Transfer Learning for Real-Time Surface Defect Detection With Multi-Access Edge-Cloud Computing Networks

Hui Li, *Student Member, IEEE*, Xiuhua Li, *Member, IEEE*, Qilin Fan, *Member, IEEE*, Qingyu Xiong, Xiaofei Wang, *Senior Member, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

***Abstract*—The development of deep learning and edge computing provides rapid detection capability for surface defects. However, components produced in actual industrial manufacturing environments often have tiny surface defects and training data for each specific defect type is limited. Meanwhile, network resources at the edge of industrial networks are difficult to guarantee. It is challenging to train a proper surface defect detection model for each specific surface defect type and provide a real-time surface defect detection service. To address the challenge, in this paper, we propose a real-time surface defect detection framework based on transfer learning with multi-access edge-cloud computing (MEC) networks. Furthermore, we improve the original YOLO-v5s framework by introducing the spatial and channel attention mechanism, and adding an additional detection head to enhance the detection ability on tiny surface defects. Evaluation results demonstrate that the proposed framework has superior performance in terms of improving detection accuracy and reducing detection delay in the considered MEC network.**

***Index Terms*—Surface defect detection, multi-access edge-cloud computing networks, transfer learning, YOLO-v5s.**

Manuscript received 11 May 2023; revised 18 July 2023; accepted 30 July 2023. Date of publication 3 August 2023; date of current version 7 February 2024. This work is supported in part by National Key R & D Program of China (Grants No. 2022YFE0125400), National NSFC (Grants No. 62102053 and 62072060), Chongqing Research Program of Basic Research and Frontier Technology (Grant No. cstc2022ycjhbgzxm0058), Key Research Program of Chongqing Science & Technology Commission (Grant No. cstc2021jscx-dxwtBX0019), Haihe Lab of ITAI (Grant No. 22HHXCJC00002), the Natural Science Foundation of Chongqing (Grant No. CSTB2022NSCQ-MSX1104), the General Program of Chongqing Science & Technology Commission (Grant No. CSTB2022TIAD-GPX0017), Regional Innovation Cooperation Project of Sichuan Province (Grant No. 2023YFQ0028), Regional Science and Technology Innovation Cooperation Project of Chengdu City (Grant No. 2023-YF11-00023-HZ), Guangdong Pearl River Talent Recruitment Program (Grants No. 2019ZT08X603 and 2019JC01X235), and Science and Technology Plan Project of Chongqing Economic and Information Commission (Grant No. 2211R49R03). The associate editor coordinating the review of this article and approving it for publication was X. Fu. *(Corresponding author: Xiuhua Li.)*

Hui Li, Xiuhua Li, Qilin Fan, and Qingyu Xiong are with the School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China (e-mail: h.li@cqu.edu.cn; lixiuhua1988@gmail.com; fanqilin@cqu.edu.cn; xiong03@cqu.edu.cn).

Xiaofei Wang is with the Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin 300072, China (e-mail: xiaofeiwang@tju.edu.cn).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T1Z4, Canada (e-mail: vleung@ieee.org).

Digital Object Identifier 10.1109/TNSM.2023.3301718

I. Introduction

RECENTLY, with the rapid development of smart manufacturing and industrial automation, component production efficiency has been significantly improved [1], [2]. However, due to differences in technical level and working condition, the quality of manufactured components are easily affected, and surface defects (e.g., surface scratches, oil spot, holes and wrinkles) occur frequently [3], [4], [5]. Surface defects not only affect the aesthetics of component, but also have a significant impact on product performance. At present, manual detection methods are still widely used by various industrial component manufacturers [6]. Training workers to identify these complex and tiny surface defects has many disadvantages such as high workload and low detection accuracy, which cannot meet the requirements for defect detection consistency and high efficiency in industrial networks.

Deep learning (DL) technology has achieved excellent results in defect detection scenarios [7], [8], [9]. DL has huge computational complexity and requires high-performance graphics processing units to support model training and inference [10], [11]. Although cloud-based computing architecture has enough computing and storage resources to complete the training of DL model, the real-time requirement of defect detection is usually difficult to guarantee. Multi-access edge-cloud computing (MEC) networks have emerged as a novelty promising technique that enables large amounts of raw data processed at network edges (e.g., base stations (BSs)) to improve the quality of service [12], [13], [14]. To deal with high detection delay in cloud servers, MEC networks can deploy DL model at edge servers of industrial networks to provide real-time defect detection services for Internet of Things (IoT) devices nearby. It can be expected that this new network architecture will be popular in the next-generation industrial networks.

Although MEC has advantages in handling delay-sensitive services, there still exist several technical bottlenecks in performing defect detection. DL model requires a large number of marked datasets for model training [15], [16]. The distribution of surface defects in actual industrial manufacturing environments is quite different and produced components often have tiny surface defects. It is difficult to obtain sufficient defect data for each specific defect type. Detection performance for tiny surface defects is poor when sufficient data can not be

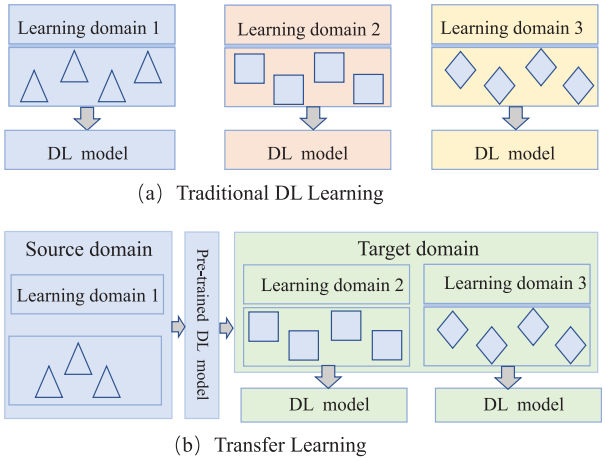


Fig. 1. Paradigm of transfer learning and traditional deep learning.

obtained [17], [18]. Meanwhile, centralized training requires collecting different types of defect data, and finally training detection models separately. It will increase the difficulty of defect detection, leading to lower system scalability. Transfer learning (TL) [19], [20], [21] as a novel DL approach that focuses on learning common knowledge from one or more related fields but different application scenarios to help DL models achieve better performance. As shown in Fig. 1, compared with traditional DL models, TL first trains a pre-trained model from a larger dataset in the source domain, and then fine-tunes the pre-trained model via limited training data in the target domain. Therefore, TL can effectively address the challenges of limited training data for tiny surface defects in MEC networks described earlier [22], [23].

Several existing studies have investigated defect detection problems based on TL method in MEC networks. The authors in [19] proposed a TL-enabled edge convolutional neural network framework in industrial edge networks. To address the limited training data for defects, the proposed framework general pre-trained defect detection model on the edge layer. The locally limited dataset is used to further fine-tune the pre-trained model parameters from IoT devices. Gai et al. [24] proposed an end-to-end flexible hotspot detection framework based on a fully convolutional network with TL to improve detection performance. The authors in [25] proposed a deep TL-based detection framework for tag signal detection, which deploys the TL and offline learning for online detection. Therefore, it is attractive to investigate how the MEC and TL technology can be efficiently combined to enhance the defect detection performance of industrial MEC networks.

To accelerate the defect detection process in MEC networks, there is a fundamental challenge to overcome.

How to efficiently combine the MEC and TL technology with limited training data to improve the surface defect detection performance? TL can train a proper surface defect detection model with minimal training samples by learning common knowledge between source domain and target domain in MEC networks. The cloud servers are usually composed of high-performance server clusters with powerful computing and storage capabilities, therefore, we store source domain with a large amount of training data at the cloud servers first and pre-train a detection model through the source domain. Then the cloud servers send the pre-trained model parameters to the edge servers near the IoT devices and initialize the detection model at the edge servers. The edge servers use the limited training data in the target domain to fine-tune the pre-trained model parameters and deploy the defect detection model at edge servers for surface defect detection.

In this paper, we are motivated to explore the issue of surface defect detection in MEC networks. Particularly, we integrate the MEC and TL by pre-training a general defect detection model at cloud servers, and then fine-tuning the pre-trained model at edge servers according to limited local specific surface defect data for different industrial manufacturing environments. The main contributions are stated as follows:

- We present a three-layer MEC architecture that supports neural network knowledge transfer in different feature spaces through TL, for practically improving detection accuracy and detection delay of surface defects.
- We improve the original YOLO-v5s framework by introducing the spatial and channel attention mechanism, and adding an additional detection head to strengthen the defect detection performance on tiny surface defects.
- We propose a real-time surface defects detection framework based on the TL with MEC networks. Edge servers use a relatively small amount of defect data to fine-tune the pre-trained model from cloud servers, which can effectively improve the performance of limited training data for tiny defects and reduce model training time.
- We evaluate the performance of the proposed framework through extensive experiments on real-world datasets. Experimental results demonstrate that the proposed framework has superior performance in terms of improving detection accuracy and reducing detection delay.

The remainder of the paper is organized as follows. We discuss the related work in Section II. Section III introduces the system design and problem formulation. We present the design of defect detection framework in Section IV. Performance evaluations are shown in Section V, followed by concluding remarks in Section VI.

II. Related Work

A. Edge-Cloud Computing for Defect Detection

Recently, many pioneers researchers have studied defect detection problems in edge or cloud computing networks [26], [27], [28], [29], [30], [31]. Zhu et al. [26] proposed a deep learning-based defect detection framework, by modifying the network structure of DenseNet to better adapt to resource-constrained edge networks. Liu et al. [27] proposed a fast detection method for intelligent key power line components at the network edges and improved spatial pyramid pooling techniques for the YOLO-v5 framework, which greatly improves detection accuracy. Xia et al. [28] proposed a YOLO-based detection framework for power equipment defect detection in the industrial environment, which uses a lightweight multi-layer neural network to learn the features

for image classification on cloud servers. However, these studies only considered training the defect detection model in the edge or cloud servers separately, and it is difficult to take full advantage of the edge-cloud collaborative computing mode.

To take advantage of the potential of edge-cloud collaborative computing, many researchers try to use edge-cloud collaborative computing technology to complete defect detection tasks. Zhao et al. [29] proposed an edge-cloud collaborative defect detection architecture. At the edge layer, defect inspection equipment can realize the collection of inspection image data, the cloud layer enables efficient image data storage and analysis, and adjusts the edge detection method by the TL method. Liang et al. [30] proposed an edge-cloud collaboration target detection framework, named EdgeYOLO, which can effectively avoid excessive cloud computing power dependence and uneven distribution of computational resources. Tang et al. [31] proposed a two-stage algorithm to identify defect images with high sensitivity. By installing multiple sensors and using cloud servers to establish edge-cloud computing infrastructure based on IoT devices, it could automatically detect defects with low delay and higher precision. These studies have been evidenced to improve defect detection performance.

B. Deep Learning for Defect Detection

Extensive studies based on DL have been conducted and achieved excellent performance for defect detection. Girshick et al. [32] proposed the R-CNN algorithm, which applied a convolutional neural network to the object detection field for the first time and achieved an average precision of 53.3%. Furthermore, Girshick [33] proposed the Fast-RCNN algorithm in 2015 and used the multi-task loss function to return the category judgment and bounding box regression. On the basis of Fast-RCNN, Ren et al. [34] proposed the Faster-RCNN algorithm by using a neural network instead of a selective algorithm to generate the candidate regions with a slower time. However, these studies usually focused on optimizing defect detection accuracy as the first goal and ignored detection delay in defect detection process.

To deal with scenarios with high real-time requirements, many researchers try to improve detection speed while ensuring average detection accuracy. Redmon et al. [35] proposed a real-time target detection model YOLO, which directly completed the generation of position coordinates and object category probabilities task. The authors in [36] proposed a new center point-based single-stage object detection algorithm, named CenterNet, by displaying the target through the center point and returning some attributes of the target. Although these DL models could solve real-time requirements, however, the distribution of surface defects is obviously different. Centralized training paradigm can only be applied to a single specific type of surface defect. To address the challenge, TL [19], [20], [37], [38] focuses on knowledge transfer across domains and can guide neural networks and pre-trained models in different feature spaces, which can be used for addressing the surface defect detection problem in this paper.

TABLE I
SUMMARY OF IMPORTANT NOTATIONS

Notation	Definition
$\mathcal{B}(B)$	The set (number) of edge server
D^S, D^T	The source domain and target domain
X^S	The input feature space of source domain
X^T	The input feature space of target domain
Y^S, Y^T	The label space of source domain and target domain
θ^S, θ^T	The parameters of source domain and target domain
$(x_{s_{n_s}}, y_{s_{n_s}})$	The training sample of source domain
$(x_{t_{n_t}}, y_{t_{n_t}})$	The training sample of target domain
n_S, n_T	The sample size of source domain and target domain
F	The feature map
H, W, C	The length, width and number of channels
$\mathcal{R}^S$, $\mathcal{R}^T$	The risk function
$L(\cdot)$, $J(\cdot)$	The model loss and regularization term

III. System Design and Problem Formulation

In this section, we first introduce the topology of MEC architecture, then discuss the surface defect detection process modeling. Finally, we formulate the corresponding problem for defect detection in the considered MEC network. Some key parameters are listed in Table I.

A. MEC Architecture for Defect Detection

As illustrated in Fig. 2, we consider a scenario of MEC architecture for industrial manufacturing with a cloud server and B edge servers (denoted by $\mathcal{B} = \{1, 2, \ldots, B\}$). The network topology for surface defect detection mainly includes three layers, i.e., device layer, edge layer and cloud layer.

1) Device Layer: the device layer is mainly composed of different types of IoT devices, such as cameras, machine tools, sensors or lighting devices. These IoT devices are randomly distributed in different factories or production lines in different geographical locations through wireless or wired connections to communicate with the edge servers. Components produced at different production lines often have diverse tiny defect types (e.g., hole defects or corrugated folds) and the training data for each specific surface defect type is limited. Therefore, the detection model needs to be properly designed for each specific surface defect type. Besides, the device layer completes the collection of surface defect data and transmits them to the edge layer in real time for detection.

2) Edge Layer: the edge layer consists of a certain number of edge servers, which are located at the edge of industrial network close to the data source (i.e., device layer). Each edge server has certain computing and storage capabilities, and can complete transferred pre-trained model fine-tuning and defect detection model deployment. Input data at different edge servers are specific surface defect data, which are captured at defect inspection points in different factories or production lines from the device layer. These defect data have different background textures and defect types, therefore, each edge server needs to deploy a specific defect detection model to complete the detection of surface defects.

3) Cloud Layer: the cloud layer has powerful data processing and storage capabilities that can complete the pre-training

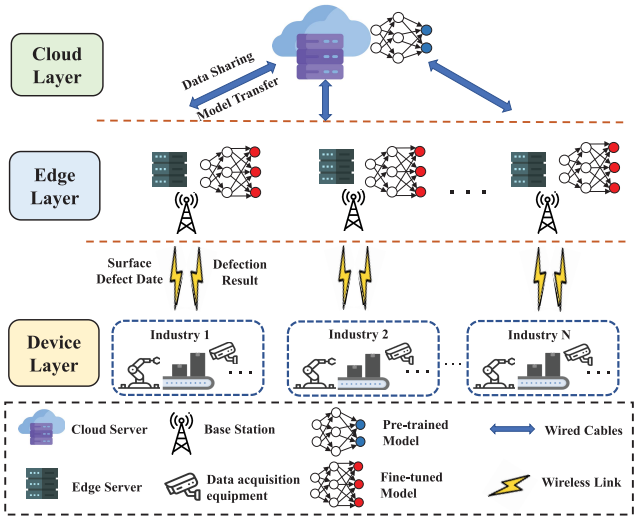


Fig. 2. Topology of the considered MEC architecture for defect detection.

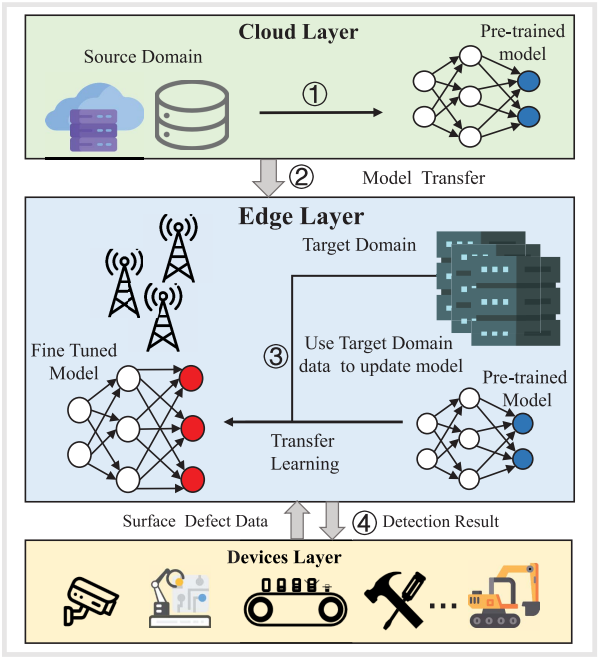


Fig. 3. Detailed defect detection process in the considered MEC network.

of general surface defect detection models. The cloud layer and edge layer are connected by wired links and can exchange information with each other. Due to the lack of prior knowledge of specific surface defect types, the pre-trained detection models usually have a certain generalization detection ability and can complete the detection of various types of surface defects, but the detection accuracy for each specific surface defect type is difficult to guarantee.

B. Surface Defect Detection Process Modeling

In the considered MEC network, we should further transfer the pre-trained surface defect detection model from the cloud servers to each edge server. The detailed defect detection process in the considered MEC network is illustrated in Fig. 3. Specifically, we train the surface defect detection model through TL in an edge-cloud collaboration manner, and then deploy the defect detection service at each edge server. The defect detection service moving from the centralized cloud layer to the edge layer near the data sources, can significantly reduce detection delay and alleviate the pressure on the core network bandwidth. The whole detection process of the surface defect in the considered MEC network can be summarized as follow: ①: The cloud servers use training data in source domain to train a DL neural network as the general pre-trained detection model for all general surface defect types. ②: Then the cloud servers transfer the pre-trained detection model to the edge servers. Note that the distribution of surface defects in cloud servers and edge servers is quite different and the training data for each specific defect type is limited. ③: The edge servers receive the pre-trained model from the cloud servers, then it uses the local specific type of limited defect data (i.e., target domain) to fine-tune the pre-trained model to obtain a new model with higher accuracy or inference speed, and apply it to local specific surface defect type. ④: After the above steps, each edge server is trained to obtain a proper detection model and deploy it on the local edge server. When the device layer has newly generated surface defect data (i.e., defect images) that needs to be detected, it only needs to send the surface defect data to the corresponding edge server, and use the fine-tuned model for surface defect detection at the edge server instead of the cloud servers.

C. Problem Formulation

In the considered MEC network, the limited training data in target domain distributed on edge servers is not enough to train a robust detection model for tiny surface defects. To address the challenge, in this paper, we propose a real-time surface defect detection framework based on TL in MEC networks. Denote $D^T = \{(x_{t_1}, y_{t_1}), (x_{t_2}, y_{t_2}), \ldots, (x_{t_{n_T}}, y_{t_{n_T}})\}$ as the dataset of target domain, where $(x_{t_{n_t}}, y_{t_{n_t}}) \in D^T$ is training sample and n_T is the total sample size of target domain D^T. Denote $D^S = \{(x_{s_1}, y_{s_1}), (x_{s_2}, y_{s_2}), \ldots, (x_{s_{n_S}}, y_{s_{n_S}})\}$ as the dataset of source domain, in which $(x_{s_{n_s}}, y_{s_{n_s}}) \in D^S$ is the training sample in source domain dataset D^S and n_S is the total sample size. Source domain dataset D^S usually has a larger training sample than target domain D^T (i.e., $n_S >> n_T$). Moreover, source domain dataset D^S usually has a certain task similarity with target domain dataset D^T, so that D^S and D^T can share certain model parameters through the TL. That is, it is feasible to transfer the model parameters trained with a large amount of data in the source domain D^S to the target domain D^T for prediction. Denote $\varpi_S = \{X^S, Y^S, f^S(X^S; \theta^S)\}$ and $\varpi_T = \{X^T, Y^T, f^T(X^T; \theta^T)\}$ as the set of mapping relationship on the source domain dataset D^S and target domain dataset D^T, respectively, in which $X^S = \{x_{s_1}, x_{s_2}, \ldots, x_{s_{n_S}}\}$ is the input feature space, $Y^S = \{y_{s_1}, y_{s_2}, \ldots, y_{s_{n_S}}\}$ is the label space, $f^S(X^S; \theta^S)$ is the predictive function and θ^S is the pre-trained model parameters on source domain D^S. Besides, the term $X^T = \{x_{t_1}, x_{t_2}, \ldots, x_{t_{n_T}}\}$, $Y^T = \{y_{t_1}, y_{t_2}, \ldots, y_{t_{n_T}}\}$, $f^T(X^T; \theta^T)$ and θ^T is the input feature space, label space,

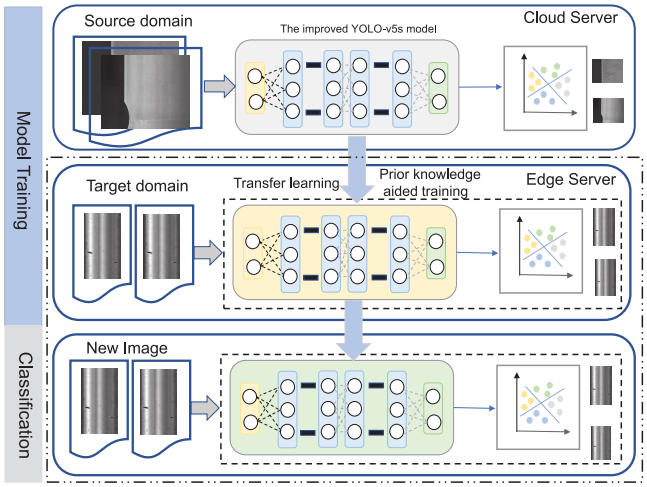


Fig. 4. Diagram of the proposed framework for surface defect detection in the considered MEC network.

predictive function and model parameters on target domain D^T, respectively.

The main goal of the proposed framework is to train a detection model with minimal training samples of target domain D^T by using common model parameters between source domain D^S and target domain D^T. The defect detection problem in the considered MEC network can be modeled as a learning task $f^S(X^S;\theta^S) \rightarrow f^T(X^T;\theta^T)$ to improve prediction accuracy on target domain dataset D^T by knowledge transfer between source domain D^S and target domain D^T. Thus, the defect detection problem can be formulated as

$$\mathcal{P}: \quad \min f^T\left(X^T;\theta^T\right) \tag{1a}$$

$$s.t. \quad \theta^T \subseteq \theta^S, \text{ and } \theta^T \text{ is initialized by } \theta^S. \tag{1b}$$

The constraint (1b) shows that θ^T should be a subset of θ^S. θ^T is initialized by θ^S through TL and can be updated by fine-tuning the model parameters using the target source D^T. To address the defect detection problem, we propose a real-time surface defects detection framework based on TL in MEC networks.

IV. Design of Defect Detection Framework

In this section, we first discuss the overview of the YOLO framework and explain the reason why YOLO-v5s framework is used instead of other versions. Then we show the detailed structure of the detection model of the improved YOLO-v5s framework. Next, we apply TL to address the challenge of limited training data for each specific defect type in MEC networks. Finally, we give the loss functions of the improved YOLO-v5s framework. The diagram of the proposed framework for surface defect detection can be shown in Fig. 4.

A. Overview of YOLO Framework

The YOLO framework [39], [40], [41] is a popular DL-based object detection system. Each version of YOLO framework has different advantages. YOLO-v1 is an object detection system capable of fast object detection and classification. However, its performance is relatively poor on the detection of tiny objects. YOLO-v2 introduces multi-scale detection technology and the expansion of convolutional layers to greatly improves defect detection performance. YOLO-v3 improves the accuracy of the detection model by integrating the residual connection and multi-scale prediction. YOLO-v4 further improves the defect detection speed and accuracy by adding the SPP module and CSPDarknet53 network. It has achieved excellent performance on object detection datasets. We do not choose the latest version of the YOLO framework (e.g., YOLO-v6 or YOLO-v7) for surface defect detection, because the new version of the YOLO framework may require more computing and storage resources caused by more complex model architecture or larger model size, which may not be suitable for the current detection tasks and application scenarios.

YOLO v5 [42] uses the cross-stage partial network [43] as the model backbone and path aggregation network [44] as the model neck for feature aggregation, which has achieved remarkable success in the field of image detection. Compared with other versions of YOLO, YOLO v5 has higher accuracy and easier deployment. To enhance the defect detection ability on tiny surface defects, in this subsection, we have made the following improvements on the original YOLO-v5s framework: 1) We introduce convolutional block attention module (CBAM) to enhance the spatial and channel attention on the area that contains tiny surface defects to obtain more key information; Specifically, spatial attention focuses the detection model on local features like the target's outline, while channel attention emphasizes channel-specific information such as color and texture. This integration enhances classification accuracy by capturing fine-grained details and discriminating between different classes. 2) On the basis of three prediction heads of original YOLO-v5s framework, we add an additional detection head in the head network and change the structure of neural network to strengthen detection performance for tiny surface defects. The detailed model architecture is shown in Fig. 5, and its architecture can be divided into four parts: input layer, backbone network, neck network and head network.

B. Detailed Structure of Improved YOLO-v5s Framework

The input layer can realize the function of data enhancement, adaptive anchor box calculation, and image scaling. The backbone network is mainly composed of focus structure [45] and CSP1_X_CBAM structure, and mainly completes the feature extraction of defect data. The focus structure can perform slice and convolution operations on input data and slice images according to the pixel interval, and integrates the width and height information into the channel. Its main purpose is to reduce the parameters and improve the speed of forward and backward propagation. To improve the performance of YOLO-v5s framework on tiny surface defects, we intend to change the size of the feature map obtained by the first downsampling in the backbone network from 76*76 to 152*152. This operation generates a larger feature map that can save more

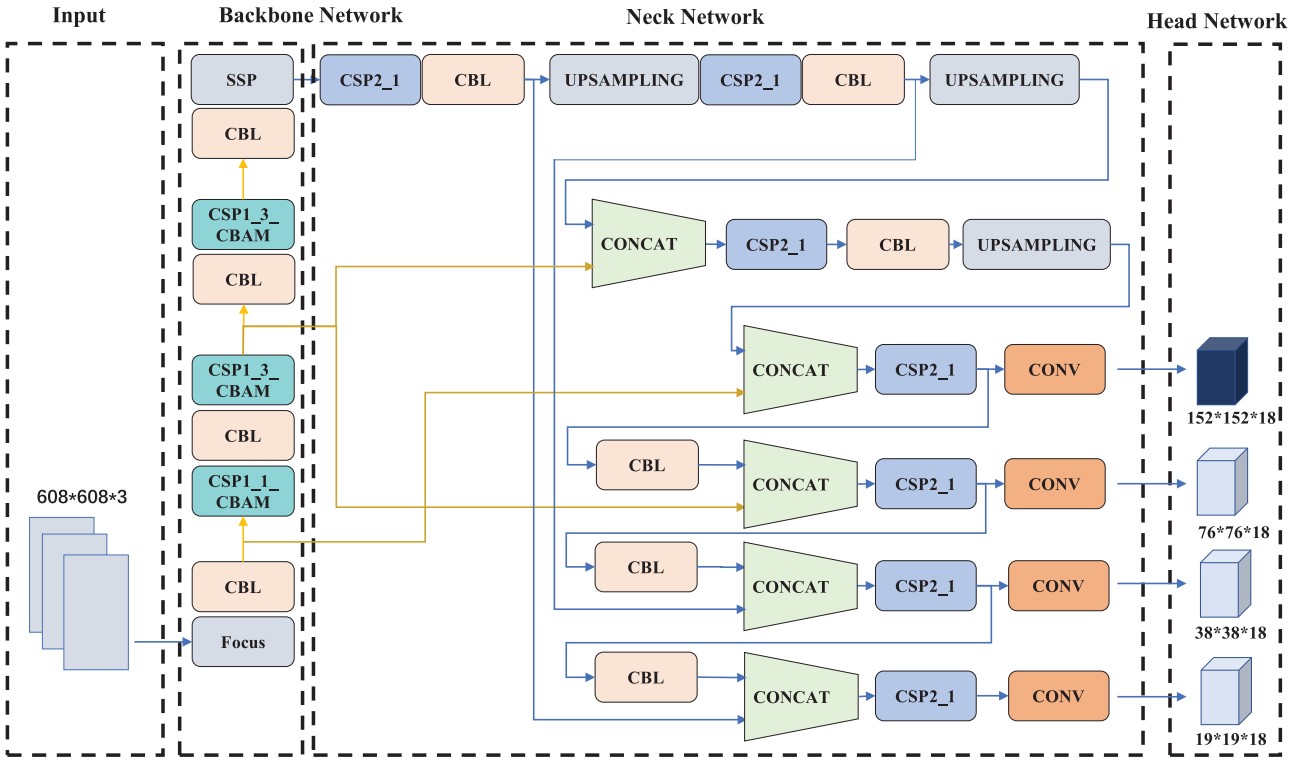


Fig. 5. Detailed structure of improved YOLO-v5s by introducing the spatial and channel attention mechanism, and adding an additional detection head.

image information and reduce the feature loss of tiny surface defects. Correspondingly, in the neck network, we also performed an upsampling operation to obtain a 152*152 feature map and spliced the feature map obtained at the channel level to integrate strong semantic features and localization features. So, we correspondingly have 4 layers of feature fusion maps in the head network, the sizes of which are 152*152, 76*76, 38*38 and 19*19. Input image area corresponding to each grid in the larger-size feature map is smaller, which is more suitable for detecting tiny surface defects.

CBAM module is an efficient and lightweight attention module that combines spatial and channel attention, which can be integrated into the YOLO-v5s framework to complete the detection of surface defects. Specifically, spatial attention can make the detection model pay more attention to local features such as the outline of the target, while channel attention can better learn the channel feature information such as the color and texture of the target, thereby improving the classification accuracy. To help the YOLO-v5s framework to pay more attention to the feature information of tiny surface defects, we connect CBAM module in parallel with the original CSP1_X module. Clearly, the CSP1_X_CBAM module divides the feature map into three branches and performs forward propagation separately, and then fuses the original feature maps of three branches to obtain a new feature map. CBAM module can obtain more key information by enhancing the attention on important regions containing defects. Denote *H*, *W* and *C* as the length, width and number of channels, respectively. For the input feature map *F*, we can calculate the weight of each channel according to the formulas

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) = \sigma\left(W_1\left(W_2\left(F_{avg}^C\right)\right) + W_1\left(W_2\left(F_{max}^C\right)\right)\right). \quad (2)$$

We perform the max pooling (*MaxPool*) and average pooling (*AvgPool*) on each channel of input feature map, and then go through multi-layer perceptron. After that, we multiply the corresponding elements of feature vector output one by one, and finally execute sigmoid activation function. Taking the feature map output by the channel attention module as input, the *MaxPool* and *AvgPool* are performed in sequence, and then the convolution operation is performed on the obtained intermediate vector. After passing the obtained result to the sigmoid activation function, we can obtain the spatial attention. Therefore, the calculation process can be expressed as

$$M_s(F) = \sigma\left(f^{7\times7}(AvgPool(F)); (MaxPool(F))\right). \quad (3)$$

C. Transfer Learning Through Fine-Tuning in MEC Networks

We select the steel surface defect dataset as the source domain D^S and the railway tracks dataset and NEU-DET dataset as target domain D^T, respectively. The specific transfer process of using the TL to train the improved YOLO-v5s framework in the considered MEC network can be shown in Fig. 6. In the process of using source domain D^S to improve the surface defect detection performance on target domain D^T through knowledge transfer, we keep the previous modules unchanged and use the parameters of the pre-trained model to

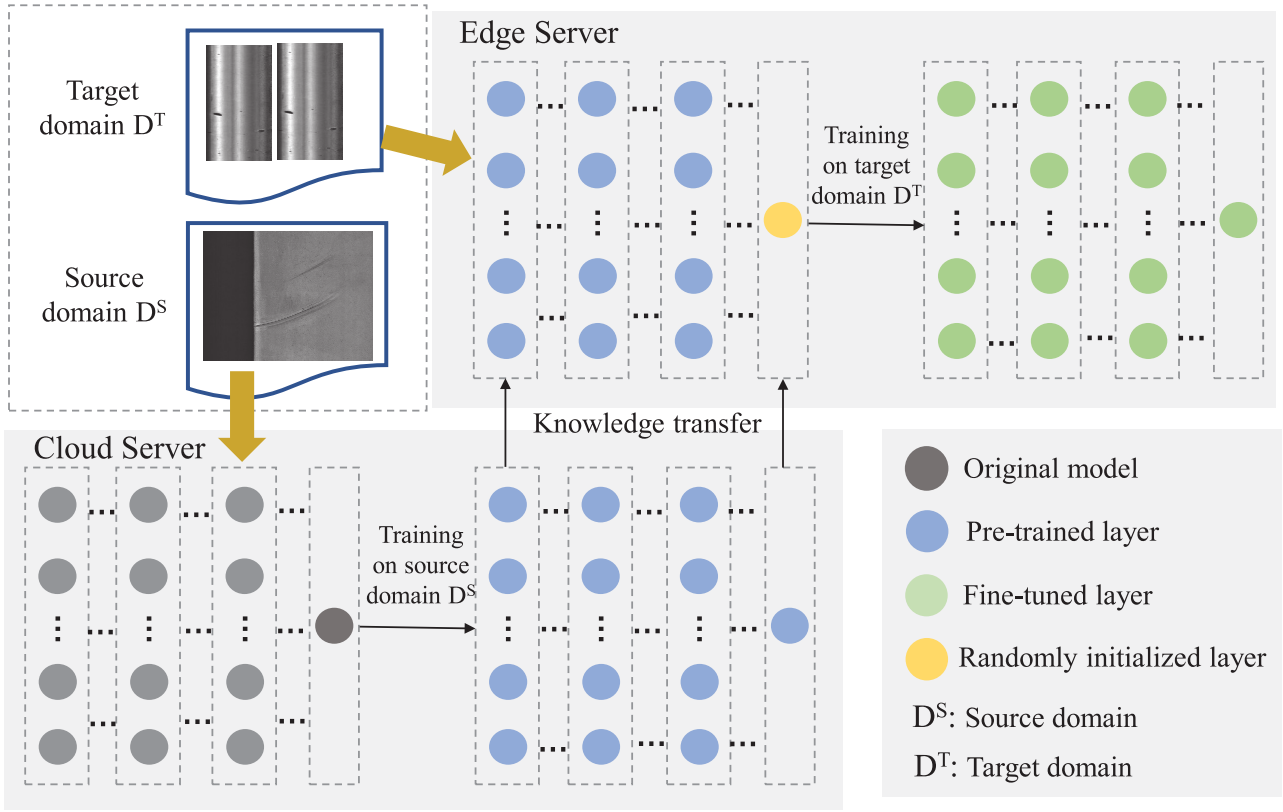


Fig. 6. Transfer learning process between cloud servers and edge servers in the considered MEC network.

initialize the model parameters at the edge servers. The cloud servers are usually composed of high-performance server clusters with powerful computing and storage capabilities, so we store source dataset D^S with a large amount of training data at first, and then pre-train detection model $F^S(\cdot;\theta^S)$ through source domain D^S. Thus, the definition of this process can be shown in the formula as

$$\mathcal{F}^S\left(\cdot;\theta^S\right) = \operatorname{argmin}\mathcal{R}^S\left(f^S\left(X^S;\theta^S\right), Y^S\right), \tag{4}$$

where $\mathcal{R}^S = \frac{1}{n_S}\sum_{i=1}^{n_S} L(f^S(x_{s_{n_i}};\theta^S), y_{s_{n_i}}) + \lambda J(f^S(\cdot))$ is the risk function, function $L(\cdot)$ is the model loss and $J(\cdot)$ is the regularization term. θ^S is the parameter of the pre-trained model, which can be defined as

$$\theta^S = \left\{\theta_i^S\right\}_{i=1}^{|\theta^S|}, \tag{5}$$

where θ_i^S is the number of parameters at each layer of the pre-trained model and $|\theta^S|$ is the number of layers. The pre-trained model from the cloud server greatly reduces the training time of the model, and enables the underlying network of the improved YOLO-v5s framework to learn some primary features through source domain D^S, such as color features, texture features, and edge features, which are used for subsequent of knowledge transfer provides guidance.

After that, the cloud server sends the pre-trained model parameters θ^S to each edge server near the IoT devices, and initializes the defect detection model at each edge server. Then edge servers use the target domain D^T to fine-tune the model parameters and update the randomly initialized layer. By training the model with target domain D^T to fine-tune the parameters, the knowledge can be transferred and the model can complete defect detection tasks for each specific defect type. The process is defined as

$$\mathcal{F}^T\left(\cdot;\theta^T\right) = \operatorname{argmin}\mathcal{R}^T\left(f^T\left(X^T;\theta^T\right), Y^T\right) \tag{6}$$

where θ^T is initialized by θ^S through TL. θ^T can be updated by fine-tuning the model parameters using the target domain D^T. $\mathcal{R}^T$ is the risk function of target domain the same as $\mathcal{R}^S$. Since the parameters θ^T of the proposed framework at the edge servers are initialized by θ^S, the training of the detection model can be guided by knowledge transfer, and the convergence speed will be improved. The detailed process of the proposed framework can be shown in Algorithm 1.

D. Network Loss Function

Surface defect detection mainly needs to consider several requirements, such as the defect size, category and location. The YOLO-v5s framework mainly uses three loss functions (named classification loss, objectness loss, and GIoU loss) to measure defect detection performance. For defect classification, the output predicted labels are usually mutually exclusive, thus, we use the binary cross entropy loss function to represent

Algorithm 1: Improved YOLO-v5s Framework Based on Transfer Learning for Surface Defect Detection

Input: Source domain dataset D^S and target domain dataset D^T, the configuration information of all IoT devices and edge servers.
Select datasets of steel surface defect and railway tracks as source domain D^S and target domain D^T.
—Step 1. [YOLO-v5s framework improvement]
Replace CSP1_ X with CSP1_X_CBAM module in the YOLO-v5s framework by introducing the spatial and channel attention mechanism (the calculation process can be shown in formula (3), (4) and (5)).
Changing feature map *F* obtained by the backbone and neck network and add additional detection head 152*152*18 for tiny defects in head network.
—Step 2. [Transfer learning]
for edge server index $b = 1, 2, 3, \ldots, B$ **do**
    Migrate the pre-trained model parameters θ^S from cloud server to edge server *b* for fine-tuning.
    Minimize loss functions (9), (10) and (1) to train the improved YOLO-v5s framework.
    Complete the training of the improved YOLO-v5s framework and deploy the defect detection model on the edge server *b*.
end
Output: mAP, precision, recall and detection time.

the class probability and its formula as

$$Loss_{cls} = \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \sum_{c\in \text{classes}} [p_i(c)\log(p_i(c)) + (1-p_i(c))\log(1-p_i(c))] \tag{7}$$

where S^2 is all grid cells, *B* is all prediction boxes, $\mathbf{1}_{ij}^{obj}$ indicates that whether it is a positive sample and $p_i(c)$ represents the real labelled classification probability. The objectness loss function is mainly used to calculate the confidence of the network and it can be calculated as

$$Loss_{obj} = \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbf{1}_{ij}^{obj}[c_i\log(c_i) + (1-c_i)\log(1-c_i)] - \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj}[c_i\log(c_i) + (1-c_i)\log(1-c_i)] \tag{8}$$

where $\mathbf{1}_{ij}^{noobj}$ indicates that whether it is a negative sample. c_i is the label value of defect detection.

To frame the detected region, we need to predict the location information of bounding box. Intersection over Union (IoU) is the ratio of intersection and union of target box and truth box, which is often used to measure the accuracy of predicted box in target detection. For predicted detection box *A* and truth box *B*, IoU loss of two boxes is expressed in the formula as $\text{IoU} = \frac{|A\cap B|}{|A\cup B|}$. Please note that when the truth box contains the predicted detection box, no matter where the predicted detection *A* box is in the truth box *B*, the location loss remains the same. Besides, the value of IoU is 0 means the predicted detection box *A* and real box *B* do not intersect, which cannot represent the distance between the predicted detection box and the truth box, and further optimization cannot be completed. Therefore, on the basis of IoU, the YOLO-v5s framework uses generalized IoU (GIoU) loss to represent the location loss. GIoU introduces the smallest box that can frame the truth box and the predicted detection box at the same time, denoted as *C*. Thus, the calculation formula of GIoU can be given as

$$\text{GIoU} = \text{IoU} - \frac{|C\backslash(A\cup B)|}{|C|}. \tag{9}$$

Thus, the GIoU localization loss of the improved YOLO-v5s framework can be calculated as

$$Loss_{GIoU} = 1 - \text{GIoU}. \tag{10}$$

E. Complexity Analysis

The computation complexity of Algorithm 1 mainly comes from the model training of source domain D^S in the cloud servers and target domain D^T at edge servers. Define the training episodes for the source domain and target domain as $|K^S|$ and $|K^T|$, respectively. The sample space sizes of the training dataset of the source and target domains as n_S and n_T, respectively. Therefore, according to the complexity computation analysis of the training and testing, the computation complexity of Algorithm 1 is $O(|K^S|\cdot n_S + |K^T|\cdot n_T)$.

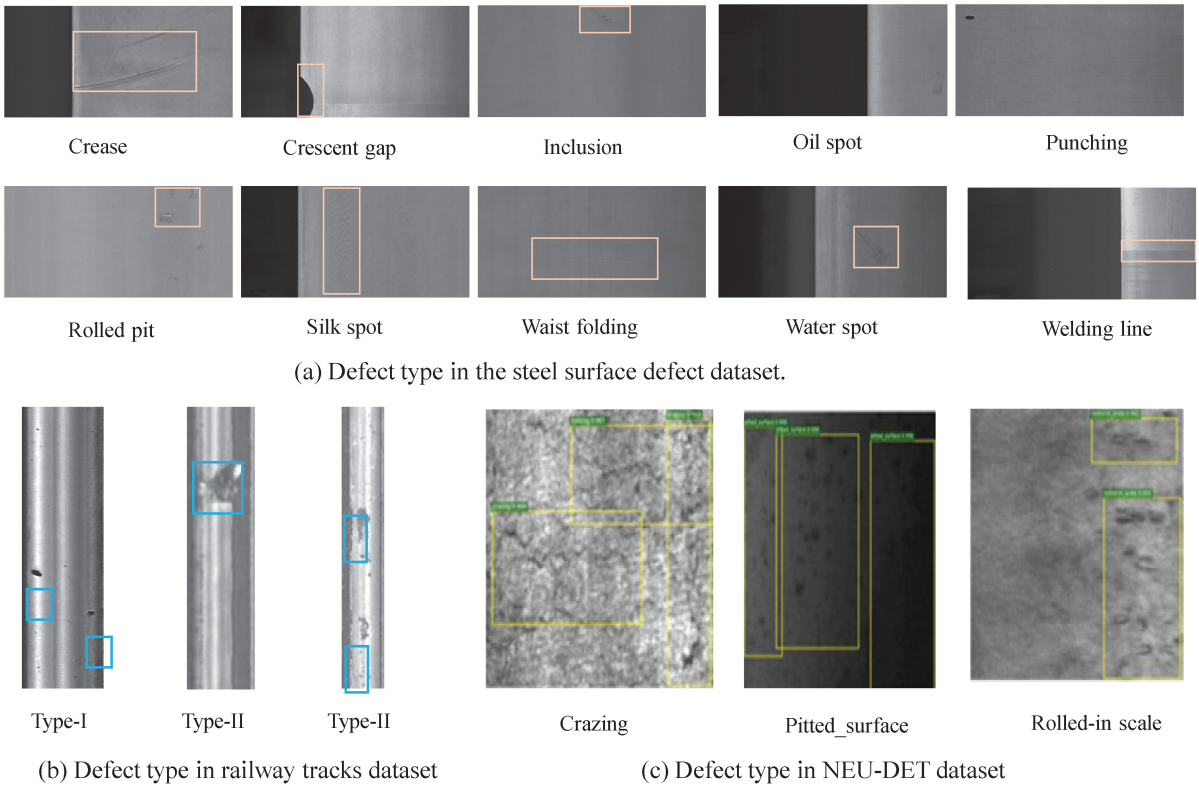
V. Performance Evaluation

A. Experiment Setup

To train the improved YOLO-v5s framework, the stochastic gradient descent method with cosine annealing and restart is used. The initial learning rate and final OneCycleLR learning rate is set as 0.00312 and 0.12, respectively. The batch size is 120, the momentum is 0.875, the weight_decay is 0.00039, the mosaic probability is 0.672 and the training is carried out for 2000 episodes. Other models are trained according to the training parameters recommended in the MMDetection framework. The training process uses servers with Intel(R) Xeon(R) Silver 4214 2.20GHz processor, 24GB of video memory GeForce RTX 3090 and RTX 2080Ti with 12GB of memory. The edge servers are distributed randomly in the considered MEC network. We set the wireless bandwidth, transmission power, noise power and antenna gain between the device layer and edge servers to be 20Mhz, [1.0, 1.5]W, 10^{-3}W and 5dBi, respectively. The average transmission rate from each edge server to the cloud computing center is 3Mbps.

B. Comparison Baselines and Metrics

In order to verify the performance of the proposed framework on surface defect detection, in this paper, we used original YOLO-v5s, Faster R-CNN [34], SSD [46] and CenterNet [36] algorithms as comparison baselines, and pre-train these models on the steel surface defect dataset, and then fine-tune the pre-trained model of all models on the railway tracks



(a) Defect type in the steel surface defect dataset.

(b) Defect type in railway tracks dataset

(c) Defect type in NEU-DET dataset

Fig. 7. Different defect types in the steel surface defect dataset, railway tracks dataset and NEU-DET dataset.

dataset at the edge servers respectively. Faster R-CNN, SSD and CenterNet algorithms are built using the MMDetection framework. Faster R-CNN uses ResNet-50 as the backbone network, SSD uses VGG16 as the backbone network, and CenterNet uses ResNet-50 as the backbone network.

In this paper, the performance evaluation metrics for experiments mainly include the detection accuracy and detection delay. We use Mean Average Precision (mAP) to evaluate detection accuracy. To evaluate detection delay performance, we calculate it in two ways, that is: i) detect directly at the edge servers through the TL; ii) use the cloud server for detection without TL. The detection delay from a-th defect detection point at device layer to b-th edge server can be calculated as $T^e_{ab} = \frac{s_a}{R_{ab}} + C_{ab}$, where $R_{ab} = B_a log_2(1 + \frac{P_a|h_{ab}|^2}{\sigma^2})$ is the transmission rate, C_{ab} is the inference delay to run detection models at b-th edge server. s_a, B_a, P_a, h_{ab} and σ^2 represent the data size of defect image, network bandwidth, transmit power, channel gain and noise power, respectively. Similarly, the detection delay at cloud server can be calculated as $T^c_{ab} = \frac{s_a}{R_{ab}} + \frac{s_a}{R_b} + C_{a0}$, where R_b is the transmission rate from b-th edge server to cloud server, C_{a0} is the inference delay to run these detection models at the cloud server.

C. Dataset Description

Fig. 7 shows different defect types in the steel surface defect dataset and railway tracks dataset, NEU-DET dataset. We use the steel surface defect dataset[1] as the source domain,

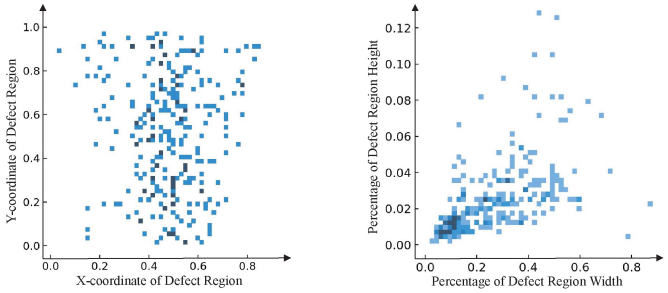
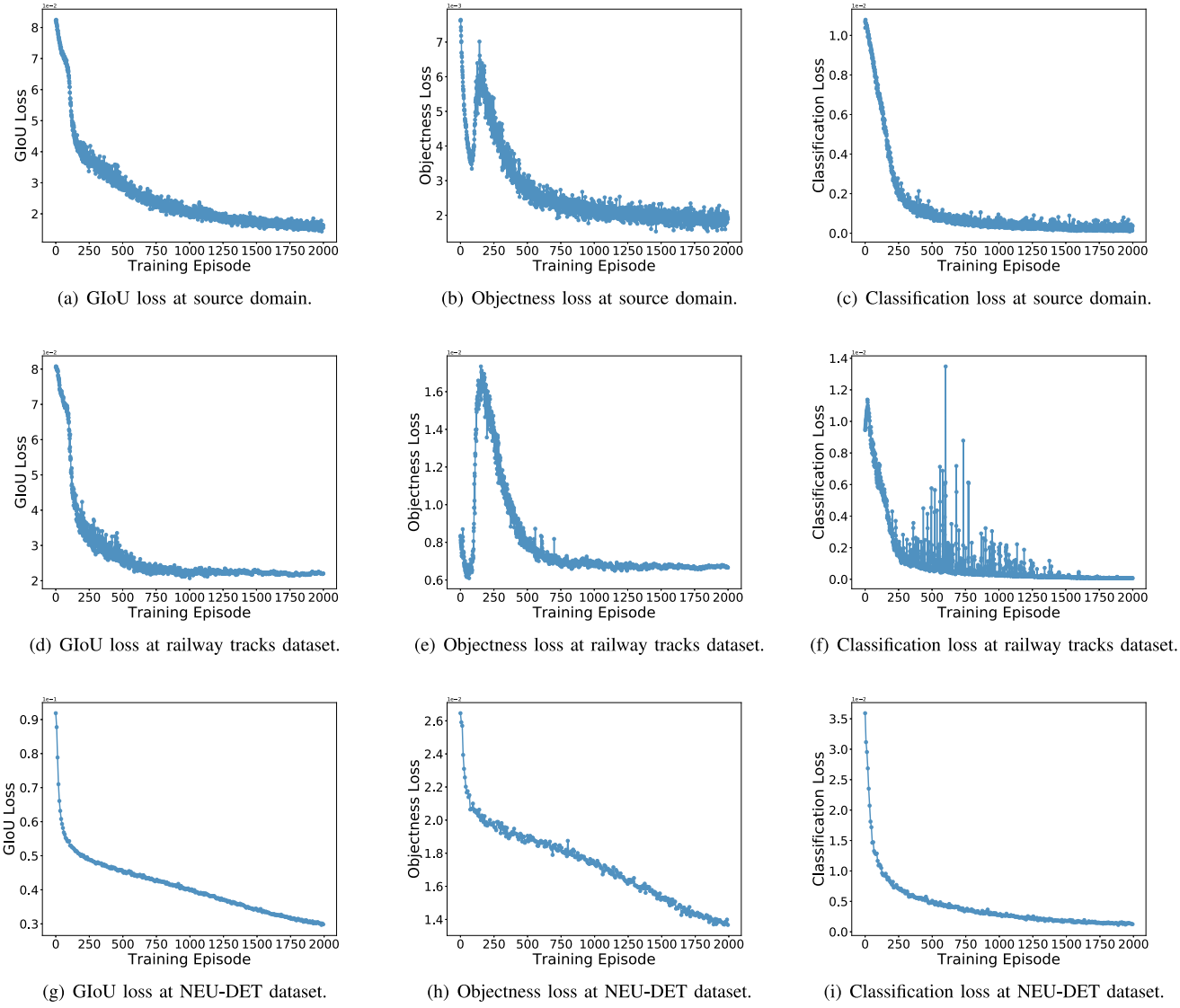


Fig. 8. Railway tracks image defect distribution.

and the railway tracks dataset and NEU-DET dataset[2] as different target domains, respectively. Specifically, the source domain dataset is steel surface defects collected in actual industrial manufacturing environments, which is selected from the Kaggle platform. It contains ten types of surface defects (e.g., 'punching', 'weld seam', 'crescent notch' and 'water spot'). The dataset includes 2294 images, each of which contains at least one surface defect type. The specific surface defect types are shown in Fig. 7(a). The railway tracks dataset [47] is obtained by the segmented shooting of railway tracks. The dataset includes 195 images, each image has at least one surface defect, and its defect types are divided into two types: Type-I type and Type-II type, as shown in Fig. 7(b). Type-I defects are manifested as black holes with obvious differences from the surface of the rail, and hole defects appear at the head of the rail in a random manner. Type-II defects appear

[1]https://www.kaggle.com/zhangyunsheng/defects-class-and-location.

[2]https://www.kaggle.com/datasets/zy12345/neudet



(a) GIoU loss at source domain. (b) Objectness loss at source domain. (c) Classification loss at source domain.

(d) GIoU loss at railway tracks dataset. (e) Objectness loss at railway tracks dataset. (f) Classification loss at railway tracks dataset.

(g) GIoU loss at NEU-DET dataset. (h) Objectness loss at NEU-DET dataset. (i) Classification loss at NEU-DET dataset.

Fig. 9. GIoU loss, objectness loss, and classification loss of proposed framework on different datasets at different training episodes.

as the corrugated folds that appear at the head of the rail in a repeatable and periodic pattern. Fig. 7(c) provides a visual representation of different categories of surface defects present in NEU-DET dataset, namely 'crazing', 'pitted surface', and 'rolled-in scale', among others. These surface defects represent various types of irregularities or anomalies that can occur on surfaces, such as cracks, indentations, and scales.

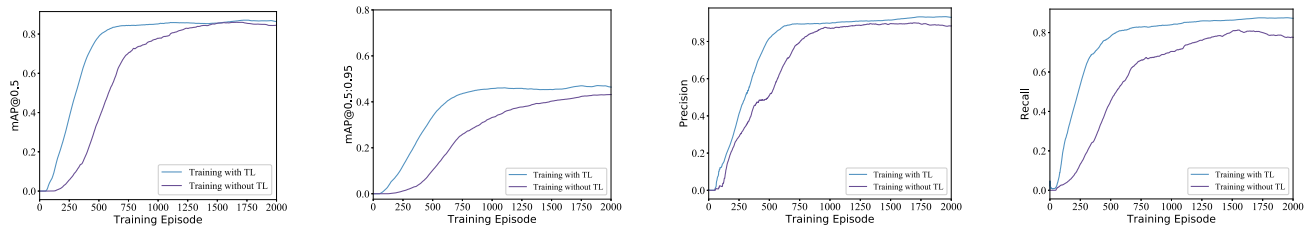
We use mosaic data augmentation during model training to enrich the defect detection datasets. Mosaic data enhancement stitches together four images obtained by randomly cropping or randomly scaling the surface defect image in a random distribution manner to form a new image, which also includes defect box and defect type information. This splicing and combination method greatly enriches the training dataset, and random scaling adds many tiny defects to the images generated by data enhancement, increasing the probability of tiny defects in the images. Fig. 8 shows the defect distribution map of the original image in the railway tracks dataset. The left subfigure shows the location distribution of the surface defects. It can be seen that the defect distribution in the upper and lower directions of rail is relatively uniform, and mainly distributed in the middle. From the right subfigure, we can observe that the surface defect area is very small, and most of the surface defect region is less than 20% of the width of the image and less than 2% of the height of the image. To train and evaluate the improved YOLO-v5s framework, we employed a common dataset splitting strategy, where 80% of the dataset was used for training and the remaining 20% was reserved for testing.

D. Convergence of Proposed Framework

Fig. 9 shows the convergence performance of the proposed framework on the source domain, railway tracks dataset and NEU-DET dataset. We can clearly observe that as the number of iterations increases, GIoU loss, objectness loss and classification loss will eventually reach the state of convergence. Fig. 9(a) - Fig. 9(c) show different losses on the source domain. We can easily observe that GIoU loss and

TABLE II
DETECTION ACCURACY OF DIFFERENT DEFECT DETECTION MODELS UNDER DIFFERENT TRAINING STRATEGIES

Model	Railway tracks dataset				NEU-DET dataset			
	Training with TL RTX 3090	Training without TL RTX 3090	Training with TL RTX 2080Ti	Training without TL RTX 2080Ti	Training with TL RTX 3090	Training without TL RTX 3090	Training with TL RTX 2080Ti	Training without TL RTX 2080Ti
Faster R-CNN	0.758	0.740	0.752	0.735	0.739	0.726	0.744	0.723
SSD	0.738	0.722	0.729	0.713	0.697	0.675	0.693	0.670
CenterNet	0.781	0.751	0.771	0.742	0.764	0.751	0.769	0.748
YOLO v5s	0.885	0.858	0.877	0.852	0.792	0.770	0.789	0.766
Proposed	**0.901**	**0.879**	**0.892**	**0.874**	**0.801**	**0.793**	**0.799**	**0.788**



(a) mAP@0.5. (b) mAP@0.5:0.95. (c) Precision. (d) Recall.

Fig. 10. Performance of mAP@0.5, mAP@0.5:0.95, precision, and recall of the proposed framework at different training episodes on railway tracks dataset.

classification loss curves are relatively smooth and gradually decrease with the increase of training episodes, while the objectness loss curve is not very stable. To explain it, it may be due to the setting of the initial learning rate, which causes the proposed framework to fall into a local optimum. As the training continues and the adjustment of the learning rate, the neural network will gradually find the global optimal solution, and the network loss will gradually decrease until it converges. Fig. 9(d) - Fig. 9(f) show the performance of the proposed framework on the railway tracks dataset, we can also observe that GIoU loss gradually decreases as the number of iterations increases, and the curve is relatively smoother, which is similar to the source domain. The curve of objectness loss and classification loss is not stable. This is because the surface defects are relatively tiny and the uneven distribution of surface defects in the railway tracks dataset. The proposed framework pays more attention to fitting the position information of the surface defects at first. Subsequently, as the number of iterations increases, the impact of GIoU loss on the overall loss becomes smaller. The proposed framework begins to focus on optimizing the objectness loss and classification loss, the network loss will gradually converge. Fig. 9(g) - Fig. 9(i) show the GIoU loss, objectness loss and classification loss on NEU-DET dataset. It can be observed that all three curves exhibit a similar decreasing trend over the training iterations. This indicates that the proposed framework is effectively learning to localize objects, distinguish them from the background, and classify them accurately on the NEU-DET dataset.

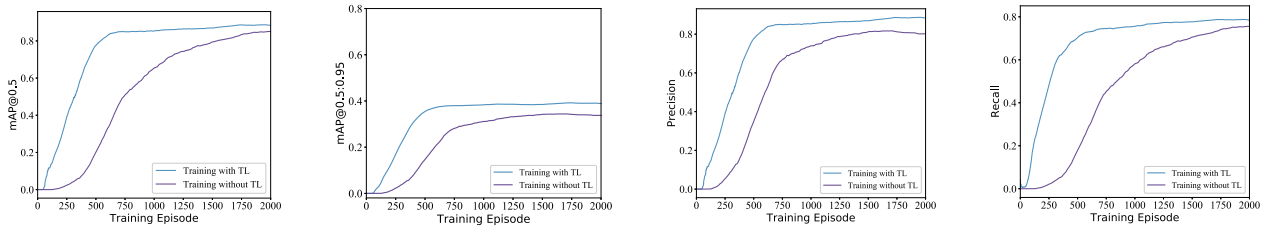
E. Performance Comparison of Different Defect Detection Models Under Different Training Strategies

In this subsection, we evaluate the detection performance of the proposed framework compared with the baseline scheme under two different training strategies (i.e., Training with TL and Training without TL) at two different target domains. As shown in Table II, we can see that the detection accuracy has improved after using the TL method in different detection models, and our proposed framework can improve the detection accuracy by 1.8% based on RTX 2080Ti on the railway tracks datasets. In addition, we can also observe the proposed framework has increased the detection accuracy by 1.6% and 2.1% compared with the original YOLO-v5s model based on RTX 3090 and RTX 2080Ti, respectively. The main reason is that the proposed framework with the spatial and channel attention mechanism, and the additional detection head increases the complexity of the detection model. The model feature extraction ability becomes stronger. It can better fit the surface defect information and capture surface defect regions to a certain extent in the considered MEC network.

In terms of indicators of mAP@0.5, mAP@0.5:0.95, precision, and recall at different episodes, we can clearly observe that these performance evaluation metrics gradually converge as the number of iterations increases, as shown in Fig. 10 and Fig. 11. To be more clear, it can be noticed that the performance of mAP@0.5 (Fig. 10(a), 11(a)), mAP@0.5:0.95 (Fig. 10(b), 11(b)), precision (Fig. 10(c), 11(c)) and recall rate (Fig. 10(d), 11(d)) without TL begin to converge after 1500 episodes, while the model is trained through TL method begins to convergence after 750 episodes. Besides, we can also observe that when the original YOLO-v5s framework training without TL, the convergence curve of average detection precision and recall is not as smooth as that training with TL. This is because due to the limited training samples in the target domain dataset and the uneven distribution of surface defects, which leads to the training process is not stable enough. However, the proposed framework with transfer learning has prior knowledge, and the learning efficiency and stability of the training process are higher.

F. Real-Time Analysis of Different Defect Detection Models

In this subsection, we will discuss the real-time performance of different defect detection models deployed at edge servers



(a) mAP@0.5. (b) mAP@0.5:0.95. (c) Precision. (d) Recall.

Fig. 11. Performance of mAP@0.5, mAP@0.5:0.95, precision, and recall of the proposed framework at different training episodes on NEU-DET dataset.

TABLE III
DETECTION TIME AND FPS PERFORMANCE OF DIFFERENT DEFECT DETECTION ALGORITHMS

Model	Railway tracks dataset				NEU-DET dataset			
	Detection time (ms) on GPU 2080Ti	Detection time (ms) on GPU 3090	FPS on GPU 2080Ti	FPS on GPU 3090	Detection time (ms) on GPU 2080Ti	Detection time (ms) on GPU 3090	FPS on GPU 2080Ti	FPS on GPU 3090
Faster R-CNN	27.5	23.8	37.7	46.6	29.4	23.9	11.8	14.9
SSD	26.8	22.4	43.86	44.6	29.1	23.2	59.72	66.87
CenterNet	11.7	10.4	89.3	96.2	13.1	11.5	71.37	79.76
YOLO v5s	**2.7**	**2.4**	**384.6**	**416.7**	3.4	3.0	**133.7**	**148.2**
Proposed	3.2	2.8	312.5	357.1	**3.3**	**2.9**	118.4	131.5

and cloud server, respectively. Specifically, we perform different defect detection models including Faster-RCNN, SSD, CenterNet, and YOLO-v5s. The GeForce RTX 3090 graphics card with 24 GB of memory and the GeForce RTX 2080 Ti graphics card with 12 GB of memory is used to simulate the cloud servers and edge servers to complete the defect detection task. The performance of FPS is shown in Table III. To explain it, Faster R-CNN is a single-stage detection model that has the longest inference time. The FPS is only 37.7 and 46.6 at the edge server and cloud server, respectively. Besides, YOLO v5s model has the fastest inference speed among all models, with FPS reaching 384.6 and 416.7. The FPS of the improved proposed framework reaches 312.5 and 357.1 at the edge server and cloud server, respectively. This is because the original YOLO-v5s network structure is improved and a new detection head is added, which increases the complexity of the neural network, resulting in higher FPS. In addition, we can observe that the FPS is lower on NEU-DET dataset, which can be attributed to the larger number and size of the objects, resulting in slower inference speed. It can still meet the inference speed requirements of industrial manufacturing environments for surface defect detection in MEC networks.

Fig. 12 compares the performance of detection delay of different detection models deployed at edge servers and cloud server, respectively (The values are shown in Table III). We can easily observe that the detection delay at the edge servers is lower than that at the cloud servers, and as the inference speed increases, the detection delay gap between the cloud server and edge servers for detection becomes more significant. Specifically, the detection delay of the improved YOLO-v5s framework on the cloud server is 2.7 times than that detection on the edge servers. This is because, with the improvement of the model inference speed, the transmission time of defect image data from the device layer to the edge layer has become an important factor affecting the detection delay. Although the computing power of the cloud server is

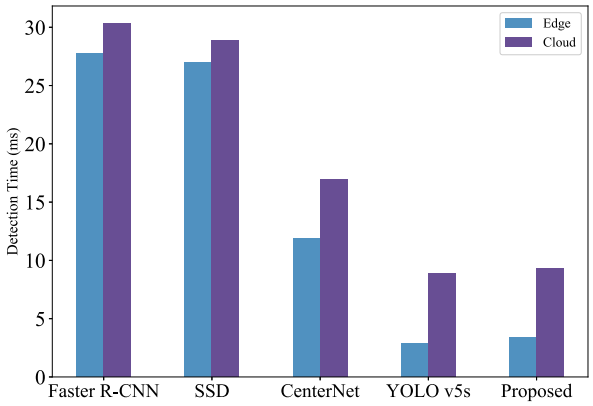


Fig. 12. Detection delay of different defect detection models.

stronger, the long data transmission link leads to higher transmission delay excessively. The GPU performance difference used in this experiment is small, however even if the inference speed of the cloud server is many times faster than edge servers, we calculate that the detection delay is still higher than that at the edge servers. On the other hand, with the increase of uploaded defect detection tasks, the network bandwidth of the cloud server will gradually become the bottleneck of defect detection efficiency. Therefore, the proposed framework can provide a more stable and real-time defect detection service for tiny surface defects in MEC networks.

VI. CONCLUSION

In this paper, we have proposed a real-time surface defect detection framework to improve the detection performance on tiny surface defects that supports the neural network knowledge transfer in different feature spaces based on the TL with MEC networks. Specifically, we pre-train a general surface defect detection model with the source domain dataset at the cloud servers first. Then we fine-tune the pre-trained

model with the different target domain datasets through TL at edge servers according to the limited local specific defect type data for different industrial manufacturing environments. To address the challenge of limited training data for tiny defects, we have improved the original YOLO-v5s framework by introducing the spatial and channel attention mechanism, and adding an additional detection head in the head network to enhance the detection performance. Experimental results have demonstrated that our proposed framework has superior performance in terms of improving detection accuracy and reducing the detection delay in the considered MEC network.

REFERENCES

[1] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4225–4234, Jul. 2019.

[2] G. Nain, K. Pattanaik, and G. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *J. Manuf. Syst.*, vol. 62, pp. 588–611, Jan. 2022.

[3] H. Yang, Y. Wang, J. Hu, J. He, Z. Yao, and Q. Bi, "Deep learning and machine vision-based inspection of rail surface defects," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, Dec. 2021.

[4] D. Ai, G. Jiang, S.-K. Lam, P. He, and C. Li, "Computer vision framework for crack detection of civil infrastructure–A review," *Eng. Appl. Artif. Intell.*, vol. 117, Oct. 2023, Art. no. 105478.

[5] J. Feng, Q. Li, Q. Xiao, and G. Wang, "A method of rayleigh wave combined with coil spatial pulse compression technique for crack defects detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, Feb. 2023.

[6] M. Niu, K. Song, L. Huang, Q. Wang, Y. Yan, and Q. Meng, "Unsupervised saliency detection of rail surface defects using stereoscopic images," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2271–2281, Mar. 2021.

[7] J. Tang et al., "Anomaly detection in social-aware IoT networks," *IEEE Trans. Netw. Service Manag.*, early access, Feb. 6, 2023, doi: 10.1109/TNSM.2023.3242320.

[8] T. Cao, L. Liu, K. Wang, and J. Li, "A fractional integral and fractal dimension-based deep learning approach for pavement crack detection in transportation service management," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4201–4212, Dec. 2022, doi: 10.1109/TNSM.2022.3197457.

[9] Y. Liu, H. Xiao, J. Xu, and J. Zhao, "A rail surface defect detection method based on pyramid feature and lightweight convolutional neural network," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, Apr. 2022.

[10] H. Jin, W. Wu, X. Shi, L. He, and B. B. Zhou, "TurboDL: Improving the CNN training on GPU with fine-grained multi-streaming scheduling," *IEEE Trans. Comput.*, vol. 70, no. 4, pp. 552–565, Apr. 2021.

[11] H.-C. Shin et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[12] Y.-C. Lai et al., "Task assignment and capacity allocation for ML-based intrusion detection as a service in a multi-tier architecture," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 672–683, Mar. 2023, doi: 10.1109/TNSM.2022.3203427.

[13] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[14] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 596–630, 1st Quart., 2020.

[15] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable intrusion detection system for IoT: A deep transfer learning based approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 1006–1017, Jan. 2023.

[16] Y. Singh and A. Biswas, "Robustness of musical features on deep learning models for music genre classification," *Expert Syst. Appl.*, vol. 199, Aug. 2022, Art. no. 116879.

[17] Z. Zhang, P. Zhao, P. Wang, and W.-J. Lee, "Transfer learning featured short-term combining forecasting model for residential loads with small sample sets," *IEEE Trans. Ind. Appl.*, vol. 58, no. 4, pp. 4279–4288, Jul./Aug. 2022.

[18] X. Ni, H. Liu, Z. Ma, C. Wang, and J. Liu, "Detection for rail surface defects via partitioned edge feature," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5806–5822, Jun. 2022.

[19] B. Yang et al., "A joint energy and latency framework for transfer learning over 5G industrial edge networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 531–541, Jan. 2022.

[20] H. Lin, J. Hu, X. Wang, M. F. Alhamid, and M. J. Piran, "Toward secure data fusion in industrial IoT using transfer learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7114–7122, Oct. 2021.

[21] W. Qi, R. Zhang, J. Zhou, H. Zhang, Y. Xie, and X. Jing, "A resource-efficient cross-domain sensing method for device-free gesture recognition with federated transfer learning," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 393–400, Mar. 2023, doi: 10.1109/TGCN.2022.3233825.

[22] S. Shao, S. McAleer, R. Yan, and P. Baldi, "Highly accurate machine fault diagnosis using deep transfer learning," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2446–2455, Aug. 2019.

[23] W. Li et al., "A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges," *Mech. Syst. Signal Process.*, vol. 167, Mar. 2022, Art. no. 108487.

[24] T. Gai *et al.*, "Flexible hotspot detection based on fully convolutional network with transfer learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4626–4638, Nov. 2022.

[25] C. Liu, Z. Wei, D. W. K. Ng, J. Yuan, and Y.-C. Liang, "Deep transfer learning for signal detection in ambient backscatter communications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1624–1638, Mar. 2021.

[26] Z. Zhu, G. Han, G. Jia, and L. Shu, "Modified densenet for automatic fabric defect detection with edge computing for minimizing latency," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9623–9636, Oct. 2020.

[27] M. Liu, Z. Li, Y. Li, and Y. Liu, "A fast and accurate method of power line intelligent inspection based on edge computing," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, Feb. 2022.

[28] W. Xia, J. Chen, W. Zheng, and H. Liu, "A multi-target detection based framework for defect analysis of electrical equipment," in *Proc. IEEE ICCCBDA*, Apr. 2021, pp. 483–487.

[29] S. Zhao, J. Wang, J. Zhang, J. Bao, and R. Zhong, "Edge-cloud collaborative fabric defect detection based on Industrial Internet Architecture," in *Proc. IEEE INDIN*, Jul. 2020, pp. 483–487.

[30] S. Liang et al., "Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25345–25360, Dec. 2022.

[31] W. Tang, Q. Yang, X. Hu, and W. Yan, "Edge intelligence for smart EL images defects detection of PV plants in the IoT-based inspection system," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3047–3056, Feb. 2023.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE/CVF CVPR*, Jun. 2014, pp. 580–587.

[33] R. Girshick, "Fast R-Cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 91–99.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE/CVF CVPR*, Jun. 2016, pp. 779–788.

[36] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proc. IEEE/CVF ICCV*, Oct. 2019, pp. 6569–6578.

[37] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big data*, vol. 3, no. 1, pp. 1–40, May 2016.

[38] J. Talukdar, S. Gupta, P. Rajpura, and R. S. Hegde, "Transfer learning for object detection using state-of-the-art deep neural networks," in *Proc. IEEE SPIN*, Feb. 2018, pp. 78–83.

[39] W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu, and L. Zhang, "Image-adaptive YOLO for object detection in adverse weather conditions," in *Proc. AAAI*, vol. 36, 2022, pp. 1792–1800.

[40] G. Li, Z. Ji, X. Qu, R. Zhou, and D. Cao, "Cross-domain object detection for autonomous driving: A stepwise domain adaptative YOLO approach," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 603–615, Sep. 2022.

[41] B. Chen, X. Wang, Q. Bao, B. Jia, X. Li, and Y. Wang, "An unsafe behavior detection method based on improved YOLO framework," *Electronics*, vol. 11, no. 12, p. 1912, Jun. 2022.
[42] J. Wang, Y. Chen, Z. Dong, and M. Gao, "Improved YOLOv5 network for real-time multi-scale traffic sign detection," in *Proc. Neural Comput. Appl.*, Dec. 2022, pp. 1–13.
[43] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF CVPR Workshops*, 2020, pp. 390–391.
[44] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8759–8768.
[45] J. Xue, Y. Zheng, C. Dong-Ye, P. Wang, and M. Yasir, "Improved YOLOv5 network method for remote sensing image-based ground objects recognition," *Soft Comput.*, vol. 26, no. 20, pp. 10879–10889, May 2022.
[46] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Springer ECCV*, Sep. 2016, pp. 21–37.
[47] H. Yu et al., "A coarse-to-fine model for rail surface defect detection," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 3, pp. 656–666, Mar. 2019.

Hui Li (Student Member, IEEE) received the B.S. degree from Northeast Agricultural University, Harbin, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. His current research interests include edge computing and caching, and edge intelligence.

Xiuhua Li (Member, IEEE) received the B.S. and M.S. degrees from the Harbin Institute of Technology, China, in 2011 and 2013, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada, in 2018. He is currently a Tenure-Track Assistant Professor with the School of Big Data and Software Engineering, Chongqing University, Chongqing, China, and also a member of Haihe Laboratory, ITAI. He is the Head of the Institute of Intelligent Software and Services Computing associated with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University. Focusing on the research of edge computing and caching, and edge intelligence, he has published more than 90 technical papers in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, ICC, and GLOBECOM.

Qilin Fan (Member, IEEE) received the B.E. degree from the College of Software Engineering, Sichuan University, Chengdu, China, in 2011, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2017. She is currently an Associate Professor with the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. Her research interests include network optimization, mobile edge computing and caching, network virtualization, and machine learning.

Qingyu Xiong received the M.Sc. degree from Chongqing University, Chongqing, China, in 1991, and the Ph.D. degree in electrical and electronic systems engineering from Kyushu University, Fukuoka, Japan, in 2002. He is currently a Professor with the School of Big Data and Software Engineering, Chongqing University. His current research interests include intelligent system and intelligent computation, and pervasive computation and embedded system. He is a member of the China Computer Federation.

Xiaofei Wang (Senior Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, China, and the M.S. and Ph.D. degrees from Seoul National University, Seoul, South Korea. He was a Postdoctoral Fellow with The University of British Columbia, Vancouver, Canada, from 2014 to 2016. He is currently a Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. Focusing on the research of edge computing, edge intelligence, and edge systems, he has published more than 150 technical papers in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON MULTIMEDIA, INFOCOM, and ICDCS. He was the recipient of the IEEE ComSoc Fred W. Ellersick Prize in 2017. He received the IEEE ComSoc Asia–Pacific Outstanding Paper Award in 2022.

Victor C. M. Leung (Life Fellow, IEEE) is a Distinguished Professor of Computer Science and Software Engineering with Shenzhen University, China. He is also an Emeritus Professor of Electrical and Computer Engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems, The University of British Columbia, Canada. His research is in the broad areas of wireless networks and mobile systems and he has published widely in these areas. He received the APEBC Gold Medal in 1977, NSERC Postgraduate Scholarships from 1977 to 1981, the IEEE Vancouver Section Centennial Award, UBC Killam Research Prize in 2011, the Canadian Award for Telecommunications Research in 2017, the IEEE TCGCC Distinguished Technical Achievement Recognition Award in 2018, and the ACM MSWiM Reginald Fessenden Award in 2018. He coauthored papers that won the IEEE ComSoc Fred W. Ellersick Prize in 2017, the IEEE Systems Journal Best Paper Award in 2017, the IEEE CSIM Best Journal Paper Award in 2018, and the IEEE TCGCC Best Journal Paper Award in 2019. He is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, and IEEE ACCESS. He is named in the current Clarivate Analytics list of "Highly Cited Researchers." He is a Fellow of the Royal Society of Canada (Academy of Science), the Canadian Academy of Engineering, and the Engineering Institute of Canada.