot learning datasets in a federated scenario, and the experimental results further validated that our framework outperforms other state-of-the-art baselines.

REFERENCES

[1] Luo Q, Fang X, Liu L, *et al*. Automated visual defect detection for flat steel surface: a survey. IEEE Transactions on Instrumentation and Measurement 2020; 69: 626-644. https://doi.org/10.1109/TIM.2019.2963555

[2] Cheng X, Yu J. Retinanet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection. IEEE Transactions on Instrumentation and Measurement 2021; 70: 1-11. https://dx.doi.org/10.1109/tim.2020.3040485

[3] Chen R, Yu G, Qin Z, *et al*. Patch matching for few-shot industrial defect detection. IEEE Transactions on Instrumentation and Measurement 2024; 73: 1-11. https://dx.doi.org/10.1109/TIM.2024.3413170

[4] Song G, Song K, Yan Y. EDRNet: encoder–decoder residual network for salient object detection of strip steel surface defects. IEEE Transactions on Instrumentation and Measurement 2020; 69: 9709-9719. https://dx.doi.org/10.1109/tim.2020.3002277

[5] Gao Y, Lin J, Xie J, *et al*. A real-time defect detection method for digital signal processing of industrial inspection applications. IEEE Transactions on Industrial Informatics 2021; 17: 3450-3459. https://dx.doi.org/10.1109/tii.2020.3013277

[6] Li Z, Wang H, Swistek T, *et al*. Enabling the network to surf the internet. arXiv.cs.CV: 2102.12205 2021. https://doi.org/10.48550/arXiv.2102.12205

[7] Li F, Rob F, Pietro P. A bayesian approach to unsupervised one-shot learning of object categories. Proceedings Ninth IEEE International Conference on Computer Vision 2003; 1134-1141. https://doi.org/10.1109/ICCV.2003.1238476

[8] Kang B, Liu Z, Wang X, *et al*. Few-shot object detection via feature reweighting. Proceedings of the IEEE/CVF international conference on computer vision 2019; 8420-8429. https://dx.doi.org/10.1109/iccv.2019.00851

[9] Yan X, Chen Z, XU A, *et al*. Meta r-cnn: towards general solver for instance-level low-shot learning. Proceedings of the IEEE/CVF International Conference on Computer Vision 2019; 9577-9586. https://dx.doi.org/10.1109/iccv.2019.00967

[10] Fan Q, Zhuo W, Tang C K, *et al*. Few-shot object detection with attention-rpn and multi-relation detector. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020; 4013-4022. https://dx.doi.org/10.1109/cvpr42600.2020.00407

[11] Perez R, JM, Zhu X, *et al*. Incremental few-shot object detection. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020; 13846-13855

[12] Agboka KM, Tonnang H E Z, Abdel-Rahman E M, *et al*. Data-driven artificial intelligence (AI) algorithms for modelling potential maize yield under maize–legume farming systems in East Africa. Agronomy 2022; 3085. https://doi.org/10.3390/agronomy12123085

[13] Liu C. Research on the perfection of personal information legal protection in the age of big data. Open Journal of Legal Science 2023; 11: 1445-1451. https://dx.doi.org/10.12677/ojls.2023.113206

[14] Mcmahan HB, Moore E, Ramage D, *et al*. Communication-efficient learning of deep networks from decentralized data. Artificial intelligence and statistics. PMLR 2017; 1273-1282. https://doi.org/10.48550/arXiv.1602.05629

[15] Fan C, Huang J. Federated few-shot learning with adversarial learning. 19th International Symposium on Modeling and Optimization in Mobile. IEEE 2021; 1-8. https://doi.org/10.48550/arXiv.2104.00365

[16] Zhao Y, Li M, Lai L, *et al*. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 2018 https://doi.org/10.48550/arXiv.1806.00582

[17] Xu D, Tian Y. A comprehensive survey of clustering algorithms. Annals of Data Science 2015; 165-193. https://dx.doi.org/10.1007/s40745-015-0040-1

[18] Cao H, Jia L, Si G, *et al*. A clustering-analysis-based membership functions formation method for fuzzy controller of ball mill pulverizing system. Journal of Process Control 2013; 34-43. https://dx.doi.org/10.1016/j.jprocont.2012.10.011

[19] Jinyin C, Xiang L, Haibing Z, *et al*. A novel cluster center fast determination clustering algorithm. Applied Soft Computing 2017; 539-555. https://doi.org/10.1016/j.asoc.2017.04.031

[20] Narendra, Kumpati S, Thathachar M, *et al*. Learning automata-a survey. IEEE Transactions on systems, man, and cybernetics 1974; 323-334. https://doi.org/10.1109/TSMC.1974.5408453

[21] Chen T, Kornblith S, Norouzi M, *et al*. A simple framework for contrastive learning of visual representations. International conference on machine learning. PMLR 2020; 1597-1607. https://doi.org/10.48550/arXiv.2002.05709

[22] Mnih V, Kavukcuoglu K, Silver D, *et al*. Human-level control through deep reinforcement learning. Nature, 2015; 518: 529-533. https://dx.doi.org/10.1038/nature14236

[23] Girshick R. Fast R-CNN. arXiv preprint arXiv:1504.08083 2015; 1440-1448. https://dx.doi.org/10.1109/iccv.2015.169

[24] Ren S, He K, Girshick R, *et al*. Faster R-CNN: towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497 2017; 1137-1149. https://dx.doi.org/10.1109/tpami.2016.2577031

[25] Dalal N, Triggs B. Histograms of oriented gradients for human detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2005; 1: 886-893. https://dx.doi.org/10.1109/cvpr.2005.177

[26] Bay H, Tuytelaars T, Van Gool, *et al*. SURF: speeded up robust features. Lecture Notes in Computer Vision 2006; 404-417.

[27] Mao T, Ren L, Yuan F, *et al*. Defect recognition method based on hog and SVM for drone inspection images of power transmission line. International Conference on High Performance Big Data and Intelligent Systems 2019; 254-257. https://dx.doi.org/10.1109/hpbdis.2019.8735466

[28] Zhao C, Chen Y, MA J. Fabric defect detection algorithm based on MFS and SVM. International Conference on Image and Video Processing and Artificial Intelligence 2018; 10836: 77-82. https://doi.org/10.1117/12.2513987

[29] Pasadas D J, Ramos H G, Feng B, *et al*. Defect classification with SVM and wideband excitation in multilayer aluminum plates. IEEE Transactions on Instrumentation and Measurement 2020; 241-248. https://doi.org/10.1109/TIM.2019.2893009

[30] Liu W, Anguelov D, Erhan D, *et al*. SSD: single shot multibox detector. Springer International Publishing 2016; 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

[31] Zeng W, You Z, Huang M, *et al*. Steel sheet defect detection based on deep learning method. International Conference on Intelligent Control and Information Processing 2019; 152-157. https://doi.org/10.1109/ICICIP47338.2019.9012199

[32] Jin X, Wang Y, Zhang H, *et al*. Dm-ris: deep multimodel rail inspection system with improved MRF-GMM and CNN. IEEE Transactions on Instrumentation and Measurement 2019; 1051-1065. https://doi.org/10.1109/TIM.2019.2909940

[33] Hao R, Lu B, Cheng Y, *et al*. A steel surface defect inspection approach towards smart industrial monitoring. Journal of Intelligent Manufacturing 2021; 32: 1833-1843. https://doi.org/10.1007/s10845-020-01670-2

[34] Tian Y, Wang Y, Krishnan D, *et al*. Rethinking few-shot image classification: a good embedding is all you need. Computer Vision 2020; 266-282. https://doi.org/10.1007/978-3-030-58568-6_16

[35] Mcmahan, Moore E, Ramage D, *et al*. Communication-efficient learning of deep networks from decentralized data. Artificial intelligence and statistics 2017; 1273-1282. https://doi.org/10.48550/arXiv.1602.05629

[36] Yang Q, Liu Y, Chen T, *et al*. Federated machine learning: concept and applications. ACM Transactions on Intelligent Systems and Technology 2019; 10: 1-19. https://doi.org/10.1145/3298981

[37] Zhu H, Xu J, Liu S, *et al*. Federated learning on non-iid data: a survey. Neurocomputing 2021; 465: 371-390. https://dx.doi.org/10.1016/j.neucom.2021.07.098

[38] Oord, Li Y, Vinyals, *et al*. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 2018. https://doi.org/10.48550/arXiv.1807.03748

[39] Hsu, QI H, Brown, *et al*. Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 2019. https://doi.org/10.48550/arXiv.1909.06335

[40] Yu, Bagdasaryan, Eugene, *et al*. Salvaging federated learning by local adaptation. arXiv preprint arXiv:2002.04758 2020. https://doi.org/10.48550/arXiv.2002.04758

[41] He, Zhang X, Ren S, *et al*. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE

Transactions on Pattern Analysis and Machine Intelligence 2015; 37: 1904-1916.
https://doi.org/10.1109/TPAMI.2015.2389824

[42] Redmon J, Divvala S, Girshick R, *et al*. You only look once: unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition 2016; 779-788.
https://doi.org/10.48550/arXiv.1506.02640

Received on 02-10-2024 Accepted on 15-11-2024 Published on 17-11-2024

https://doi.org/10.31875/2409-9694.2024.11.08