Performance Evaluation of a Decentralized Learning Architecture for PCB Defect Classification

***Abstract*—Printed circuit board (PCB) defect detection, which is an important task in industrial factories, receives great attention from both researchers and practitioners. To achieve high detection accuracy, the traditional training method requires data collection from multiple industrial factories. However, in practice, factories possess their own data and do not want to share the private data with other participants. Therefore, we introduce a decentralized learning method that makes use the knowledge of clients in the system. By leveraging the federated learning technique, a consensus global detection model can be produced while maintaining data privacy. We have conducted extensive experiments to evaluate the detection performance under various learning methods: federated learning, centralized learning, and local learning. We also compare the detection performance of two well-known detection models: YOLOv5 and YOLOv8. The experimental results show that the federated learning based method yields better detection performance than the local learning.**

***Index Terms*—PCB defect detection, federated learning, deep learning**

I. Introduction

A printed circuit board (PCB) is a fundamental component in modern electronics, playing as a platform to connect and support electronic components. To make sure that PCB operates properly, PCB defects need to be detected before deploying PCB in real devices. Common PCB defects include missing holes, open circuits, short circuits, spur, mouse bite, and breakout, which can be detected using the naked eye. Thank to the advanced development of computer vision, the PCB defect detection model can be constructed using deep learning models, thus supporting automatic verification at manufacturing sites and enhancing the PCB defect detection performance. In this work, we assume that there are multiple factories and each factory obtains a set of PCB defects data. Our work aims to answer the following question: *How to train a PCB fault detection model using multiple supervised data from various distributed factory sites?*

To build a PCB defect detection model, the traditional centralized training method first requires supervised data to be shared with the server [1]. Then, the detection model is trained at a centralized server by minimizing a custom loss function via multiple training epochs. The training procedure is terminated when there is no improvement in detection performance. Although the PCB fault detection model shows high performance in localizing and detecting PCB defects, the centralized training method suffers from the data privacy problem. Moreover, sending a large amount of data to the server requires high communication bandwidth, which may be difficult, especially for edge computing devices.

To overcome the problem of data privacy and high communication bandwidth in the conventional training method, we leverage the federated learning (FL) paradigm that allows collaborative training of a global deep learning model without directly exchanging the training data [2], [3]. Particularly, at the beginning of training, a global model, which can be pre-trained, is selected and shared with all participating factories. Then, at each training epoch, clients update their models using the private data and then send the updated local models to the server. Finally, the server performs model aggregation before distributing a new global model to other clients. If training properly, the FL architecture can produce similar performance compared to the centralized training method while protecting the data privacy of participants. Therefore, we introduce and evaluate the distributed learning-based PCB defect detection architecture.

To verify the proposed FL-based PCB defect detection model, we select a PCB defect dataset [4] with 693 synthetic images and more than 2,500 faults instances. The YOLOv5 and YOLOv8 models are selected for the experiment due to the feasibility of these models at edge devices. In our experiments, three different learning strategies are considered: centralized, federated, and local learning. In centralized learning, the data from local devices is shared with a central server where the defect detection model is trained. Meanwhile, in federated learning, a global model is created at the central server without sharing private data between clients. Finally, in local learning, a local detection model is trained using only the private dataset at the local device.

First, the performance of centralized detection models based on YOLOv5 and YOLOv8 is evaluated. The performance results show that YOLOv8 achieves higher detection accuracy and converges faster than YOLOv5. Therefore, YOLOv8 is used as the default detection model for other experiments. Then, we analyze the accuracy of the centralized YOLOv8 detection model on a real board. Finally, we compare the detection performance of the YOLOv8 model using two learning methods: federated learning and local learning. The FL-based detection model shows superior performance to the local model, which is trained using the private dataset.

The rest of the paper is constructed as follows. Section II provides a brief introduction to the YOLOv5 model for object detection followed by federated learning-based models for fault detection. Then, we explain the PCB defect detection architecture based on distributed learning in Section III followed by the performance evaluation in Section IV. Finally, we give the conclusion of our work and discuss the potential work of PCB defect detection in Section V.

II. Related Work

In this section, we analyze the underlying detection architecture and the training procedure of the YOLOv5 detection

model. Then, we summarize some studies related to the FL-based fault detection problem in industry and PCB fault detection models.

A. YOLOv5 Architecture

There are two types of object detection models: two-stage and one-stage detectors. In both detectors, we have a backbone network to extract features from images and reduce feature dimension. Common backbone networks are VGG [5], CSP-Darknets [6], ResNet [7]. In two-stage detectors, region proposals are first generated and followed by object classification for each region proposal. Meanwhile, instead of the region proposal generator, the one-stage detectors contain a detection generator after the feature extraction block or model head. Therefore, two-stage detectors usually achieve higher accuracy but slower than one-stage detectors. In this work, we focus on one-stage detectors such as YOLOv5 or YOLOv8 since they are suitable for edge-devices at industrial factories.

The architecture of YOLO v5 is shown in Figure 1. The YOLOv5 model consists of 3 components: CSP-Darknet53 as a backbone, SPP and PANet in the model neck, and the head of three convolutional layers. CSP-Darknet53 is just the convolutional network Darknet53 used as the backbone for YOLOv3 to which the authors applied the Cross Stage Partial (CSP) network strategy. CSP network makes use of the advantage of DenseNet's feature reuse characteristics while reducing the great amount of redundant gradient paths. CSP divides the input feature maps into two parts and only one part will go to the dense block, thus reducing the model complexity.

Spatial Pyramid Pooling (SPP) block concatenates the information from the inputs and returns an output with a fixed length. SPP allows us to increase the receptive field without sacrificing network speed. There are three max-pooling blocks in SPP, which generate feature maps with different sizes. These feature maps are combined before feeding into a convolutional block. The model head is composed of three convolutional layers with the number of channels $c = n_{anchors}(5+n_{classes})$, where $n_{anchors}, n_{classes}$ are the number of anchors and the number of classes. $n_{anchors}$ is typically set to 3. The combination of outputs of these convolutional layers produces the final detection including the localization and classification.

The loss function l used to train YOLOv5 contains three main components: Bounding box regression loss, objectness loss, and classification loss. We explain each component of the loss function based on the code provided at https://github.com/ultralytics/yolov5. The bounding box regression loss L_{GIOU} or the localization loss is computed as:

$$L_{GIOU} = \lambda_{loc}\left(1 - (IOU - \frac{C \setminus (A \cup B)}{C})\right) \quad (1)$$

where IOU is the intersection over union ratio between the predicted bounding box A and the truth bounding box B, which indicates localization accuracy of predictions. Meanwhile, C is the smallest enclosing box of A and B; λ_{loc} is the coefficient for the localization loss.

The classification loss measures the inaccuracy of the classification task using the binary cross-entropy loss as below:

$$L_{class} = \lambda_{class}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{i,j}^{obj}\sum_{c=1}^{n_{classes}}\Big(p_i(c)\log\hat{p}(c) + (1-p_i(c))\log(1-\hat{p}(c))\Big) \quad (2)$$

where λ_{class} is the coefficient for the classification loss, S^2 is the number of grids, and B is the number of anchors. If the anchor box at grid (i, j) contains at least a target, then the value $I_{i,j}^{obj}$ is set to 1; otherwise, the value is 0. $\hat{p}(c)$ indicates the predicted probability of the target, and $p_i(c)$ is the true value of the category.

Finally, the objectiveness loss is also based on binary cross-entropy loss and it is used to measure the error in detecting if an object appears in a grid cell.

$$L_{obj} = \lambda_{obj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{i,j}^{obj}\Big(IOU_{i,j}\log\hat{p'} + (1-IOU_{i,j})\log(1-\hat{p'})\Big) \quad (3)$$

where λ_{obj} is defined as the coefficient of the objectiveness loss and $IOU_{i,j}$ is the intersection over union ratio at grid cell i and anchor j. Meanwhile, $\hat{p'}$ is the predicted object probability at grid cell i and anchor j.

For performance evaluation, we use the following metrics: precision, recall, mAP50, and mAP50-95. Precision presents the percentage of correct predictions and all positive predictions. Meanwhile, recall indicates the percentage of predicted defects and the actual defects. mAP50 is defined as the mean of average precision calculated at the intersection over union (IoU) threshold of 0.5. Finally, mAP50-95 refers to the mean of mAP calculated when the threshold changes from 0.5 to 0.95.

B. Federated Learning for Defect Detection Problem

In the literature, some research papers implemented FL for the fault detection problem such as [8]–[10]. In [8], they argued that the existing data-driven machinery fault diagnosis methods required the collection of high-quality supervised data for training. In fact, it is difficult to collect data from multiple factories and send the data to a centralized server due to data privacy concerns. To improve the distributed learning performance, a validation set is used to verify the model update made by each participating client. If a local model produces low validation performance, it may not be used for model aggregation at the centralized server. The proposed FL architecture was evaluated using two rotating machinery datasets and performance results are promising in terms of data privacy and distributed learning.

Jiang *et. al* introduced the problem of insulator fault detection using a data privacy-preserving FL architecture. Two different networks including fully connected neural networks and convolutional neural networks are evaluated for insulator

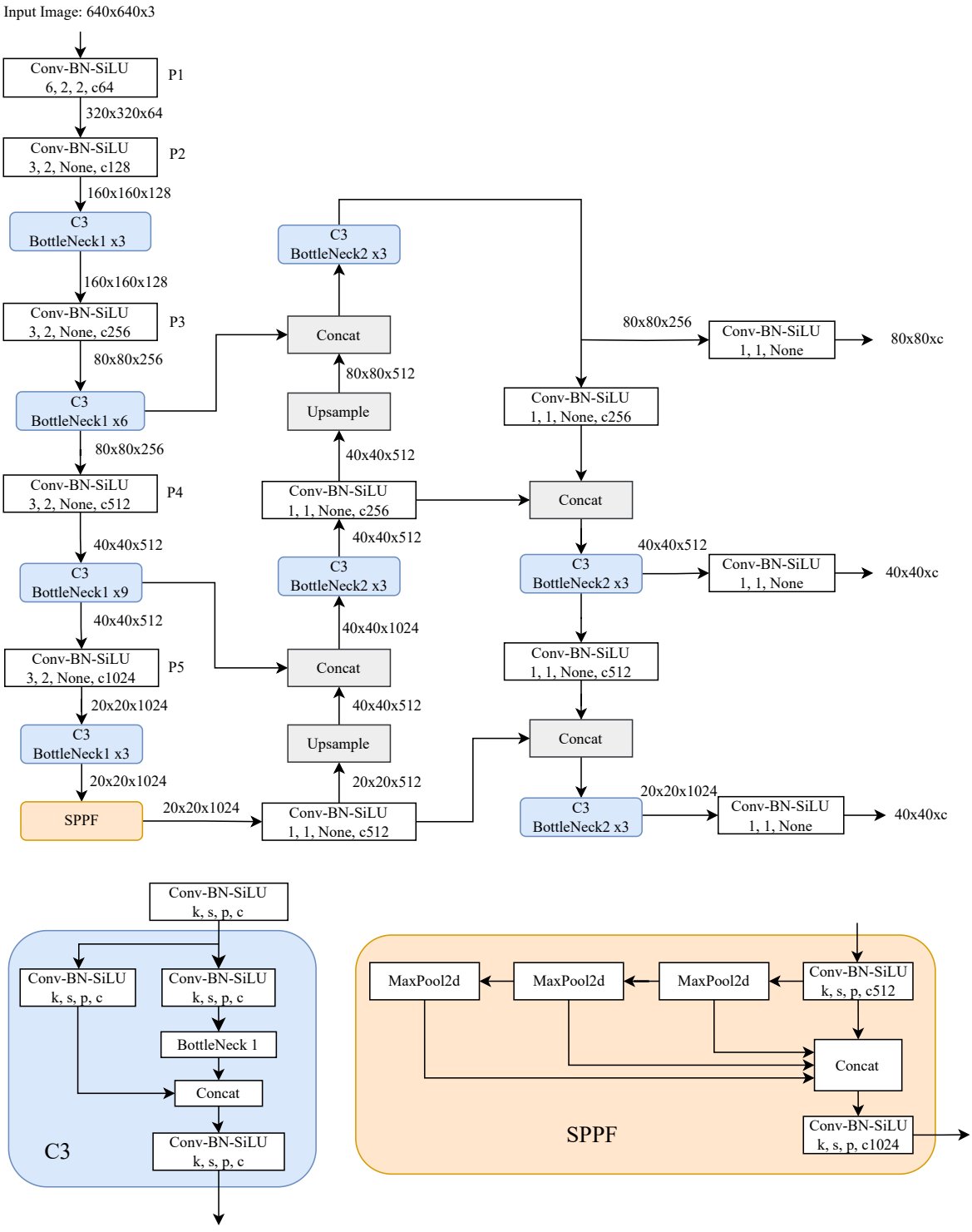


Fig. 1: Architecture for PCB Defects Detection

fault detection. The distributed learning method achieves similar performance on a collected insulator fault detection dataset compared to the centralized learning one.

A wind turbine fault detection method based on FL that jointly trains a global fault detection model was presented in [10]. Unlike conventional training, the FL-based model does not require communication data shared by local wind turbines. In addition, a multi-scale residual attention network model is used to extract multi-scale spatial features from raw sensor data that are related to each fault. The designed decentralized training paradigm shows similar performance to the conventional learning method while protecting data privacy from wind turbines.

C. Research on PCB Defect Detection

The authors in [11] summarize multiple PCB defect detection methods based on image processing, machine learning, and deep learning. Image processing detection approaches can be classified into three categories: reference comparison, non-reference inspection, and hybrid inspection methods. The most widely used method in the manufacturing industry is reference comparison in which we compare the difference between the reference and the examined images to find defects in PCBs. Meanwhile, in the non-reference approach, a predefined closed-form expression is proposed to detect PCB defects. Finally, the hybrid method combines both reference comparison and non-reference inspection methods. In machine learning-based methods, features are mainly extracted by traditional image processing operations such as edge detection, morphological processing, various image thresholding techniques. Then, these features are sent to machine learning models such as support vector machine, neural network, genetic algorithm, and decision tree. As a result, the detection accuracy of machine learning-based methods highly depends on extracted features.

Recently, deep learning-based methods, especially convolutional neural networks (CNNs), have been widely used in image processing, object detection, and segmentation [12]. Unlike traditional machine learning methods, CNN-based approaches can automatically extract image features, thus enhancing detection accuracy and speed. Particularly, CNN-based object detection algorithms are usually robust to noise. Therefore, deep learning algorithms have been widely used by researchers for PCB defect detection to achieve high detection performance. Zheng *et al.* [13] add successive convolutional modules into the MobileNetV2 architecture and this integration, combined with an improved skip connection, significantly improves detection speed and accuracy compared to VGG-16 and ResNet-50 models. Lim *et al.* [14] developed a novel multi-scale feature pyramid network based on YOLOv5 to tackle the detection of PCB defects. They also incorporated the CIoU loss function to precisely determine the spatial parameters, effectively finding the exact locations of these imperfections. Furthermore, Yu *et al.* [15] introduced a lightweight and efficient network architecture specifically tailored for detecting PCB defects. They introduced the diagonal feature pyramid mechanism to obtain feature maps to enhance fault detection. Additionally, they devised a multi-scale necking network to accommodate defects of different sizes.

III. Federated Learning for PCB Defect Classification in Industrial Factories

We present a cross-silo FL-based architecture for PCB fault detection that can be used at multiple distributed factories. There are two computing levels: edge and cloud. Assume that the edge layer contains multiple industrial factories where electronic devices are manufactured such as mobile phones, computers, transmitters, and receivers as shown in Figure 2. The global detection model is built with the support of a cloud server where an aggregation algorithm is used to update the global model at each training epoch. At the participating factories, there is a requirement for the detection of any fault caused by electronic devices. Each company or factory usually possesses a private PCB defects dataset and they do not want to send their data to other companies. Instead, an aggregation global model is constructed based on local model updates from all factories without sharing the raw datasets with other participants.

The training procedure is described as follows. At the beginning of the training, the cloud server selects and initializes a global model. We assume that a pre-trained global model is used to accelerate the training process. Then, the initial global model is distributed to all participating factories in the network. At a training epoch, each factory trains the detection model using its private dataset and then sends the model update to the cloud server. The model update may be the parameters gradient or the real network parameters. In the cloud server, after receiving the model updates from all participants, an aggregation model such as FedAvg [16], FedProx [17], or SCAFFOLD [18] is used to generate a newly consensus global model. A similar procedure continues at the next training epoch until the global model converges or the number of training epochs exceeds a threshold value.

After obtaining a consensus global model, the cloud server shares this global model with all industrial factories for the inference phase. The PCB defect detection model is executed at each factory site close to the product manufacturing line. Therefore, the detection time is expected to be lower than the case of sending raw images of PCB to external devices for examination. Moreover, since the global model is generated by using model updates from all participating factories, the detection performance is enhanced compared to the local learning technique, which is built using the dataset from only one factory. In summary, by using two computing layer FL-based architecture, we leverage the computing capacity of both layers while maintaining data privacy and achieving more accurate detection performance and lower detection time than the local learning model.

IV. Performance Evaluation

In this section, we first present the dataset for performance evaluation followed by performance comparison between centralized detection models based on YOLOv8 and YOLOv5. We also show the visualization of PCB defect detection on a

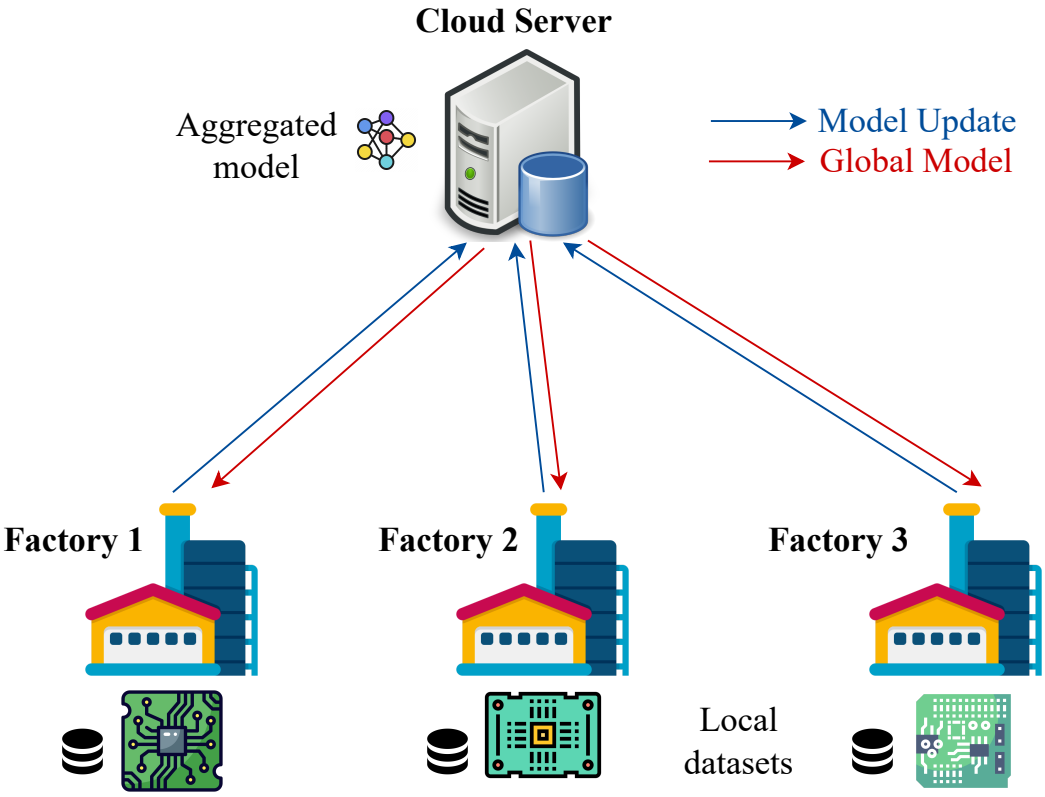


Fig. 2: Federated Learning for PCB Defects Detection

real PCB. Then, the performance of the federated and local learning models is verified and compared. The experiments are conducted on a PC with Intel Core i7-9700 and 16 GB RAM.

A. The PCB Defects Dataset

We consider a public synthetic PCB defects dataset [4] with six fault types: Missing hole, mouse bite, open circuit, short, spur, and spurious copper. We summarize the defect distribution in Table I. The dataset can be downloaded from https://robotics.pkusz.edu.cn/resources/dataset/. Figure 3 shows some examples of these defects. We randomly divide the dataset into the training and validation sets with the ratio 8:2. The experimental results are collected on the validation set.

TABLE I: The class distribution in the dataset

Defect Name	#Images	#Instances
Missing hole	115	420
Mouse bite	115	430
Open circuit	116	414
Short	116	410
Spur	115	426
Spurious copper	116	434
Sum	693	2,534

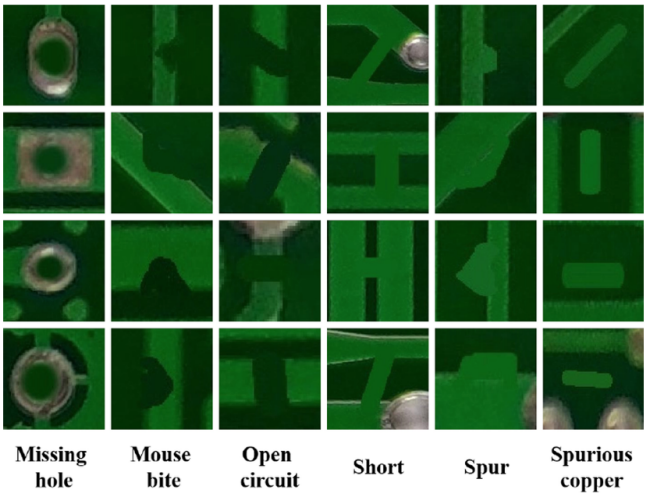


Fig. 3: Examples of PCB defects in the dataset [1]

B. Detection Performance Comparison between YOLOv8 and YOLOv5

The training parameters are set as follows. The image size and batch size are 640x640 and 16, respectively. The number of epochs is set to 80. Other parameters use default values as shown in the official website of Ultralytics. For YOLOv5s, the number of network parameters is 7.2M and the training

time is 4.539 hours. Meanwhile, YOLOv8s contains 11.13M parameters and the training time is 5.677 hours.

Figures 4 and 5 present learning curves of YOLOv5s and YOLOv8s models. The six left figures on both rows show bounding box regression, objectiveness, and classification losses while the four right figures on both rows present precision and recall, mAP 0.5 and mAP 0.5-0.95. During training epochs, in both YOLOv5 and YOLOv8, bounding box regression, objectiveness, and classification losses decrease gradually while performance metrics increase generally. YOLOv5 converges at around epoch 60 while the performance of YOLOv8 is saturated at roughly epoch 40, which means that YOLOv8 converges more quickly than YOLOv5.

After each training epoch, the currently best performance is recorded. Tables II and III show the final results of YOLOv5 and YOLOv8, respectively, after 80 training epochs. Generally, YOLOv8 produces higher performance than YOLOv5, especially in terms of recall, mAP50, and mAP50-95. More specifically, the recall of YOLOv8 is nearly 9% higher than that of YOLOv5. Recall values of YOLOv8 for all defect classes are superior to that of YOLOv5. For example, for the spurious copper defect, recall of YOLOv8 is around 67.9% compared to 54.3% of YOLOv5. Since YOLOv8 produces better results than YOLOv5, we use YOLOv8 as the default model for other experiments.

The confusion matrix at the IOU threshold of 0.25 is presented in Figure 6. The highest detection accuracy of 96% belongs to missing holes. Meanwhile, spurious copper defects have the lowest detection accuracy of 69%. Totally, the average accuracy on the validation set is $\frac{379}{448} = 84.5\%$.

TABLE II: The best results of the YOLOv5 model

Defect	Precision	Recall	mAP50	mAP50-95
Missing hole	1	0.891	0.956	0.382
Mouse bite	0.969	0.78	0.921	0.429
Open circuit	0.939	0.654	0.778	0.376
Short	0.844	0.797	0.785	0.258
Spur	0.972	0.679	0.746	0.310
Spurious copper	0.936	0.543	0.707	0.263
Total	**0.943**	**0.724**	**0.815**	**0.336**

TABLE III: The best results of the YOLOv8 model

Defect	Precision	Recall	mAP50	mAP50-95
Missing hole	0.970	0.911	0.939	0.387
Mouse bite	0.986	0.914	0.939	0.493
Open circuit	0.981	0.843	0.899	0.45
Short	0.813	0.809	0.795	0.326
Spur	1	0.715	0.855	0.352
Spurious copper	0.92	0.679	0.755	0.346
Total	**0.945**	**0.812**	**0.864**	**0.387**

Next, we verify the performance of YOLOv8 model on a real board as shown in Figure 7. YOLOv8 accurately detects three open-circuit defects with high probability. Meanwhile, the model only detects two out of three missing holes and some resistors are wrongly classified as missing holes. The main reason for this observation is that the training images do not contain any resistor. Hence, the YOLOv8 model could not learn resistor features. In future work, we collect more training data, especially real PCBs, and re-train the YOLOv8 model to improve detection performance.

C. FL-based PCB Fault Detection Performance

Assume that there are two participants who collaboratively train the global PCB fault detection model. The dataset is randomly divided into two parts and each part belongs to a particular client. We consider two settings A and B for the FL-based detection model. In setting A, each factory contains 240 training images and there are 63 shared images between factories. The overlapping percentage of the shared images is $\frac{63}{417} = 15.1\%$. Meanwhile, in setting B, each factory contains 300 training images and the percentage of shared training data is 43.8%. At each training epoch, the clients update the local models using their private dataset and share the local models with the cloud server. Then, the cloud server leverages FedAvg to combine local models and generate the consensus global model. The number of training epochs is set to 80.

Now, we compare the detection performance of FL-based architecture with local learning that uses only the dataset of one factory to train a local model. In this experiment, the FL setting A is used. As can be seen in Figure 8, the FL-based accuracy is $\frac{333}{448} = 74.3\%$ compared to the local model performance of $\frac{303}{448} = 67.6\%$. Thanks to the shared knowledge between local devices, the FL-based detection method can achieve higher performance than the local model. Even though the FL-based detection model achieves higher detection results than the local learning method but quite lower than the centralized learning one. The reason can be the lack of training samples in each client or the distribution difference between local datasets [19].

To enhance the performance of FL-based detection model, we increase the number of training samples in each client from 240 (setting A) to 300 (setting B). Table IV shows the performance of detection models with the highest performance on the validation set in terms of mAP50. Table IV contains six rows corresponding to six different models: local learning-based model, FL-based model with setting A on device 1, FL-based model with setting A on device 2, FL-based model with setting B on device 1, FL-based model with setting B on device 2, centralized learning-based model. The experimental results show that the centralized learning-based model achieves the highest detection performance followed by the FL-based model and local learning-based model. Particularly, using more training samples in each client results in higher detection performance. For example, on device 1, the mAP50 of the FL-based model with setting A is 0.71 compared to 0.751 on setting B.

To further evaluate the impacts of the number of training samples on detection performance, Figure 9 shows the learning curves of FL models with two settings A and B. The experimental results indicate that the detection model converges faster using setting B than setting A. In addition, precision, recall, mAP values can be improved when more training samples are shared between clients.

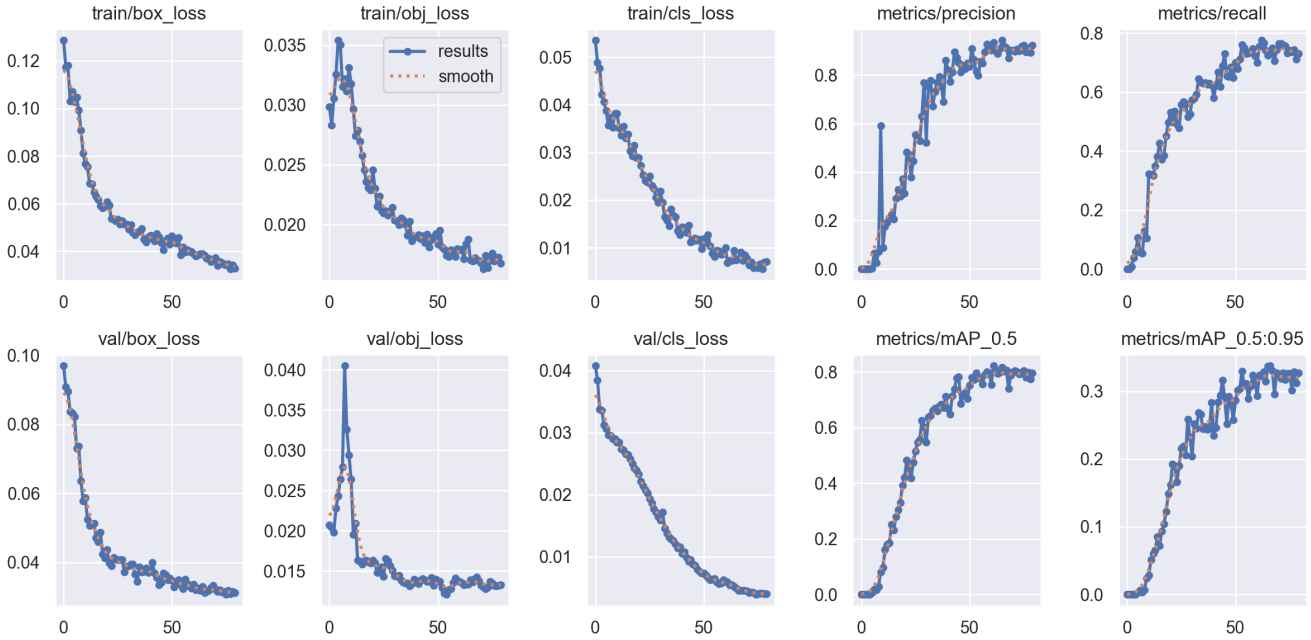


Fig. 4: Learning curves on the training and validation sets with the centralized YOLOv5s model.

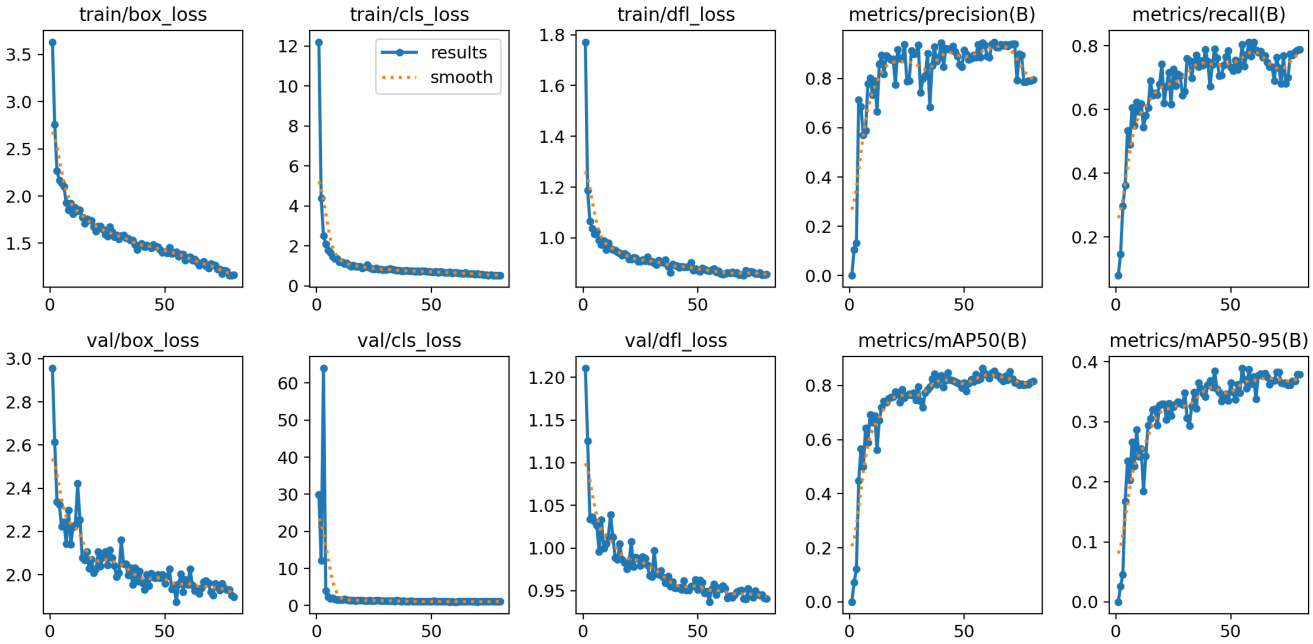


Fig. 5: Learning curves on the training and validation sets with the centralized YOLOv8s model.

TABLE IV: Performance Comparison between Detection Models

Metric	Precision	Recall	mAP50	mAP50-95
Local learning	0.915	0.560	0.715	0.322
FL, setting A, device 1	0.896	0.618	0.710	0.284
FL, setting A, device 2	0.906	0.592	0.733	0.313
FL, setting B, device 1	0.850	0.689	0.751	0.345
FL, setting B, device 2	0.916	0.664	0.763	0.345
Centralized learning	0.943	0.724	0.815	0.336

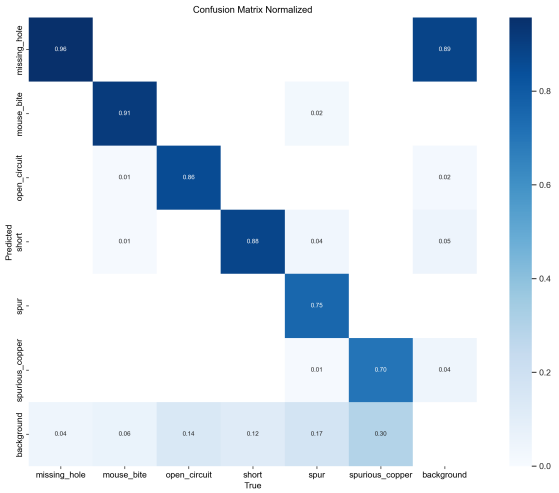


Fig. 6: Confusion matrix on the validation set with the centralized YOLOv8 model

Fig. 7: Performance evaluation on the collected dataset

V. Conclusion

In this work, we leverage the federated learning technique for PCB defect detection by constructing a global model using shared local updates from multiple factories. FL can preserve data privacy, which is an essential problem in machine learning. The FL-based PCB fault classification model is evaluated and compared with two other learning techniques: centralized and local learning. The experimental results show that the FL-based model produces higher performance than local learning but lower than centralized learning. In future work, we plan to improve the FL-based PCB defect detection architecture such that the performance of the FL-based model approaches that of the centralized detection model.

References

[1] A. Bhattacharya and S. G. Cloutier, "End-to-end deep learning framework for printed circuit board manufacturing defect classification," *Scientific reports*, vol. 12, no. 1, p. 12559, 2022.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[3] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[4] W. Huang and P. Wei, "A pcb dataset for defects detection and classification," *arXiv preprint arXiv:1901.08204*, 2019.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, "Federated learning for machinery fault diagnosis with dynamic validation and self-supervision," *Knowledge-Based Systems*, vol. 213, p. 106679, 2021.

[9] G. Jiang, W. Fan, W. Li, L. Wang, Q. He, P. Xie, and X. Li, "Deepfedwt: A federated deep learning framework for fault detection of wind turbines," *Measurement*, vol. 199, p. 111529, 2022.

[10] Z. Luan, Y. Lai, Z. Xu, Y. Gao, and Q. Wang, "Federated learning-based insulator fault detection for data privacy preserving," *Sensors*, vol. 23, no. 12, p. 5624, 2023.

[11] Q. Ling and N. A. M. Isa, "Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey," *IEEE Access*, 2023.

[12] X. Chen, Y. Wu, X. He, and W. Ming, "A comprehensive review of deep learning-based pcb defect detection," *IEEE Access*, 2023.

[13] J. Zheng, X. Sun, H. Zhou, C. Tian, and H. Qiang, "Printed circuit boards defect detection method based on improved fully convolutional networks," *IEEE Access*, vol. 10, pp. 109 908–109 918, 2022.

[14] J. Lim, J. Lim, V. M. Baskaran, and X. Wang, "A deep context learning based pcb defect detection model with anomalous trend alarming system," *Results in Engineering*, vol. 17, p. 100968, 2023.

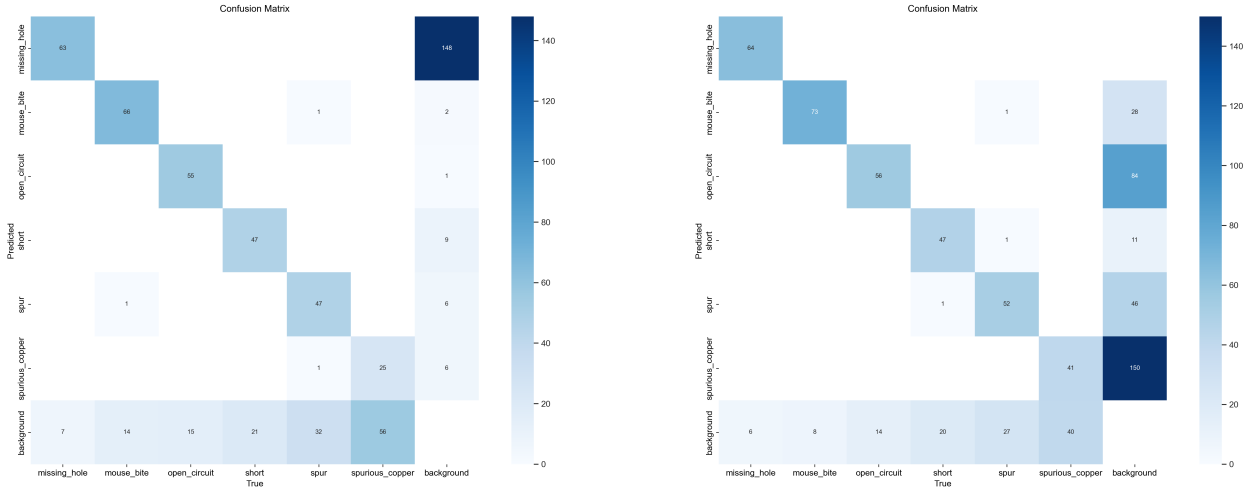
[15] Z. Yu, Y. Wu, B. Wei, Z. Ding, and F. Luo, "A lightweight and efficient model for surface tiny defect detection," *Applied Intelligence*, vol. 53, no. 6, pp. 6344–6353, 2023.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[17] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

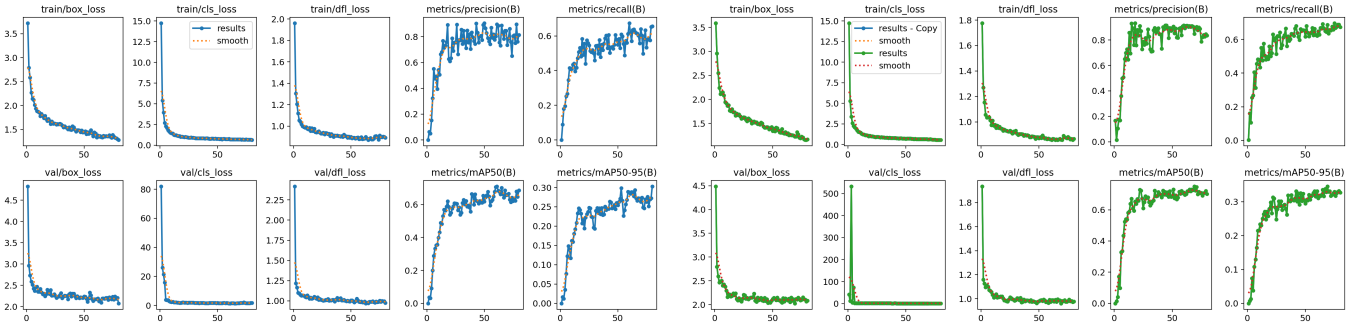
[18] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*. PMLR, 2020, pp. 5132–5143.

[19] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.



(a) (b)

Fig. 8: Performance comparison between two learning techniques (a) local learning and (b) federated learning.



(a) (b)

Fig. 9: Performance comparison between two FL-based model settings: (a) the ratio of shared data is 0.151 and (b) the ratio of shared data is 0.438.