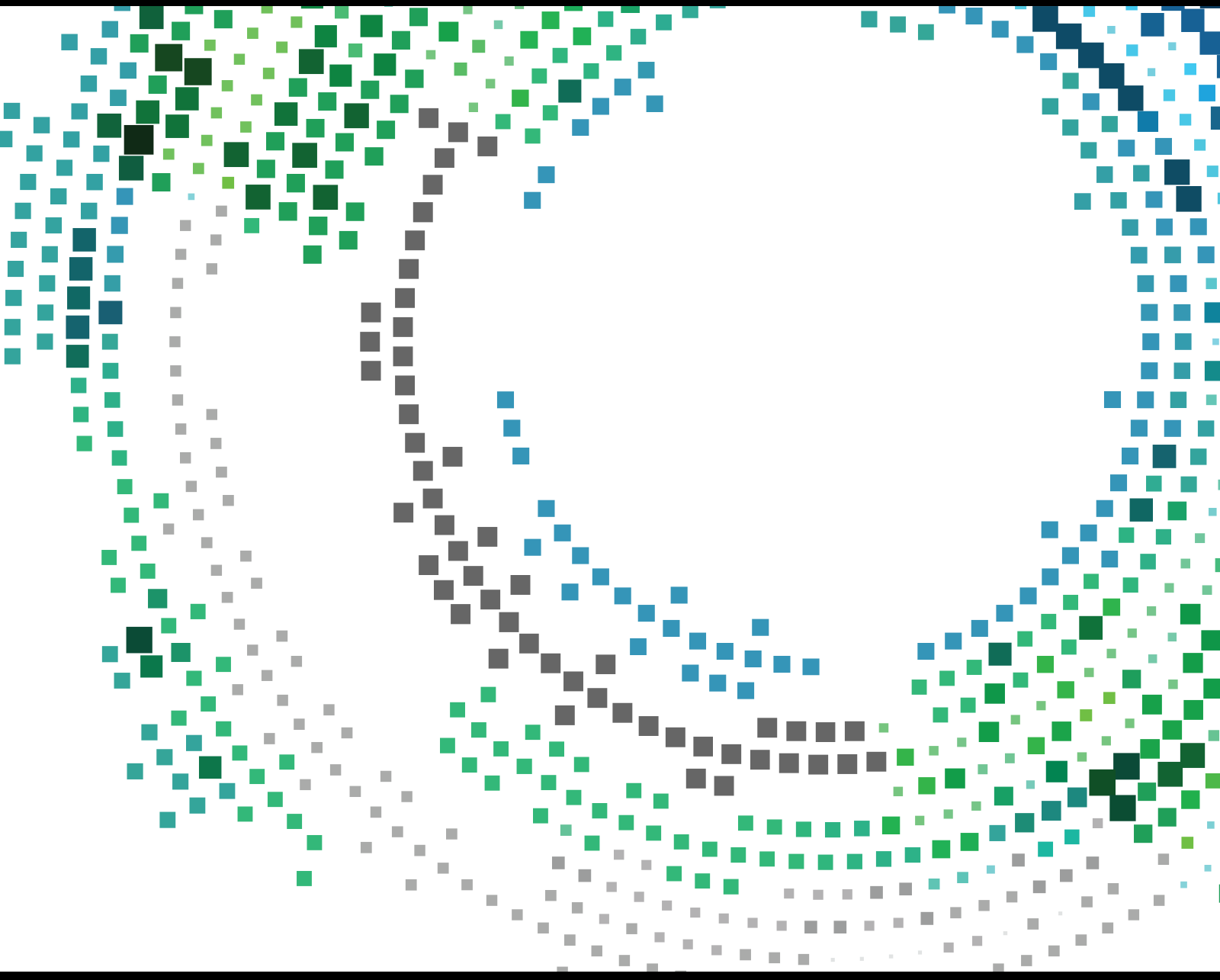
Mobile Information Systems

Architecture, Algorithms, and Applications of Future Generation Wireless Communication Networks

Lead Guest Editor: Yuexia Zhang
Guest Editors: Xiong Luo and Basem M. ElHalawany



Architecture, Algorithms, and Applications of Future Generation Wireless Communication Networks

Mobile Information Systems

Architecture, Algorithms, and Applications of Future Generation Wireless Communication Networks

Lead Guest Editor: Yuexia Zhang
Guest Editors: Xiong Luo and Basem M. ElHalawany

Contents

Hindawi
Mobile Information Systems
Volume 2023, Article ID 1687672, 15 pages
https://doi.org/10.1155/2023/1687672

Research Article

Joint Optimization of Resources in Fog-Radio Access Network with Binary Computation Offloading

Wenle Bai[1] **and Zhuoqi Wang**[2]

[1]*Information Science and Technology, North China University of Technology, Shijingshan District, Beijing 100043, China*
[2]*School of Information Science and Technology, North China University of Technology, Shijingshan District, Beijing 100043, China*

Correspondence should be addressed to Zhuoqi Wang; yugyuailun@qq.com

Received 1 August 2022; Revised 25 August 2022; Accepted 27 September 2022; Published 21 April 2023

Academic Editor: Yuexia Zhang

With the dramatic increase in the number of emerging Internet services, the Fog-Radio Access Network (F-RAN) has recently emerged as a promising paradigm to enhance high-load task processing capabilities for mobile devices, such as the Internet of things (IoT) and mobile terminals. Hence, it becomes a challenge for the F-RAN to reduce the offloading cost by designing an effective offloading strategy and rational planning of limited network resources to improve the quality of experience (QoE). This article investigates the F-RAN with a binary offload policy. It proposes an intelligent algorithm capable of optimally adapting to task offload policy, fog computing resource allocation, and offload channel resource allocation. To evaluate the offloading strategy intuitively, we design a system utility metric defined as a delay-energy weighted sum. The joint optimization problem is converted into a convex problem based on this metric, i.e., a mixed integer nonlinear programming (MINLP) problem. A novel algorithm based on improved double-deep Q neural networks is DDQN, which is proposed to address this problem. Furthermore, an action space mapping method in the DDQN framework is presented to obtain offloading decisions. Extensive experimental data indicate that the proposed DDQN algorithm can effectively reduce the offloading cost and is adaptable to different offloading scenarios.

1. Introduction

Nowadays, with the rapid development of mobile communication technologies represented by the fifth generation (5G) and the wide application of artificial intelligence, our society has become increasingly intelligent, and the number of resulting Internet of things (IoT) services [1] has increased dramatically. However, several highly anticipated applications including virtual reality (VR), augmented reality (AR), and the Internet of vehicles (IoV) necessitate extremely low latency and energy consumption while being constrained by cost and computational resources. Fog computing, also known as the fog-radio access network (F-RAN) [2] and mobile fog computing (MFC) [3], was established to satisfy the needs of IoT services, fully exploit the benefits of IoT, and overcome the problem of limited computing resources of user equipment (UE). Queuing delays caused by offloading tasks to remote cloud servers [4] through the core network can be reduced by allowing UEs to offload tasks to nearby fog access points (F-APs) for processing. Meanwhile, the addition of fog servers reduces the communications between the base station and the core network significantly [5], thus relieving the load of the backhaul network.

In practice, however, in practical applications, fog servers' computational and network resources are not unlimited. Different resource allocation schemes significantly impact users' quality of experience (QoE) [6] of users. Hence, it becomes a challenge in the F-RAN to design an effective offloading strategy with proper planning of limited network resources. Several existing studies have proposed offloading methods to solve these problems. Goudarzi et al. in [7] proposed a new technique for task layout based on the memetic algorithm to maximize the number of tasks computed in parallel on each server. In [8], the concept of

distributed decision-making is proposed. The algorithm is distributed to each device, and the offloading decision will be generated directly by the local device, which dramatically reduces the complexity of the network. However, as information is not shared among each device, it is obvious for server congestion to occur. Lan et al. in [9] divided the offloading time into peak and off-peak time. Then, different offloading algorithms are applied for each case to find the offloading decision of tasks.

1.1. Related Work. In the existing works, most of them transform offloading as a constrained convex optimization (CCO) problem with different metrics and constraints chosen, such as service delay, network capacity, backhaul rate, and energy consumption [4]. Wang et al. in [10] jointly optimize the computation of offloading decisions, resource allocation, and content caching policies and transform the original problem into a convex optimization problem. Then, they offered an alternating direction method of multipliers-based to solve the convex problem. Ma et al. proposed a genetic convex optimization algorithm (GCOA) in [11] to satisfy the diverse quality of service requirements of different users. In [12], Jiang et al. transformed the offloading problem into a nondeterministic polynomial solution (NDPS) problem with the objective of minimizing the delay.

In [13, 14], the authors have provided a novel method to solve the decision and resource allocation problems in the F-RAN. They showed the offloading decisions of tasks in binary variables. By deriving the total offloading cost expression, the allocation problem can be converted into a mixed integer programming (MIP) problem. In [15], the resource optimization problem is formulated as a quadratically constrained quadratic programming (QCQP) problem. Then, the optimal offloading decision is obtained by solving the QCQP problem. Tang et al. [16] innovatively defined the offloading optimization problem as a decentralized partially observable Markov decision process (Dec-POMDP). Each device gives the offloading decision based on its local observation of the environment. Meanwhile, to reduce the computational complexity of the CCO problem, the coordinate descent method [17] and the convex relaxation method [18] have been proposed.

On the other hand, game theory and its variants are also adopted to solve offloading problems [19–22]. In [20], a distributed game method with group perception is studied to ensure the maximum utilization of resources. Jie et al. proposed a Stackelberg-based online task offloading scheme in [21]. Shuchen and Waqas [22] proposed a multiuser partial computation offloading strategy based on game theory. Based on this, the authors in [22, 23] added intelligent gateways with migration functions to the network to relieve server congestion. The abovementioned methods, nonetheless, are studied under the assumption that the transfer probabilities of each state and the complete system model can be obtained, while such assumptions are too ideal in realistic scenarios.

Furthermore, in the F-RAN, a key research problem is the joint design of computational resource allocation and channel resource allocation [4, 24]. In [25], an iterative algorithm is proposed to solve the problem of joint allocation of computational and radio resources during offloading. In [26], a multistage stochastic planning approach for offloading tasks with high computational overhead is investigated. Cao et al. [20] have studied the optimal and suboptimal resource allocation problems in F-RANs based on nonorthogonal multiple access techniques. In [27], the main problem of joint computation and communication resource allocation for a multiuser is that the multiserver system is divided into subproblems, which are then solved using matching and sequential convex programming algorithms. In [28], Liu et al. considered a fog network with energy harvesting, where each user gets energy from a hybrid access point (HAP). They aim to maximize the minimum energy balance among all users and jointly optimize the offloading time and fog resource allocation. Similarly, the authors in [29] proposed an energy-efficient computational offload-resource allocation (ECORA) scheme to optimize computational resource allocation and transmission power jointly. Gu et al. [30] combined the reputation mechanism with offloading. The system will assign a reputation value to each device. If a task is offloaded, the algorithm will allocate the computational resources to the device based on its reputation value. Not coincidentally, in work [31], the idea of pricing different resources was proposed. Gai et al. [32] proposed an EFRO model to manage the resources in the F-RAN.

In recent years, with the development of neural networks [33], deep learning has been increasingly applied to offloading computation. For instance, the authors in [34] proposed a joint offloading decision and resource allocation algorithm based on deep reinforcement learning (DRL). The computational offloading strategy for the case of the F-RAN with multiple UEs was studied in [35], and the total utility of UEs was optimized by using the DQN algorithm. Based on this, the authors in [35, 36] improved it by considering a computational offloading strategy for device-to-device (D2D) communication between UEs in the F-RAN. The DDQN algorithm has been used in the literature [37] to predict the offloading actions of UEs in semionline distribution tasks, while also calculating and updating the total reward after each offloading decision until it reaches its maximum value. In [38], deep reinforcement learning for online offloading (DROO) was proposed to solve the problem of generating decisions quickly in fast-fading channel conditions. In [39], the deep Q networks were used to predict each device with unknown channel state information that obtains its most suitable offloading pattern. Similarly, the LSMT networks and the double-deep Q networks were combined in [40] to obtain the offloading decision of tasks. In [41], Baccarelli et al. applied a network of CDDNs to mobile fog computing to generate offloading policies. However, most of the existing intelligent algorithms are premised on the assumption that a task is an indivisible whole. In real scenarios, parts of tasks can be split into multiple independent subtasks, which cannot be ignored. Other than that, the above intelligent algorithms allocate resources equally, which is too idealistic in practice. Hence,

a new intelligent algorithm is needed to tackle the problems of offloading policy and resource allocation in detachable task offloading.

1.2. Approach and Contributions. In this article, we propose a novel offloading framework in the F-RAN aiming at the independence of detachable tasks with double-deep Q-learning. In an F-RAN, there are multiple users, multiple fog access points, a remote cloud server, an edge router, and a core network layer. Users can offload their tasks to any server, such as the fog server or the cloud server, to maintain a high QoE while saving battery power.

Edge routers are arranged at the edges of the network, which manage all-fog computation resources. By collecting information about offloading tasks, communication channel status, and F-APs status, the edge router outputs offloading decisions for tasks and resource allocation decisions, which contain upload channel resource allocation decisions and fog computing resource allocation decisions, with the DDQN algorithm. Meanwhile, the proposed DDQN algorithm is being trained on the edge server. The core network layer consists of a large number of routes, which are mainly responsible for data routing and forwarding.

The main contributions of this article are as follows:

(1) A novel fog offloading framework is proposed, where both the offloading decision and the resource allocation policy of the task are determined by the edge router. The user uploads the task information to the edge router through the F-AP. Then, the router uses the DDQN algorithm trained by the edge server to give the offloading decision and resource allocation policy for each task based on the information.

(2) In the F-RAN, we model the system utility as a weighted sum of delay and energy consumption to compute all tasks. To minimize the system utility, a joint offloading decision and resource allocation problem for the F-RAN is proposed. The problem jointly optimizes the offloading decision, the fog computation resources, and the upload bandwidth allocated by the system to each task.

(3) A double-deep Q-learning-based offloading algorithm for the F-RAN and the DDQN algorithm is proposed, which consists of a main network and a target network. The DDQN generates the action space from the main network and uses the target network to evaluate the action at the next moment, improving the performance of the main network. Besides, these generated offloading decisions and resource allocation policies are stored in a public experience pool to further train and improve the double-deep Q networks.

(4) Simulation results show that the proposed DDQN algorithm has better convergence and lower average cost than the benchmark. Meanwhile, it is highly adaptive in multiuser and different focus scenarios.

The rest of this article is as follows: the offloading model and the closed-form expressions for delay and energy are in Section 2, as is the construction of a delay-energy weighted sum minimization problem. The DDQN algorithm is referred to in Section 3. Section 4 mainly provides the analysis of the simulation. Moreover, the conclusions are given in the last section.

2. System Model

In this article, we consider a Fog-Radio Access Network, as shown in Figure 1, consisting of N user equipment (UE), K fog access points (F-AP), a remote cloud server, an edge router, and a core network layer. The UE can be represented by a set $N = \{1, 2, \mathrm{I}, N\}$. Similarly, a set $\mathrm{K} = \{1\mathrm{I}..., K\}$ is used to denote the F-AP. These F-APs can provide computation services for the device, but they do not have the decision-making capability. Furthermore, the F-APs communicate with local devices through wireless.

Assume that each UE has M unrelated tasks to compute, denoted as $\mathrm{M} = \{1, 2, \ldots M\}$. At the beginning of time t, a UE has only one detachable task request, noted as $O_{nm} = (D_{nm}, C_{nm})$, where D_{nm} denotes the size of the offloading data for UE n's m-th task, i.e., the workload of the task which needs to be transmitted from the device to the server, and C_{nm} represents the number of revolutions required by the local device to process this task (expressed in cycles).

2.1. Communication Model. It is assumed that the environmental state of the F-RAN remains constant at the same moment. The wireless channel gain between the k-th AP and the n-th UE at time t is denoted by $h_{nk}(t)$. Besides, the channel gain follows the free-space path loss model. Then, the wireless channel gain at time t can be represented as follows:

$$h_{nk}(t) = A_d\left(\frac{3 \times 10^8}{4\pi f_c d_{nk}(t)}\right)^{d_e}, \tag{1}$$

where A_d denotes the antenna gain, f_c is the carrier frequency, and d_e is the path loss index. $d_{nk}(t)$ represents the linear distance from the n-th UE to the k-th AP.

Without loss of generality, assume that all F-APs use the same channel. The total uplink channel bandwidth is noted as B, which can be split into multiple mutually orthogonal subchannels. Furthermore, there is no mutual interference between these subchannels. According to Shannon's formula, the band utilization between the n-th UE and the k-th F-AP can be represented as follows:

$$r_{nk}(t) = \log_2\left(1 + \frac{P_n \cdot h_{nk}(t)}{\sigma^2}\right), \tag{2}$$

where P_n is the transmit power of device n and σ^2 denotes the power of white Gaussian noise.

Further, $b_n(t) \in (0, 1)$ is used to represent the proportion of channel resources allocated to the n-th UE at time

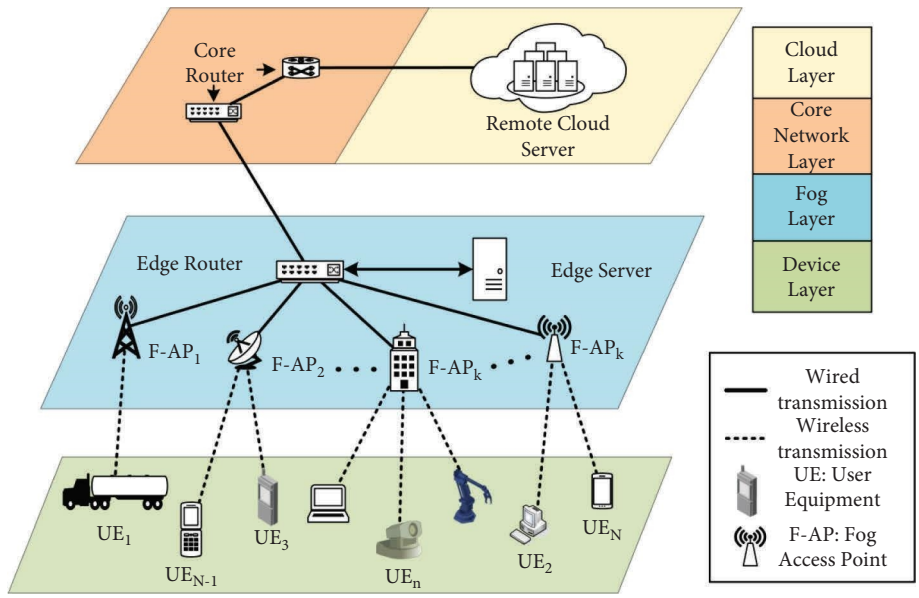


Figure 1: Architecture of the Fog-Radio Access Network.

t. The transmission delay Tup/nm of uploading task is as follows:

$$T_{nm}^{up} = \frac{D_{nm}}{r_{nk}(t)\, b_n(t)\, B}. \tag{3}$$

Meanwhile, it is accompanied by the energy consumption of the device during uploading. Let ε^{up} denote the energy consumption to upload 1 KB of data. The energy cost of the local device when offloading can be expressed as follows:

$$E_{nm}^{up} = \varepsilon^{up} D_{nm}. \tag{4}$$

2.2. Transmission Time Allocation Model. Here, a model of transmission time allocation for offloading is considered, as illustrated in Figure 2. If the computation task is generated, the UE will first send the data information O_{nm} as well as the distance $\mathrm{d}_n(t) = \{d_{nk}(t), \forall k \in \mathrm{K}\}$ between UE n and each F-AP to the edge router via the nearest fog node. The router will use the trained DDQN algorithm to give the offloading decision and resource allocation policy for each task. Additionally, the time cost of uploading the relevant information is represented as t_1.

Once a detachable task O_{nm} is offloaded, the UE will first send the task to the F-AP with the optimal channel state, forwarding it to the edge router. The edge route computes tasks by scheduling the computation resources of the fog server. After the computation, the results are returned to the device via the backhaul link. After the computation is completed, the results are delivered by the backhaul link to the device. Similarly, let t_2 denote the transmission time of tasks during offloading, and t_3 be the backhaul delay of the result.

Generally speaking, the size of the uploading information and the backhaul data is much smaller than the offloading task [13, 38, 40], while the downlink transmission rate is much faster than the uplink rate [40]; hence, the delay of t_1 and t_3 can be ignored. Thus, the transmission delay Ttr/nm for UE n's m-th offloading task can be approximated by $Ttr/nm \approx t_2$.

2.3. Offloading Computation Modes

2.3.1. Local Computing Mode. We use the binary variable $XL/nm \in \{0, 1\}$ to represent the decision of UE n's m-th task on the local side. $XL/nm = 1$ means the task will be executed locally, while $XL/nm = 0$ means it will be offloaded to the server. The computational capacity of device n is denoted by λ_n. As there is no transmission cost for the task in local mode, the delay TL/nm can be expressed as follows:

$$T_{nm}^{L} = \frac{C_{nm}}{\lambda_n}. \tag{5}$$

Meanwhile, the tasks are computed locally with energy consumption, defining the local energy consumption as follows:

$$E_{nm}^{L} = \varepsilon^{local} C_{nm}, \tag{6}$$

where ε^{local} indicates the amount of energy consumed by the CPU per cycle.

2.3.2. Fog Computing Mode. Using $XF/nm \in \{0, 1\}$ to indicate the decision of tasks at the fog side, if $XF/nm = 1$ ($XF/nm = 0$), the task will be computed in the fog server (not in the fog).

Consider the detachable task, which can be divided into multiple mutually independent subtasks, for the offloading decisions. Hence, multiple subtasks can be simultaneously assigned to different F-APs for parallel computation, which can fully utilize all-fog servers. Let the set $\{f_1, f_2, \ldots, f_K\}$ be

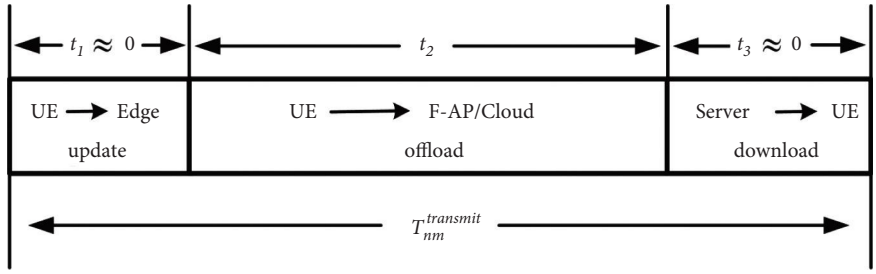


Figure 2: An example of transmission time allocation in the F-RAN for the m-th task of user n.

the computation resources of KF-APs managed by the edge router. Thus, the total computation resources F that are scheduled by the edge router is as follows:

$$F = \sum_{k=1}^{K} f_K. \tag{7}$$

The distance from the F-AP to the edge router is close to a high transmission rate, so the communication delay between both can be ignored. Accordingly, the time cost T_{nm}^{F} in the fog computing mode is as follows:

$$T_{nm}^{F} = T_{nm}^{up} + \frac{D_{nm}}{\varphi_n(t) F}, \tag{8}$$

where $\varphi_n(t) \in (0, 1)$ is the proportion of fog computation resources allocated to the n-th UE by the edge router at time t.

Concerning energy consumption, only the cost of the local device side is concerned, while the server-side cost is ignored. Thus, the energy consumption of tasks in fog computing mode is as follows:

$$E_{nm}^{F} = E_{nm}^{up}. \tag{9}$$

2.3.3. Cloud Computing Mode. Similarly, the variable $Xc/nm \in \{0, 1\}$ is adopted to represent the decision of tasks in the cloud. When $Xc/nm = 1$, the task will be executed in the cloud. If the task will be processed in other servers, then $Xc/nm = 0$.

Without loss of generality, assume that the cloud server has nearly unlimited computation resources and can process multiple tasks in parallel. As the cloud server is located at the top of the network, which is far away from the local side, the delay of cloud computing is mainly affected by the propagation delay. The propagation delay T^{fix} is generally fixed and can be expressed as a constant. Accordingly, the total delay Tc/nm of the cloud mode is shown as follows:

$$T_{nm}^{C} = T_{nm}^{up} + T^{fix}, \tag{10}$$

and the energy consumption in this mode is as follows:

$$E_{nm}^{C} = E_{nm}^{up}. \tag{11}$$

2.4. Problem Formulation. According to the above-mentioned offloading models, the expressions for the total delay and energy consumption of offloading can be concluded, respectively, as follows:

$$\begin{aligned} T &= \sum_{n=1}^{N} \sum_{m=1}^{M} \left(X_{nm}^{L} T_{nm}^{L} + X_{nm}^{F} T_{nm}^{F} + X_{nm}^{C} T_{nm}^{C} \right), \\ E &= \sum_{n=1}^{N} \sum_{m=1}^{M} \left(X_{nm}^{L} E_{nm}^{L} + X_{nm}^{F} E_{nm}^{F} + X_{nm}^{C} E_{nm}^{C} \right). \end{aligned} \tag{12}$$

To minimize the delay and energy consumption for all UEs, we introduce a cost function Λ modified as the weighted sum of delay and energy as follows:

$$\Lambda(\mathrm{O}, \mathrm{d}, \mathrm{X}, \mathrm{b}, \varphi) = \mu T + (1 - \mu) E, \tag{13}$$

where $\mathrm{O} = \{O_{nm} | n \in \mathrm{N}, m \in \mathrm{M}\}$, $\mathrm{b} = \{b_n(t) | n \in \mathrm{N}\}$, $\mathrm{d} = \{d_{nk}(t) | n \in \mathrm{N}, k \in \mathrm{K}\}$, $\varphi = \{\varphi_n(t) | n \in \mathrm{N}\}$ and $\mathrm{X} = \{XL/nm, XF/nm, XC/nm | n \in \mathrm{N}, m \in \mathrm{M}\}$. $\mu \in [0, 1]$ is weighting factor to represent the focus ratio of delay to energy.

For each input O and d, we are interested in minimizing the cost function to obtain the desired suboptimal decision ($\{\mathrm{X}, \mathrm{b}, \varphi\}$) as follows:

$$\Lambda^{*}(\mathrm{O}, \mathrm{d}) = \underset{\mathrm{X}, \mathrm{b}, \varphi}{\text{minimize}}\, \Lambda(\mathrm{O}, \mathrm{d}, \mathrm{X}, \mathrm{b}, \varphi), \tag{14a}$$

$$\textit{subject to} \quad X_{nm}^{L} + X_{nm}^{F} + X_{nm}^{C} = 1, \tag{14b}$$

$$0 \le \sum_{n=1}^{N} b_n(t) \le 1, \tag{14c}$$

$$0 \le \sum_{n=1}^{N} \varphi_n(t) \le 1. \tag{14d}$$

Equation (14b) restricts the task to be computed in only one of the local, fog, or cloud modes at time t. Equation (14c) represents that the sum of the allocated upload channel resources cannot be more than the total channel resources. Equation (14d) means that the sum of allocated fog computation resources cannot exceed the total in the F-RAN.

Specifically, there are binary variables along with continuous variables of $[0, 1]$ and nonlinear terms of multiplication of unknown variables in equation (14). Thus, the minimization cost function problem can be attributed to the mixed integer nonlinear programming (MINLP) problem, which is a nonconvex problem with a difficult solution.

In the next section, we will transform this problem into a tractable convex problem and propose a deep Q-learning-

based algorithm to solve it. Additionally, the meaning of symbols in Section 2 is shown in Table 1.

3. Offloading Solution

In this section, to address the MINLP problem, a model-free offloading algorithm based on a double-deep Q network, DDQN, is proposed that enables offloading decision-making and the allocation of resources in the F-RAN.

It is assumed that at the beginning of time t, the edge router will collect the environmental information of $\mathbf{N}$ devices to get the state $\mathbf{s_t}$ of the F-RAN, such as the following:

$$\mathbf{s_t} = \{\mathbf{D_{nm}}, \mathbf{C_{nm}}, \mathbf{d_n}(\mathbf{t}), \mathbf{B^r}, \mathbf{F^r}, \mathbf{U^n}\}, \tag{15}$$

where $\mathbf{F^r}$ is the remained fog computation resources and $\mathbf{B^r}$ is the remained channel bandwidth in the F-RAN. $\mathbf{U^n}$ denotes the number of unprocessed tasks for UE $\mathbf{n}$.

After that, the DDQN algorithm will generate numerous possible actions based on $\mathbf{s_t}$. Once an action is implemented, the algorithm will feed a reward based on the current state and the taken action. According to equation (13), the reward $\mathbf{r_t}$ for time $\mathbf{t}$ is defined as follows:

$$r_t = -[\mu T + (\mathbf{1} - \mu)E]. \tag{16}$$

3.1. Design of Mapping-Based Action Vector. In equation (15), there are three constraint variables that determine the loss function, namely, the offloading decision $\mathbf{X}$, the allocated fog computation resource $\mathbf{b}$, and the allocated uplink channel resource φ, respectively. Hence, at time $\mathbf{t}$, let the edge router output the following decision:

(1) In the offloading location of UE $\mathbf{n}$'s $\mathbf{m}$-th task, $\mathbf{w_{nm}} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. $\mathbf{w_{nm}} = \mathbf{1}$ represents the UE $\mathbf{n}$'s $\mathbf{m}$-th task that will be processed locally. Moreover, $\mathbf{w_{nm}} = \mathbf{2}$ indicates that this task will be offloaded to the fog server, while $\mathbf{w_{nm}} = \mathbf{3}$ means it will be processed in the cloud server.

(2) The channel resources are assigned to UE $\mathbf{n}$: $\mathbf{b_n}(\mathbf{t}) \in \{\mathbf{0}, \mathbf{0.2}, \mathbf{0.4}, \ldots, \mathbf{4}\} \times \mathbf{B/N}$. As Q-learning does not apply to the continuous action space, the resources cannot be allocated in a continuous percentage manner, e.g., the proportion of channel resources and the fog computation resources allocated. Hence, intending to improve the sample quality and speed up the convergence, this article proposes using the average allocation as the benchmark. It indicates that the channel resources allocated to a UE are distributed between 0 and 4 times the average channel resources at intervals of 0.2.

(3) The allocated fog computation resources for UE $\mathbf{n}$: $\varphi_{\mathbf{n}}(\mathbf{t}) \in \{\mathbf{0}, \mathbf{0.2}, \mathbf{0.4}, \ldots, \mathbf{4}\} \times \mathbf{F/N}$ are given.

Furthermore, we refer to $\mathbf{w_{nm}}$, $\mathbf{b_n}(\mathbf{t})$, and $\varphi_{\mathbf{n}}(\mathbf{t})$ as the offloading subactions, which are denoted as $\mathbf{a_t} = \{\mathbf{w_{nm}}, \mathbf{b_n}(\mathbf{t}), \varphi_{\mathbf{n}}(\mathbf{t})\}$. $\mathbf{a_t}$ represents the decision action of the UE at time $\mathbf{t}$. The action space $\mathbf{A}$ contains the set of all decision actions that may be output. Since the number of possible actions for output is {3, 21, 21}, we can calculate the size of the total action space $\mathbf{A}$ to be 1323.

For the DDQN algorithm, we use the single-intelligence approach to output actions. First, the edge router collects the current state s_t. Then, the DDQN algorithm outputs the action with the maximum Q-value in the action space A. Finally, the DDQN algorithm maps the output actions into the corresponding offloading subactions to get the executable decision. Hence, we construct a one-to-one mapping relationship with the iterative approach, as shown in Algorithm 1.

3.2. DDQN Algorithm. The structure of the proposed DDQN algorithm is shown in Figure 3. It consists of two networks, namely, the main neural network and the target neural network. The DDQN uses the main network to generate the action space with the largest Q-value. The target network is for updating the main network and evaluating the next action to confirm whether the generated action space is the suboptimal solution or not.

3.2.1. Main Neural Network. As a first step, we construct a main neural network with policy π as the decision criterion, whose network parameter is θ. Assume that if the DDQN algorithm generates a decision action $\mathbf{a_t}$ at time $\mathbf{t}$ with the policy π, noted as π: $\theta_{\mathbf{t}} \longrightarrow \mathbf{a_t}$, it will still adopt the same policy to generate the decision action later. We note the expected value of the reward $\mathbf{Q}^{\pi}$, which is obtained by the algorithm after completing a trajectory τ with policy π, as the Q-value of the algorithm. The Q-value is given as follows:

$$\mathbf{Q}^{\pi}(\mathbf{s_t}, \mathbf{a_t}; \boldsymbol{\theta}_{\mathbf{t}}) = \mathbf{E}^{\pi}[\mathbf{G_t} | \mathbf{A_t} = \mathbf{a}, \mathbf{S_t} = \mathbf{s}], \tag{17}$$

where $\theta_{\mathbf{t}}$ is the parameter of the main network at time $\mathbf{t}$.

Once the Q-value is calculated, the main network will record the Q-values corresponding to all selectable actions. Then, the main network selects the action space, which corresponds to the maximum Q-value, as the current output action.

$$\mathbf{a_t} = \underset{\mathbf{a}}{\operatorname{argmin}}\, \mathbf{Q}(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}). \tag{18}$$

However, if the action selection is only based on equation (18), it will lead to the conclusion that in the iterative computation, the DDQN algorithm always follows the same policy for decision-making. Thus, the output will still be the same decision action. The policy π cannot be efficiently updated by the main network as well.

To avoid only following the same strategy π while outputting the same actions, we import an ε-greedy method to extend the exploration of actions as follows:

$$\mathbf{a_t} = \begin{cases} \underset{\mathbf{a}}{\operatorname{argmin}}\, \mathbf{Q}(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}), & \mathbf{P} = \boldsymbol{\varepsilon}, \\ \mathbf{Rando\,mAction}, & \mathbf{P} = \mathbf{1} - \boldsymbol{\varepsilon}. \end{cases} \tag{19}$$

where $\mathbf{P}$ represents the probability of adopting current action selection methods. Equation (19) states the main

Table 1: The meaning of symbols in Section 2.

Symbol	Definition
N	The number of UEs
M	The number of tasks generated for each UE
K	The number of F-APs
O_{nm}	The data size of UE n's m-th task
D_{nm}	The size of the offloading data for UE n's m-th task
C_{nm}	The size of the required computation for UE n's m-th task
$h_{nk}(t)$	The wireless channel gain between the k-th F-AP and n-th UE at time t
$d_{nk}(t)$	The linear distance from the k-th F-AP to n-th UE at time t
$r_{nk}(t)$	The band utilization between the k-th F-AP and n-th UE at time t
$b_n(t)$	The proportion of channel resources allocated to the n-th UE at time t
A_d	The antenna gain
f_c	The carrier frequency
d_e	The path loss index
P_n	The transmit power of device n
σ^2	Power of the white Gaussian noise
ε^{up}	The energy consumption to upload 1 KB of data
ε^{local}	The amount of energy consumed by the CPU per cycle
t_1	The time cost of uploading the relevant information
t_2	The transmission time of tasks during offloading
t_3	The backhaul delay of the result
Ttr/nm	The total transmission delay for UE n's m-th task
Tup/nm	The transmission delay of UE n's m-th uploading task
TL/nm	The time consumption of UE n's m-th task in local side
TF/nm	The time consumption of UE n's m-th task in fog side
TC/nm	The time consumption of UE n's m-th task in cloud side
Eup/nm	The energy consumption of UE n's m-th uploading task
EL/nm	The energy consumption of UE n's m-th task in local side
EF/nm	The energy consumption of UE n's m-th task in fog side
EC/nm	The energy consumption of UE n's m-th task in cloud side
XL/nm	Offloading decision of UE n's m-th task in local side $XL/nm = \begin{cases} 0 & \text{offloading} \\ 1 & \text{local computing} \end{cases}$
XF/nm	Offloading decision of UE n's m-th task in fog side $XF/nm = \begin{cases} 0 & \text{offloading} \\ 1 & \text{fog computing} \end{cases}$
XC/nm	Offloading decision of UE n's m-th task in cloud side $XC/nm = \begin{cases} 0 & \text{offloading} \\ 1 & \text{cloud comupting} \end{cases}$
λ_n	The computational capacity of UE n
F	The computation resources managed by the edge router
B	The total upload channel resources
f_K	The computation resources of the K-th F-AP
$\varphi_n(t)$	The proportion of fog computation resources allocated to the n-th UE at time t
$\mathbf{T^{fix}}$	The propagation delay to the cloud server
$\Lambda(\cdot)$	The weighted sum cost function
$\mathbf{T}$	The time consumption of computing all tasks
$\mathbf{E}$	The energy consumption of computing all tasks
μ	The focus ratio of delay to energy

network will select the action with the highest Q-value as $\mathbf{a_t}$ with probability ε or random action with probability $\mathbf{1} - \varepsilon$.

3.2.2. Target Neural Network. In the deep Q algorithm, if only the main network evaluates the Q-value, it will result in an overestimation after several iterations. Here, a target neural network is additionally added, which has precisely the same structure as the main network but with different parameters. The action space is generated by the main network. Additionally, the target network is responsible for making corrections to the main network while selecting the least costly action for output.

Specifically, if the main network determines a new decision action at the t-th time, it will first pass the action to the target network. Then, the target network will evaluate the Q-value of that action according to a specific prediction function. Meanwhile, the Q-value will be substituted into the loss function to determine whether the main network should be updated or not. The prediction function is shown as follows:

$$y_t = r_t + \gamma Q\left(s_{t+1}, \left(a_t; \theta_t\right); \theta_t^-\right), \tag{20}$$

where y_t is the Q-value calculated by the target network based on the current action, θ_t^- is the parameter of the target network. $(a_t; \theta_t)$ denotes the action of the main network, which is selected according to the ε-greedy method in the context of the parameter θ_t.

Separately, in this article, the mean squared deviation function is used as the loss function to update the parameters θ_t of the main network. The loss function is as follows:

$$L\left(\theta_t\right) = E\left[\left(y_t - Q\left(s_t, a_t; \theta_t^-\right)\right)^2\right], \tag{21}$$

where $Q(s_t, a_t; \theta_t^-)$ is the Q-value of the target network output under the old parameter θ_t^-.

After generating all possible action spaces, the target network will select the action with the lowest cost Λ for output. Furthermore, the action outputs by the target network are recorded as the suboptimal decision $a*/t$ in the state at time t.

$$a_t^* = \underset{\mathrm{a}}{\operatorname{argmin}}\, \Lambda^*\left(\mathrm{O}, \mathrm{d}, a_t\right), \forall a_t \in \mathrm{a}. \tag{22}$$

3.2.3. Network Improvement and Training. About updating network parameters, this article uses the empirical replay method. The suboptimal decision obtained in equation (22) will be for updating the offloading policy of the main network. Specifically, after outputting each suboptimal decision by the target network, the DDQN algorithm will store a set of samples, noting as DATA $= \{s_t, a_t, r_t, s_{t+1}\}$, into a finite storage space. This storage space is called the public experience pool, from which the main network randomly selects DATA to update its parameters θ. Meanwhile, if the public experience pool is full, the oldest DATA will be replaced with the new one.

As an example, suppose that in the DDQN algorithm interacting with the environment, one trajectory τ is able to generate 100 sets of transfer samples. Meanwhile, assuming that the size of the public experience pool is 500, it will be filled up after 5 complete trajectories. When the pool is filled, the algorithm randomly draws a certain batch of DATA from it and gives them to the network for learning. Through learning, the parameters of the main network are updated, which results in improving its policy π. Besides, the updated policy continues to interact with the environment and generate new DATA. Accordingly, the DDQN replaces the

(1) **Initialization:** Let a = 0;
(2) **for** $i = 1$ **to** *len* (w_{nm}) **do**
(3) **for** $j = 1$ **to** *len* $(b_n(t))$ **do**
(4) **for** $k = 1$ **to** *len* $(\varphi_n(t))$ **do**
(5) Mapping [a] = [ij, k];
(6) a+ = 1;
(7) **end for**
(8) **end for**
(9) **end for**

Algorithm 1: The mapping relationship between a_t and offloading subaction.

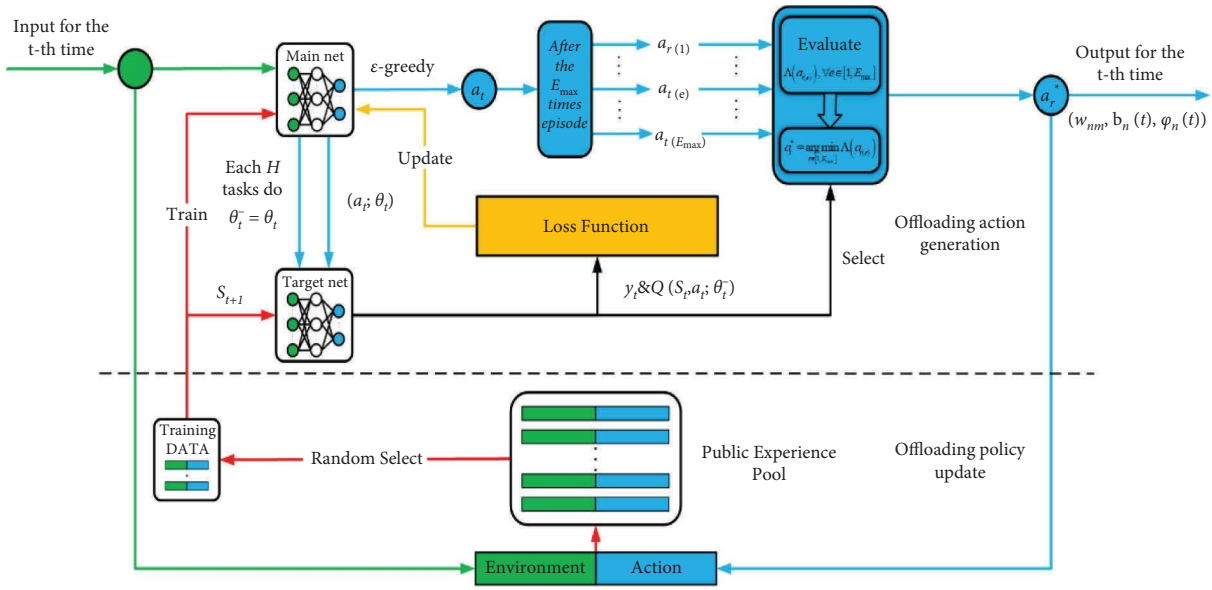


Figure 3: The schematics of the proposed DDQN algorithm.

oldest DATA with the new DATA and repeats this step over and over again.

In the next section, the performance and accuracy of the proposed DDQN algorithm are evaluated based on numerous simulations. Furthermore, the pseudocode of the DDQN is shown in Algorithm 2.

4. Simulation and Evaluation

In the simulation, we construe a F-RAN with 30 UEs and 5 F-APs, where each UE has 100 unrelated and detachable tasks to compute. The range of values for uploading data D_{nm} is [150 KB, 1024 KB], and the amount of computation needed to process locally C_{nm} is in [100 MHz, 500 MHz]. Moreover, all-local devices are assumed to have the same processing power of 1 GHz. Besides, the distance $d_{nk}(t)$ from a UE to a F-AP is evenly distributed within [20 m, 200 m]. Other parameters are set in Table 2.

We adopt the deep Q network model of a fully connected network with 4 hidden layers, where each layer contains 80 neurons. The hidden layer uses Relu as the activation function. The learning rate is 0.001, and the size of the public experience pool is 512. During each training, the algorithm will randomly pick up 32 DATAs. Besides, we consider 500 episodes, i.e., $E_{\max} = 500$, where each episode has 1000 DATAs, i.e., $t_{\max} = 1000$.

4.1. Convergence and Performance Evaluation. The convergence of the DDQN algorithm was evaluated under different algorithm settings. The simulation results are shown in Figure 4. In the subplots, the x-axis represents each episode, and the y-axis shows the average cost of offloading for each episode.

The convergence performance of the DDQN algorithm under different public experience pool sizes is shown in Figure 4(b), where the size of the pool is noted as memory. Lacking sufficient DATAs, the converged cost of the algorithm is pretty high with small memory (e.g., 256). As the memory gradually increases (from 512 to 4096), the average cost of offloading is kept at a low state. However, the larger memory corresponds to a slower convergence speed. Hence, in the next simulation, we adopt a memory size of 512.

In Figure 4(c), we investigate the convergence performance under different batch sizes, i.e., the number of DATAs sampled in each training round. As the batch size increases from 4 to 32, the algorithm converges at a significantly faster speed. As it further increases from 32 to 128, the performance does not improve significantly in terms of

Input: Number of UEs N, size of tasks O and distance between UE and F-AP d;
Output: Suboptimal decision a_t^*;
(1) **Initialization:** Initialize the parameter of the main network with random weight θ and the parameter of the target network with random weight θ^- and empty the public experience pool;
(2) Set training interval H;
(3) **for** episode = 1 **to** $E_{\max}$ **do**
(4) Reset starting environment information s_1;
(5) **for** $t = 1$ **to** $t_{\max}$ **do**
(6) Reset remaining channel resources B^r and remaining fog computation resources F^r;
(7) **for** $n = 1$ **to** N **do**
(8) The main network generates action a_t with the ε-greedy method according to s_t;
(9) Map a_t to subactions $\{w_{nm}, b_n(t), \varphi_n(t)\}$ and implement them in the environment;
(10) Obtain status s_{t+1} and reward r_t based on the environmental changing (a_t);
(11) Mark DATA = $\{s_t, a_t, r_t, s_{t+1}\}$ and store it in the public experience pool;
(12) **if** (*the public experience pool is full*) **then**
(13) The target network calculates $y_t = r_t + \gamma Q(s_{t+1}, (a_t; \theta_t); \theta_t^-)$;
(14) Update the main network parameter θ_t based on $L(\theta_t) = \mathrm{E}[(y_t - Q(s_t, a_t; \theta_t^-))^2]$ and replace the oldest data with the new one;
(15) **end if**
(16) **while** (episode **mod** $H = 0$) **do**
(17) Assign the main network parameters to the target network, i.e., $\theta_t^- = \theta_t$;
(18) **end while**
(19) **end for**
(20) **end for**
(21) **end for**
(22) The target network picks the action with the smallest $\Lambda(\mathrm{O}, \mathrm{d}, a_t)$ as the suboptimal decision $a*/t$: $a*/t = \underset{a}{\operatorname{argmin}} \Lambda^*(\mathrm{O}, \mathrm{d}, a_t), a = \{a_1, a_2, ..., a_{t_{\max}}\}$.

Algorithm 2: The DDQN algorithm.

Table 2: Simulation parameters involved in this article.

Parameters	Values	Parameters	Values
P_n	0.5w	σ^2	$10^{-3}\,w$ [38]
B	20 MHz	F	30 GHz
T^{fix}	1s	f_c	915 MHz [38]
H	32	μ	0.5
A_d	4.11 [38]	d_e	2.8 [38]
ε^{loacl}	$1.37J/Gigacycles$	ε^{up}	$1.39 \times 10^4/KB$

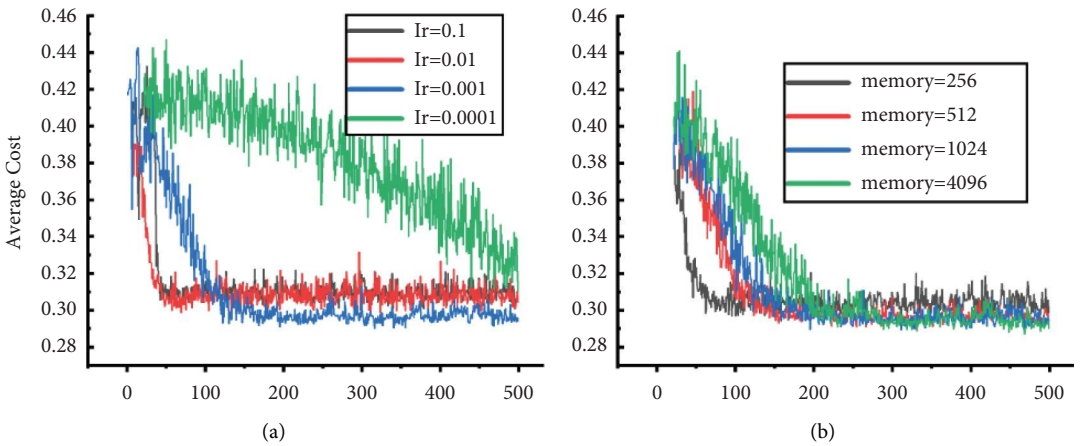


Figure 4: Continued.

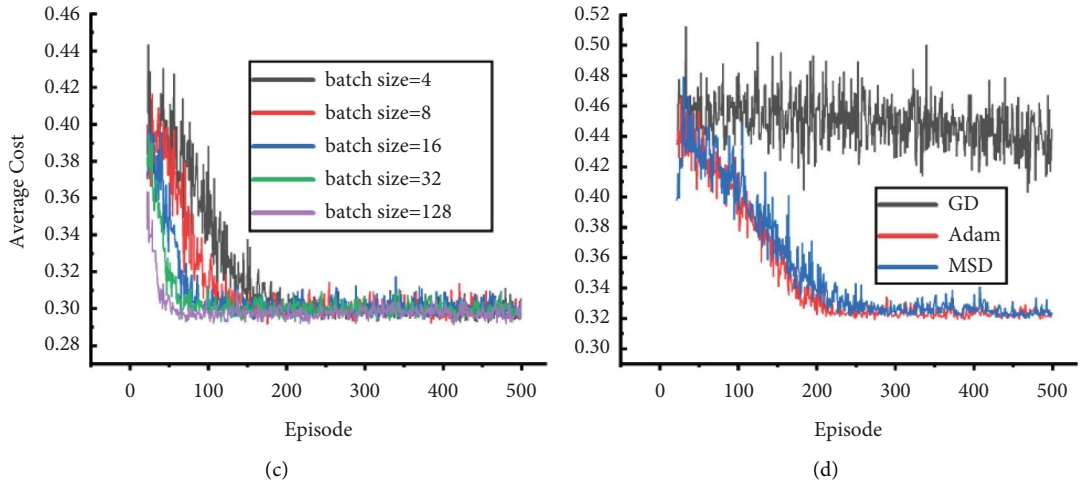


Figure 4: Convergence of the proposed DDQN algorithm under different criteria as follows: (a) learning rate; (b) public experience pool size; (c) batch size; (d) loss function.

convergence speed and cost. Furthermore, a larger batch size means more training time is required. Thus, in this article, we can choose a suitable batch size that not only reduces the training time of one round but also does not significantly decrease the performance of the DDQN algorithm, such as batch size = 32.

Figure 4(d) shows the convergence performance under different loss functions, including gradient drop (GD), mean square deviation (MSD), and adaptive moment estimation (Adam). Just as shown in Figure 4(d), the performance of the CD function is poor, which may not be suitable for the DDQN algorithm. The MSD and Adam functions lead to similar convergence speed and cost. From the above simulation results, in Figure 4, our proposed DDQN algorithm exhibits a stable convergence performance under different parameter settings.

Figure 5 shows the impact on the offloading strategy of the DDQN under different numbers of UEs. When the number of UEs is small, the tasks are mainly offloaded to the fog server. As the number increases, the percentage of the fog server gradually decreases while the percentage of local execution increases. Meanwhile, the cloud is only involved in a small amount of computation in this process. Under the conditions of Table 2 (the main influencing parameter is the computational volumes of tasks), the cost of computing locally is lower than offloading to the cloud server regardless of the number of UEs (as shown in Figure 6). Hence, when fog computation resources are insufficient, the overloaded tasks will be processed locally without the option of offloading to the cloud.

In Figure 7, we compare the impact of different computational volumes on the offloading strategy of the DDQN algorithm. With an increase in the computation volume, the proportion of tasks computed locally is decreasing dramatically, while the proportion on the cloud is increasing rapidly. When the computational volume is small, the DDQN algorithm mainly allocates tasks to be computed locally or in the fog. As the computational volume grows, the computation resources at the fog and the local are insufficient to support the current demand. Thus, the DDQN algorithm offloads more tasks to the cloud server, where computation resources are abundant for processing. It illustrates that the proposed algorithm can be applied in scenarios with different computational requirements.

Figure 8 studies the effect of different weight values μ on the offloading strategy. When $\mu = 0$, it means that we only care about the offloaded energy consumption. In this situation, most tasks are offloaded to the fog or cloud server, with the energy costs being lower. When the weight is increased to 0.1, we can observe that the proportion of cloud servers decreases significantly, while the proportion of the fog side increases. With the introduction of time utility, for the current computation volumes (as shown in Table 2), the task takes much less time to compute in the fog than the propagation delay T^{fix} for the cloud. Moreover, the time cost in the local computing is smaller than the time cost in the fog with current settings, but the local energy consumption is much higher than the conditions of the fog. That is why the processing percentage of the fog server decreases with increasing μ while the local has been rising. Besides, Figure 8 shows that the DDQN algorithm can be well applied to scenarios with different foci.

In Figure 9, we further study the average computation time of the DDQN algorithm under different numbers of UEs. For DDQN employed with a different number of UEs, the time cost is almost the same for each offloading task, which stays at 0.27s. Since the number of UEs does not affect the time cost of the algorithm, it can be applied in massive user offloading scenarios, such as unmanned factories.

4.2. System Utility Comparison. Regarding the practical performance of the system, our DDQN algorithm is compared with five representative benchmarks as follows:

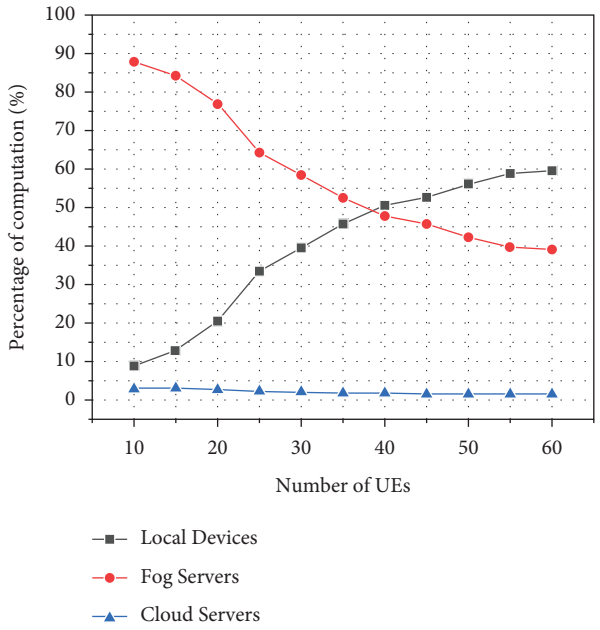


Figure 5: The impact on the offloading strategy of the DDQN algorithm under different number of UEs.

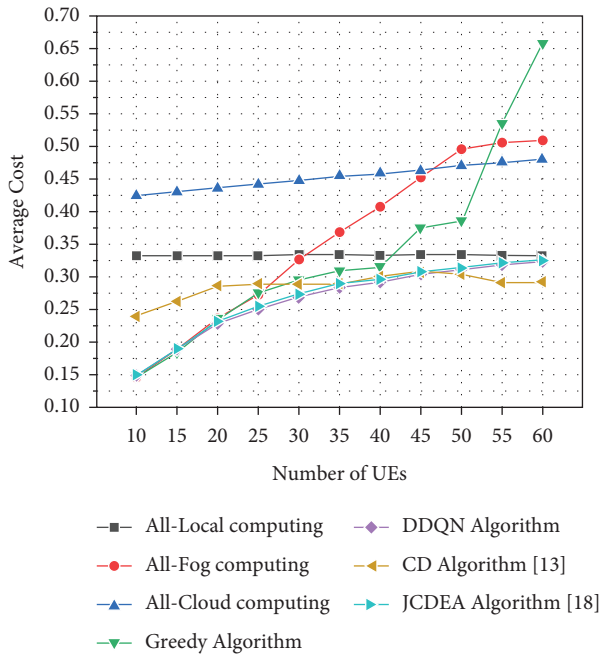


Figure 6: The impact of different number of UEs on the average cost of offloading.

(i) The coordinate descent (CD) algorithm [13] iteratively swaps the offloading patterns of the UE, resulting in minimal delay and energy cost in each round. The iterations will stop when the offloading mode swapping does not further improve the system performance. Moreover, the CD algorithm is proven to achieve near-optimal decisions at different N.

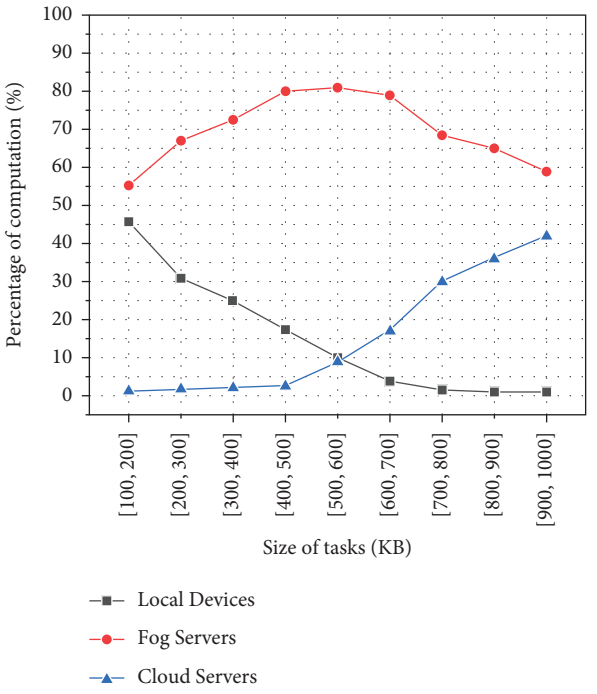


Figure 7: The impact on the offloading strategy of DDQN algorithm under different computational volumes.

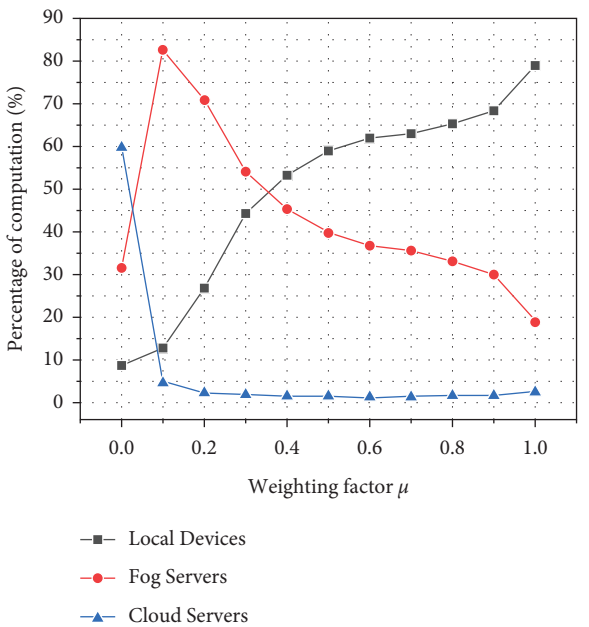


Figure 8: The impact on the offloading strategy of DDQN algorithm under different weight values μ.

(ii) The Joint Computation offloading, Data compression, Energy harvesting, and Application scenarios (JCDEA) algorithm [18] is a comprehensive offloading algorithm that solves the joint computation offloading, data compression, energy harvesting, and application scenario optimization problems in the F-RAN. The JCDEA algorithm obtains the

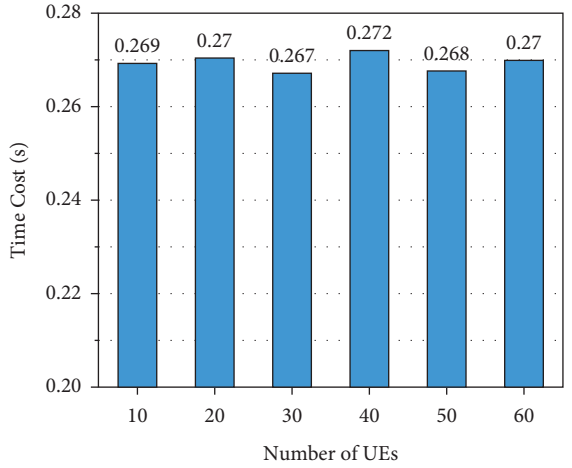


Figure 9: Average computation time of the DDQN algorithm under different number of UEs.

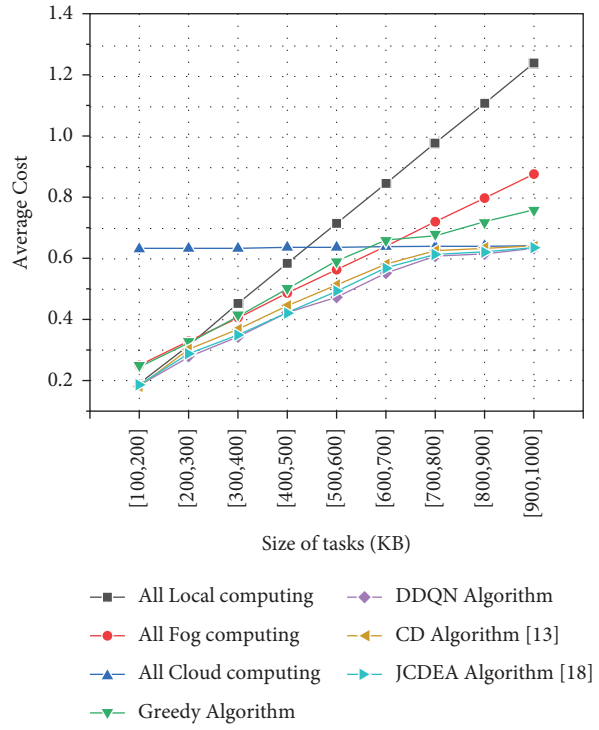


Figure 10: The impact of different computational volumes on the average cost of offloading.

optimal offloading decision and resource allocation policy by transforming the problem of finding an offload policy into solving the minimum cost of local, fog, and cloud computing. We assume that the data compression ratio is 1, i.e., the task is not compressed when introducing this benchmark algorithm. The energy harvesting efficiency is 0, i.e., the local device does not collect energy from the outside.

(iii) The greedy algorithm prioritizes all tasks to a specific F-AP and invokes all of the computation resources of the current fog node. If the fog node has reached its maximum processing capacity, the priority is randomly assigned to the next empty F-AP. If all the fog computation resources are overloaded, the task will be processed locally or in the cloud, depending on which mode has the lower average cost.

(iv) All-local computing tasks are processed on the local device.

(v) All-fog computing tasks are randomly offloaded to the fog server in the F-RAN for processing.

(vi) All-cloud computing, the remote cloud server, processes all user's tasks.

As shown in Figure 6, while the number of UEs changes, the average cost of all-local computing remains stable.

However, for the other benchmarks, the average cost of offloading increases as the user grows. As for the increasing number, the problem of bandwidth resource competition for uplink channels arises, which leads to a decrease in the upload rate allocated to tasks that increases the time cost of offloading. Moreover, when the limited fog computation resources are insufficient to support numerous tasks, additional queuing delays are forced to suffer, which further adds to the extra offloading delay. Hence, the algorithm with a single offloading mode, for the offloading of multiuser scenarios, is not suitable.

For the CD algorithm, the change in the number of UEs has little influence on the average cost, and even the cost in the multi-UE state ($N = 60$) is lower than in the few-UE state ($N = 10$). Thus, the CD algorithm is more inclined to be applied in multiuser scenarios. For the greedy algorithm, we conclude that with the increasing of UEs, it is no longer effective in arriving at the optimal offloading decision. The JCDEA algorithm outperforms other benchmark algorithms but the average cost consumption is always higher than that of the DDQN algorithm. In summary, the DDQN algorithm offers a lower offloading cost scheme with better performance, compared with the benchmark algorithms.

Figure 10 investigates the impact of different computational volumes on the average cost. For the cloud server with abundant computing resources, the main time cost is determined by the propagation delay, which does not vary with the computation volume. This is why the average cost of all algorithms, except the all-cloud computing model, rises linearly with the amount of computation. The performance of the greedy algorithm is weaker in comparison, especially when the fog servers are overloaded. The DDQN, CD, and JCDEA algorithms increase their average cost relatively slowly as the amount of computation adds, eventually converging to the cost of all-cloud computing. However, the proposed DDQN algorithm consistently maintains a lower average cost than other benchmark algorithms. In contrast, the task offloading will have better performance, which always maintains a lower cost in the different focused offloading scenarios by employing our DDQN algorithm.

5. Conclusions

In this work, a novel model-free offloading algorithm, the DDQN, is proposed for the offloading scenario of detachable tasks in the F-RAN, which is based on the double-deep Q network, to allocate the offloading decision of the task, the uplink channel bandwidth, and the fog computation resources that arrive at a minimized cost. By importing binary variables in the offloading strategy, we transform the offloading into a problem of finding binary offloading decisions with resource allocation. Next, we design the delay-energy weighted sum metric and further convert the above problem into a mixed integer nonlinear programming (MINLP) problem based on delay and energy, i.e., obtaining suitable offloading decisions and resource allocation strategies that minimize the cost. Furthermore, a delay-energy weighted sum metric is designed for evaluating the offloading strategy. Moreover, we further convert the above problem into a mixed integer nonlinear programming (MINLP) problem based on the delay and energy weighted sum, i.e., to obtain a reasonable and efficient offloading decision and resource allocation strategy to minimize the offloading cost. Since the MINLP problem is intractable to resolve in a general way, the DDQN algorithm is proposed to generate decisions. Meanwhile, we innovatively combine the action space mapping method with deep reinforcement learning. Numerical simulations illustrate that the DDQN algorithm, compared to the benchmark algorithm, can significantly reduce the offloading cost of task execution.

Finally, it is hoped that the proposed DDQN offloading framework can be expanded in future F-RANs, such as smart IoTs and driverless cars, to optimize real-time offloading for various scenarios with multiple users.

Data Availability

The underlying data supporting the results of this study can be found at the official website of Beijing Natural Science Foundation.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by Beijing Natural Science Foundation, Haidian Original Innovation Joint Fund Project (No. L182039).

References

[1] V. Sharma, I. You, K. Andersson, F. Palmieri, M. H. Rehmani, and J. Lim, "Security, privacy and trust for smart mobile-internet of things (m-iot): a survey," *IEEE Access*, vol. 8, pp. 167123–167163, 2020.

[2] J. Jijin, B.-C. Seet, and P. H. J. Chong, "Performance analysis of opportunistic fog based radio access networks," *IEEE Access*, vol. 8, pp. 225191–225200, 2020.

[3] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A Survey on Cloudlets, mobile Edge, and Fog Computing," in *Proceedings of the 2021 8th IEEE International SCloud*, pp. 139–142, Washington, DC, USA, June 2021.

[4] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management, and evaluation of fog computing systems: a survey," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2494–2516, 2021.

[5] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[6] S. Chen, L. Rui, Z. Gao, W. Li, and X. Qiu, "Cache-assisted collaborative task offloading and resource allocation strategy: a meta reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 9.

[7] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1298–1311, 2021.

[8] Y. Wang, L. Wang, S. Amir, and Q.-G. Wang, "Distributed optimal control for traffic networks with fog computing," *China Commun*, vol. 16, no. 10, pp. 202–213, 2019.

[9] Y. Lan, X. Wang, D. Wang, Z. Liu, and Y. Zhang, "Task caching, offloading, and resource allocation in d2d-aided fog computing networks," *IEEE Access*, vol. 7, Article ID 104876, pp. 876–104891, 2019.

[10] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[11] Y. Ma, H. Wang, J. Xiong, J. Diao, and D. Ma, "Joint allocation on communication and computing resources for fog radio access networks," *IEEE Access*, vol. 8, pp. 108310–108323, 2020.

[12] Y. Jiang, A. Peng, C. Wan et al., "Analysis and optimization of cache-enabled fog radio access networks: successful transmission probability, fractional offloaded traffic and delay," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5219–5231, 2020.

[13] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[14] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[15] M. Mukherjee, S. Kumar, Q. Zhang et al., "Task data offloading and resource allocation in fog computing with multi-task delay guarantee," *IEEE Access*, vol. 7, pp. 152911–152918, 2019.

[16] Q. Tang, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Decentralized computation offloading in iot fog computing system with energy harvesting: a dec-pomdp approach," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4898–4911, 2020.

[17] E. Fakhfakh and S. Hamouda, "Optimised q-learning for wifi offloading in dense cellular networks," *IET Communications*, vol. 11, no. 15, pp. 2380–2385, 2017.

[18] W. Bai, Z. Ma, Y. Han et al., "Joint optimization of computation offloading, data compression, energy harvesting, and application scenarios in fog computing," *IEEE Access*, vol. 9, pp. 45462–45473, 2021.

[19] Y. Wang, P. Lang, D. Tian et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, 2020.

[20] B. Cao, S. Xia, J. Han, and Y. Li, "A distributed game methodology for crowdsensing in uncertain wireless scenario," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 15–28, 2020.

[21] Y. Jie, M. Li, C. Guo, and L. Chen, "Game-theoretic online resource allocation scheme on fog computing for mobile multimedia users," *China Commun*, vol. 16, no. 3, pp. 22–31, 2019.

[22] Z. Shuchen and J. Waqas, "The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment," *Computer Networks*, vol. 178, no. 3, pp. 1389–1286, 2020.

[23] S. Balasubramanian and T. Meyyappan, "Enhancing the computational intelligence of smart fog gateway with boundary-constrained dynamic time warping based imputation and data reduction," in *Proceedings of the 2019 3rd International Conference on Imaging, Signal Processing and Communication (ICISPC, Johor*, Malaysia, July 2019.

[24] B. Jia, H. Hu, Y. Zeng, T. Xu, and Y. Yang, "Double-matching resource allocation strategy in fog computing networks based on cost efficiency," *Journal of Communications and Networks*, vol. 20, no. 3, pp. 237–246, 2018.

[25] Z. Zhao, S. Bu, T. Zhao et al., "On the design of computation offloading in fog radio access networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7136–7149, July 2019.

[26] L. Zhang, B. Cao, Y. Li, M. Peng, and G. Feng, "A multi-stage stochastic programming-based offloading policy for fog enabled IoT-eHealth," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 411–425, Feb. 2021.

[27] B. Liu, C. Liu, M. Peng, Y. Liu, and S. Yan, "Resource allocation for non-orthogonal multiple access-enabled fog radio access networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3867–3878, June 2020.

[28] J. Liu, K. Xiong, D. W. K. Ng, P. Fan, Z. Zhong, and K. B. Letaief, "Maxmin energy balance in wireless-powered hierarchical fog-cloud computing networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7064–7080, 2020.

[29] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for internet of everything," *China Commun*, vol. 16, no. 3, pp. 32–41, 2019.

[30] K. Gu, L. Tang, J. Jiang, and W. Jia, "Resource allocation scheme for community-based fog computing based on reputation mechanism," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 5, pp. 1246–1263, 2020.

[31] C. Yi, S. Huang, and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1076–1091, 2021.

[32] K. Gai, X. Qin, and L. Zhu, "An energy-aware high performance task allocation strategy in heterogeneous fog computing environments," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 626–639, 2021.

[33] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran et al., "A review of fog computing and machine learning: concepts, applications, challenges, and open issues," *IEEE Access*, vol. 7, pp. 153123–153140, 2019.

[34] G. M. S. Rahman, T. Dang, and M. Ahmed, "Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 243–257, 2020.

[35] F. Jiang, R. Ma, C. Sun, and Z. Gu, "Dueling deep Q-network learning based computing offloading scheme for F-ran," in *Proceedings of the 2020 IEEE 31st Annual International Symposium on Personal Indoor and Mobile Radio Communications*, pp. 1–6, London, UK, August 2020.

[36] F. Jiang, X. Zhu, and C. Sun, "Double DQN based computing offloading scheme for fog radio access networks," in *Proceedings of the 2021 2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1131–1136, Xiamen, China, July 2021.

[37] Y. Ouyang, "Task offloading algorithm of vehicle edge computing environment based on Dueling-DQN," *Journal of Physics: Conference Series*, IOP Publishing, vol. 1873, 2021.

[38] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE*

Transactions on Mobile Computing, vol. 19, no. 11, pp. 2581–2593, 2020.

[39] S. Song, Z. Fang, Z. Zhang, C.-L. Chen, and H. Sun, "Semi-online computational offloading by dueling deep-q network for user behavior prediction," *IEEE Access*, vol. 8, pp. 118192–118204, 2020.

[40] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, p. 1, 2020.

[41] E. Baccarelli, M. Scarpiniti, A. Momenzadeh, and S. S. Ahrabi, "Learningin-the-fog (lifo): deep learning meets fog computing for the minimumenergy distributed early-exit of inference in delay-critical iot realms," *IEEE Access*, vol. 9, pp. 25716–25757, 2021.

Hindawi
Mobile Information Systems
Volume 2022, Article ID 7408265, 12 pages
https://doi.org/10.1155/2022/7408265

Research Article

Corrosion Detection Method of Transmission Line Components in Mining Area Based on Multiscale Enhanced Fusion

Jun Wu, Yingxin Sun, and Xinliang Wang

Physics and Electronic Information Engineering, Henan Polytechnic University, Jiaozuo 454000, China

Correspondence should be addressed to Xinliang Wang; wangxl@hpu.edu.cn

Received 5 August 2022; Revised 1 September 2022; Accepted 18 September 2022; Published 4 October 2022

Academic Editor: Yuexia Zhang

Component corrosion is one of the potential safety hazards in transmission lines in mining areas. In order to solve the problem of poor detection accuracy caused by the large proportion of small targets and complex background in the current distant view corrosion inspection task by UAV, we propose a PWR-YOLOV5 detection method for corrosion components based on the YOLOV5 algorithm. Firstly, a new feature fusion network, WA-PANet, is reconstructed on the basis of the path aggregation network (PANet) to make full use of the features at different stages and advance the detection accuracy of small targets in distant view by deepening the process of feature fusion and introducing the skip layer connections and adaptive feature fusion factors. Secondly, the pyramid split attention (PSA) module is introduced into the deep layers of the network to highlight the feature expression of corrosion targets and enhance the ability to detect pixel-level objects. Then, we construct a receptive feature enhancement network (RFENet), which can heighten the feature fusion effect of the WA-PANet and alleviate the problem of the feature expression ability weakening due to the fusion of different receptive field features. Finally, the EIoU Loss is adopted to optimize the loss function and improve the positioning accuracy of the bounding box. The experimental results show that the mAP of the PWR-YOLOV5 algorithm can reach up to 95.37%, which is 5.22% higher than YOLOV5, and the detection speed is 64.9FPS. Compared with the algorithms such as YOLOV4, Faster *R*-CNN, and YOLOX, the improved algorithm has better overall detection performance for the corrosion components of transmission lines in the mining area.

1. Introduction

At present, mineral security has risen to the level of a national strategy [1]. As an important branch of the national power system, the maintenance of the power supply system in mining areas plays a key role in the safety of mineral production work. Transmission lines in mining areas are mostly erected between mountains, rivers, and hills, and the climate is humid. Antivibration hammers, insulators, and other metal components are prone to rust on account of exposure to a harsh environment for long term and may even cause power supply faults such as components falling and line breakage, which will affect the normal operation of the power system in the mining area and seriously threaten the mineral security [2]. Therefore, it is necessary to effectively recognize and detect the corrosion flaws of transmission line components and timely find rust spots and repair corrosion problems, so as to ensure the safe and stable running of the power supply equipment in mining areas.

With Jones in [3] and Araar et al. in [4] introducing small unmanned aerial vehicles (UAV) into the inspection task of transmission lines, the inspection method of "UAV inspection + manual processing" appeared. A UAV was used instead of manual work to collect images, which could greatly reduce front-end labor costs. However, the way of manual detection was greatly influenced by subjective consciousness, resulting in serious missed and false detection [5]. The rapid development of computer technology had realized the combination of computational pattern recognition and power supply system inspection tasks, and then, a new inspection method of "UAV inspection + image processing" came into being [6]. In the task of detecting the corrosion components by using image processing technology, Recky and Leberl [7] used color space combined with a

k-nearest neighbor window to realize the detection of corrosion defects. A detection method for corrosion flaws of antivibration hammers was given in [8] by introducing histogram equalization, median filtering, morphological processing, and the RGB color model. The authors of [9] studied the color and texture features of corrosion images and applied the HSI color model and grey level co-occurrence matrix to identify corroded areas of images. Huang et al. [10], respectively, adopted edge features' enhancement algorithms such as local difference and anisotropic Gaussian kernel directional derivative, and threshold segmentation and morphological processing techniques on grayscale images to obtain the corrosion area ratio and color shadow index of antivibration hammers, which realized the classification of corrosion grade. On the basis of texture features of images, the authors in [11] also combined Fourier transform and multiple color models to distinguish corrosion areas. The methods mentioned above improve the accuracy of corrosion detection to a certain extent, but they are only applicable to detecting corrosion components with a relatively simple background, sparse target distribution, and obvious edge features.

In recent years, machine learning has developed rapidly owing to the successive optimization of calculation capacity and computational overhead. In [12], the authors extracted color, gradient amplitude, and direction histogram, respectively, by letting the aggregation channel feature algorithm to build a multiscale ACF pyramid and also integrated it with the AdaBoost classifier, Graph Cuts algorithm, and RGB color model to judge whether the antivibration hammers were corroded. The algorithm achieved higher detection accuracy, but it requested the way of manual design to extract features, which was complicated in steps and heavy in workload. Keeping up with the boom of object detection technology in deep learning, Petricca et al. in [13] led a convolutional neural network into the field of corrosion detection, providing a new thought for corrosion component recognition. The authors of [14] first used Retinex, an image enhancement algorithm, to decrease the interference of light and shadow on the corrosion color and then employed FPN and RPN structures to redesign the Faster-RCNN to achieve the defects classification and position regression of anti-vibration hammers, which optimized positioning accuracy of the algorithm to a certain degree. Combined with the attention mechanism, a lightweight corrosion targets' detection method for power equipment based on the SSD was presented in [15], realizing the identification of rusted areas with fewer parameters, but the average precision was only 71.35%. In [16], the authors advanced an attention-guided multitask convolutional neural network and connected it with the RPN structure to identify the corrosion degree and abnormal state of power line components. The methods mentioned above lack specific classification, and the shapes and appearances of different components are disparate, resulting in a high detection error rate.

To sum up, there are few studies on the detection of specific corrosion components of transmission lines at present, and the detection effect of related corrosion studies is not satisfactory. So, we put the YOLOV5 [17] to the task of detecting two corrosion components of transmission lines in mining areas, including antivibration hammers and insulators. According to the problems that there are a high proportion of small targets and a great many background interferents all caused by long-range shootings in image acquisition by UAV, we propose four improvements and optimizations as follows:

(i) The weight adaptive path aggregation network (WA-PANet) is constructed. On the basis of the PANet, we deepen the process of feature fusion and inlet skip layer connections and adaptive feature fusion factors to enhance the detection accuracy of different scale objects.

(ii) We introduce the PSA mechanism to fuse context information of different scales and meanwhile generate pixel-level attention for the targets, so as to highlight the feature expression of small corrosion targets.

(iii) The RFENet is built to strengthen the fusion effect of the WA-PANet and alleviate the weakening of the feature expression ability induced by feature fusion at different stages.

(iv) The loss function of bounding box regression adopts the EIoU Loss, which can improve the positioning accuracy and convergence rate of the network.

2. Materials and Methods

2.1. Data Acquisition and Processing. The data of transmission line components in the mining area required for the experiments in this study are provided by the Hemei Group Power Supply Department. We use the Pr software to extract frames from the video shot by the UAV and filter out a large number of similar and background pictures. There are two kinds of image resolutions 5184×3888 pixels and 5472×3078 pixels, respectively. On account the fact that the data given above covers all components of overhead transmission lines, we further screen out a total of 2705 images containing antivibration hammers and insulators and then adopt the LabelImg, a deep learning target annotation tool, to mark the objects. The labels are set to FangRust (corrosion antivibration hammers), Fang_NoRust (noncorrosion antivibration hammers), Jue_Rust (corrosion insulators) and Jue_NoRust (noncorrosion insulators), and the tagging format is VOC format. The data processing part applies the adaptive image scaling method to uniformly scale the read images of different scales to the network input size and accomplishes online data augmentation through random clipping, mosaic data enhancement, etc., which enhances the robustness of the model and improves the detection performance of the algorithm for different scale targets, especially the small ones.

2.2. PWR-YOLOV5 Network. Although the YOLOV5 algorithm shows a good performance in generic object detection tasks, the color of corrosion is easily confused with background items such as dead leaves and dust and is

also greatly affected by the intensity of light. In addition, the shooting distance of the UAV is not flexible to control, leading to many small targets in the distant view, all of which have a poor influence on the recognition of rusted components. In order to adapt to the corrosion targets' detection task in the actual scene, in this study, we first redesign the feature fusion network WA-PANet by deepening the feature fusion process and setting skip layer connections and learnable feature fusion arguments, which not only retains more details but also improves the feature fusion effect. Then, the PSA mechanism is pulled into the deep layers of the network, in which the softmax is used to adaptively fuse the spatial features of different scales and channel attention weights to generate pixel-level attention to the objects. Next, we apply bottleneck structures and dilated convolutions to construct the feature enhancement network RFENet to capture the multiscale features under different receptive fields in order to strengthen the fusion effect of the WA-PANet and enhance the detection accuracy of corroded targets at different scales. Finally, the bounding box regression loss function is optimized by the EIoU Loss, which solves the problem of ambiguous definition of aspect ratio loss in CIOU Loss and advances the positioning accuracy of the network. Based on the innovations mentioned above, the PWR-YOLOV5 detection algorithm for corrosion components is proposed. The structure of the PWR-YOLOV5 is shown in Figure 1.

2.2.1. Weight Adaptive Path Aggregation Network. The YOLOV5 algorithm uses the PANet for feature fusion, and its structure is shown in Figure 2(a). In this module, the method of bidirectional fusion is deployed to integrate deep semantic information and shallow location information, which promotes detection accuracy to a certain extent. However, this fusion method does not distinguish the feature information at different stages, inducing negative feature fusion results. For the purpose of solving the problem and boosting the detection performance of the rusted small targets, on the basis of the PANet, we make the process of feature fusion deeper, and meanwhile, import skip layer connections and adaptive feature fusion factors [18] to establish the weight adaptive path aggregation network. The specific structure of the WA-PANet is shown in Figure 2(b), where P_i represents the ith feature produced in the backbone and F_i and N_i are the intermediate features generated during the fusion procedure.

The WA-PANet feature fusion network consists of two branches. The one top-down branch is used to transmit the powerful semantic information and perform the fusion of deep and shallow features, where the fusion mode of channel cascade is applied to preserve more feature information. In this study, we add an up-sampling operation to form the feature map F_2, which is fused with the feature map P_2 yielded in the backbone to generate the large-scale feature N_2 with rich semantic and location information so as to improve the performance of the network to identify small corroded targets. The other bottom-up branch is employed to transfer detailed information with the purpose of enhancing the ability to locate objects at different scales. For the sake of obtaining refined features, the skip layer connections from input to output are set at nodes N_3, N_4, and N_5, respectively. Furthermore, due to the multiple input features of these three nodes coming from different stages of the network and contributing differently to the fusion results, we also introduce a learnable feature factor for each branch of each node so that the algorithm adaptively learns the importance of different stage features in the training process. The features of each node of the WA-PANet can be expressed as the following equations (1)–(5):

$$F_5 = Conv(P_5), \tag{1}$$

$$F_i = Conv(\text{Concat}(P_i, \text{Upsample}(F_{i+1}))), i = 2, 3, 4, \tag{2}$$

$$N_2 = Conv(F_2), \tag{3}$$

$$N_i = Conv\left(\frac{w_1 \times P_i + w_2 \times F_i + w_3 \times Resize(N_{i-1})}{w_1 + w_2 + w_3 + \varepsilon}\right), i = 3, 4, \tag{4}$$

$$N_5 = Conv\left(\frac{w_1 \times F_5 + w_2 \times Resize(N_4)}{w_1 + w_2 + \varepsilon}\right), \tag{5}$$

where *Conv* refers to a series of convolution operations involved in feature processing, *Upsample* stands for the nearest neighbor interpolation up-sampling, *Concat* means that two features from different stages carry out splicing and fusion on the channel dimension, that is, channel cascade, and *Resize* means to adjust the size of the feature maps in the two dimensions of space and channel. And w_i is a learnable parameter, which is multiplied by the input feature of the corresponding branch at each node, so the larger the value of w_i is, the greater the influence of the branch on the fusion results is, and ε is a constant, much less than 1, which is set to prevent the denominator from being 0.

2.2.2. Pyramid Split Attention Mechanism. There are a large number of interruptions in the actual scene, and the background is complicated. At the same time, with the deepening of the network, the features of small objects with fewer pixels are gradually blurred, and the location information is also less and less obvious. In an effort to effectively suppress the complex background information and overcome the influence of light intensity, the pyramid split attention mechanism [19] is introduced into the deep layers of the network to promote the detection accuracy for corroded components. The implementation of the PSA module is shown in Figure 3.

First of all, we utilize the split and concatenation (SPC) structure to obtain feature information of different scales along the channel direction, and the SPC composition is demonstrated in Figure 4. The module divides the input feature maps X into S parts in the channel dimension, denoted as $[X_1, X_2, \ldots, X_{S-1}]$, so each part has the same number of channels $C' = C/S$, and the feature of the ith

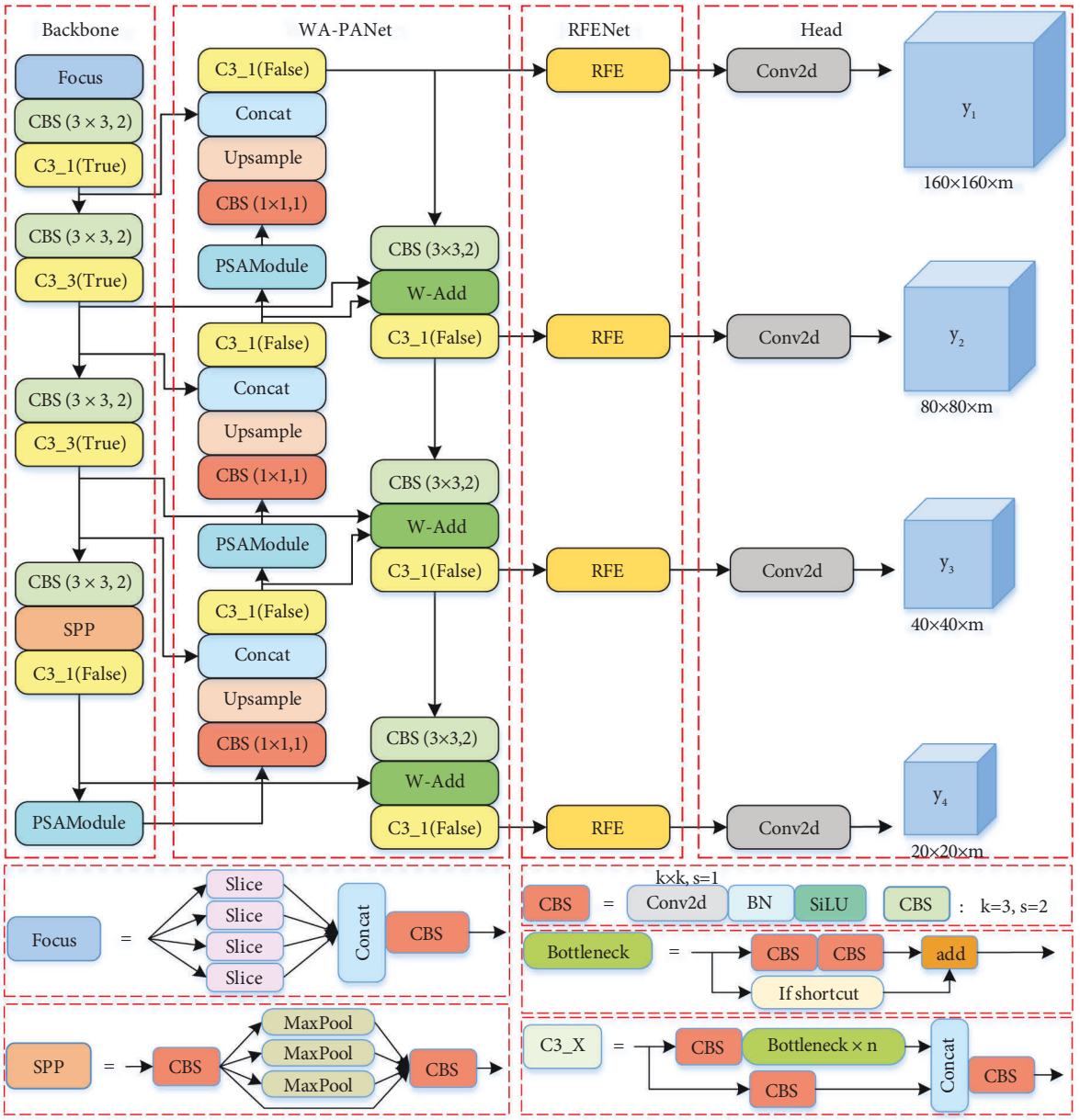


Figure 1: The PWR-YOLOV5 network structure. *P* is the PSA module; *W* presents WA-PANet; and *R* is RFENet.

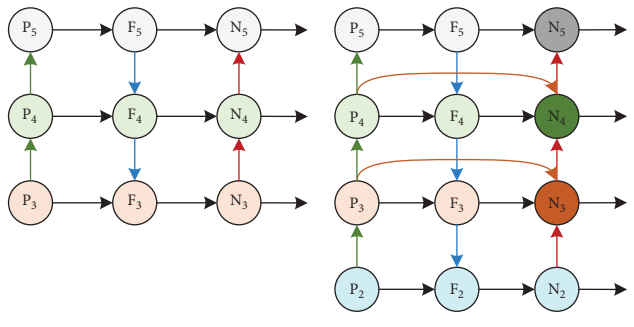


Figure 2: (a) The path aggregation network structure. (b) The weight adaptive path aggregation network structure.

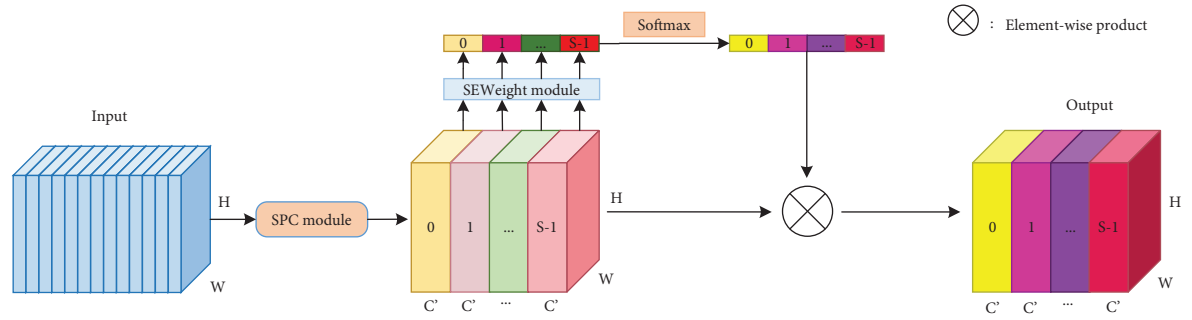


Figure 3: The pyramid split attention structure.

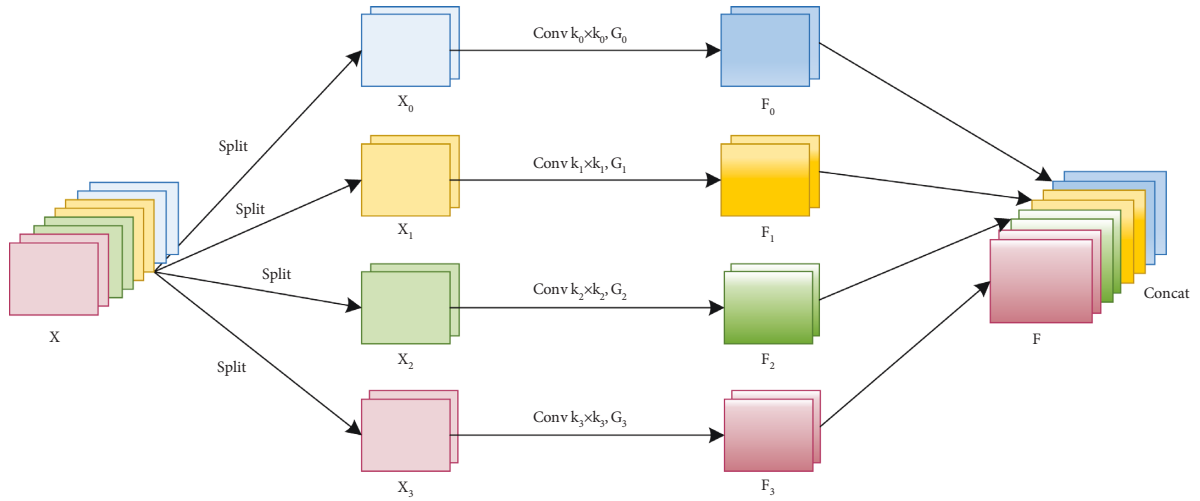


Figure 4: The split and concat structure.

branch can be represented as $X_i \in R^{C' \times H \times W}, i = 0, 1, \ldots, S-1$. In this study, the value of S is 4. After being divided in this way, the input tensors are processed in parallel by using convolution kernels of different scales, so as to extract spatial information from feature maps of each branch and gain features of different receptive fields and depths. However, with the augmentation of the convolution kernel size, the number of parameters also increases significantly. In order to save computational overhead, the group convolution is inlet; moreover, a new rule is also designed for the selection of the grouping number. The relationship between the size of multiscale convolution kernels and the number of groups can be represented as the following formula:

$$G = 2^{(k-1)/2}, \tag{6}$$

where K is the size of the convolution kernel and G is the number of groups. In particular, the number of groups is defaulted to 1 when $K = 3$. The generating process of the multiscale features is given in (7), and then, the different scale features produced by each branch are cascaded in channel to obtain the feature map F in the following formula:

$$F_i = Conv(k_i \times k_i, G_i)(X_i), i = 0, 1, 2, \ldots, S-1, \tag{7}$$

$$F = \text{Concat}([F_0, F_1, \ldots, F_{S-1}]), \tag{8}$$

where $k_i = 2 \times (i+1) + 1$ denotes the size of convolution kernel applied in the ith branch, and the grouping number of convolution operation in the ith branch is $G_i = 2^{(k_i-1)/2}$, and $F_i \in R^{C' \times H \times W}$ signifies the feature of different receptive fields emerged on each branch, and $F \in R^{C \times H \times W}$ is the complete multiscale feature maps acquired through the SPC module.

Secondly, the SEWeight [20] is used to pick up the channel attention information of each branch feature, and the attention weight vector of each branch channel is procured by (9), in which $Z_i \in R^{C' \times 1 \times 1}$ shows the channel attention weights obtained from the different scale features F_i:

$$Z_i = \text{SEWeight}(F_i), i = 0, 1, 2, \ldots, S-1. \tag{9}$$

Then, the soft attention method is employed to handle the channel attention weight vector Z_i to adaptively select the importance of multiscale spatial features across channels. The weight allocation pattern of soft attention is shown in formula (10), where Softmax is adopted to weighting fusion of the spatial and channel information of each branch to obtain the weight att_i, which contains the information of all positions in the space and the attention weight in the channel:

$$att_i = Softmax(Z_i) = \frac{\exp(Z_i)}{\sum_{i=0}^{S-1} \exp(Z_i)}, i = 0, 1, 2, \ldots, S-1. \tag{10}$$

Finally, the fused attention weight att_i of each branch is multiplied by the corresponding scale feature F_i to fetch the feature map Y_i with multiscale pixel-level channel attention in formula (11), and in the end, the multiscale refined output is acquired through the concatenation operation that can maintain the integrity of features, and the course can be denoted as follows:

$$Y_i = F_i \odot att_i, i = 0, 1, 2, \ldots, S-1, \tag{11}$$

$$\text{Out} = \text{Concat}\left(\left[Y_0, Y_1, \ldots, Y_{S-1}\right]\right), \tag{12}$$

where $\odot$ indicates pixel-wise multiplication. The PSA module can integrate multiscale spatial information and cross-channel attention through each split feature group, which can not only achieve the fusion of different scales of context information but also generate pixel-level attention to the targets. In this study, we put the PSA module into the last layer of the backbone and the top-down process of the WA-PANet to strengthen the suppression of complex interference information at a higher semantic level and highlight the feature expression effect of small corrosion targets.

2.2.3. Receptive Feature Enhancement Network. In the bottom-up course of the WA-PANet, features from different stages need to be fused. Owing to the receptive fields of different branch features are different, so the semantic information is dissimilar. The fusion of multiple feature maps at different semantic levels will greatly weaken the expression ability of multiscale features, which is not conducive to the detection results of the algorithm. Based on the problem, we use the bottleneck layers and dilated convolutions of different scales to construct the receptive feature enhancement network [21] that carries out enhanced extraction on the multibranch fusion outputs of the WA-PANet to advance the feature expression of rusted targets at various scales. The structure of the RFENet module is shown in Figure 5.

The module is a multibranch structure. Firstly, a bottleneck layer is adopted in each branch, namely, a 1×1 convolution for dimensionality reduction and a $n \times n$ convolution to achieve the extraction of different scale features, and then, a 3×3 dilated convolution with a dilation rate of n is followed to capture the feature information in a larger receptive field area, and ultimately, the concat and shortcut operations are applied to fuse the features of different receptive fields. With the purpose of compressing the number of parameters, we deploy two 3×3 convolutions, as well as 1×7 and 7×1 convolutions to replace 5×5 and 7×7 convolutions, respectively. In this study, the module is placed in the four output branches of the WA-PANet to establish the RFENet, which can intensify the effect of feature fusion and promote the recognition accuracy for corrosion objects of different scales.

2.2.4. EIoU Loss. The YOLOV5 algorithm applies the CIOU Loss as the loss function of bounding box regression, which takes into account three geometrical factors, including overlapping area, center points' distance, and aspect ratio. Given a prediction box B and a ground truth B^{gt}, the CIOU Loss can be defined as

$$\begin{cases} L_{\text{CIOU}} = 1 - IOU + \dfrac{\rho^2\left(b, b^{gt}\right)}{c^2} + \alpha\nu, \\ \nu = \dfrac{4}{\pi^2}\left(\arctan\dfrac{w^{gt}}{h^{gt}} - \arctan\dfrac{w}{h}\right)^2, \\ \alpha = \dfrac{\nu}{(1 - IOU) + \nu}, \end{cases} \tag{13}$$

where IOU is the ratio of the intersection area and the union area of the prediction box and the ground truth, b and b^{gt} represent the center of B and B^{gt}, respectively, $\rho(\cdot) = \|b - b^{gt}\|_2$ is the distance between b and b^{gt}, c is the diagonal length of the minimum bounding rectangle between the prediction box and the ground truth, and α and ν are used to reflect the similarity of the aspect ratio between the prediction box and the ground truth. In (13), ν reveals the difference between aspect ratios, rather than the real relationship between w and w^{gt} or h and h^{gt}. When w and h meet formula $\{(w = kw^{gt}, h = kh^{gt}) | k \in R^+\}$, the value of ν is 0, indicating that the length and width of the prediction box and the ground truth are completely matched. As a result, the loss of aspect ratio item is 0, and the bounding box regression process is blocked, which is inconsistent with reality. In addition, in the training process, the calculating process of ν for w and h back propagation to obtain the gradient is denoted in formula (14). According to the formula, we can get $\partial\nu/\partial w = -h/w\partial\nu/\partial h$, in which the signs of $\partial\nu/\partial w$ and $\partial\nu/\partial h$ are opposite. Thus, if one of these two variables (w or h) increases, the other will decrease, which prevents the reduction of the real difference between the prediction box and the ground truth:

$$\begin{cases} \dfrac{\partial\nu}{\partial w} = \dfrac{8}{\pi^2}\left(\arctan\dfrac{w^{gt}}{h^{gt}} - \arctan\dfrac{w}{h}\right) * \dfrac{h}{w^2 + h^2}, \\ \dfrac{\partial\nu}{\partial h} = -\dfrac{8}{\pi^2}\left(\arctan\dfrac{w^{gt}}{h^{gt}} - \arctan\dfrac{w}{h}\right) * \dfrac{w}{w^2 + h^2}. \end{cases} \tag{14}$$

Owing to the unclear definition of ν in the last item of L_{CIOU}, the convergence speed and positioning accuracy of the algorithm are limited. Therefore, we introduce the EIOU Loss [22], an optimization version of the CIOU Loss, to compute regression loss. It is defined as

$$\begin{aligned} L_{\text{EIOU}} &= L_{\text{IOU}} + L_{dis} + L_{asp} \\ &= 1 - IOU + \frac{\rho^2\left(b, b^{gt}\right)}{c^2} + \frac{\rho^2\left(w, w^{gt}\right)}{c_w^2} + \frac{\rho^2\left(h, h^{gt}\right)}{c_h^2}, \end{aligned} \tag{15}$$

where c_w and c_h are the length and width of the minimum bounding rectangle covering the prediction box and the ground truth, respectively. The EIOU loss also consists of

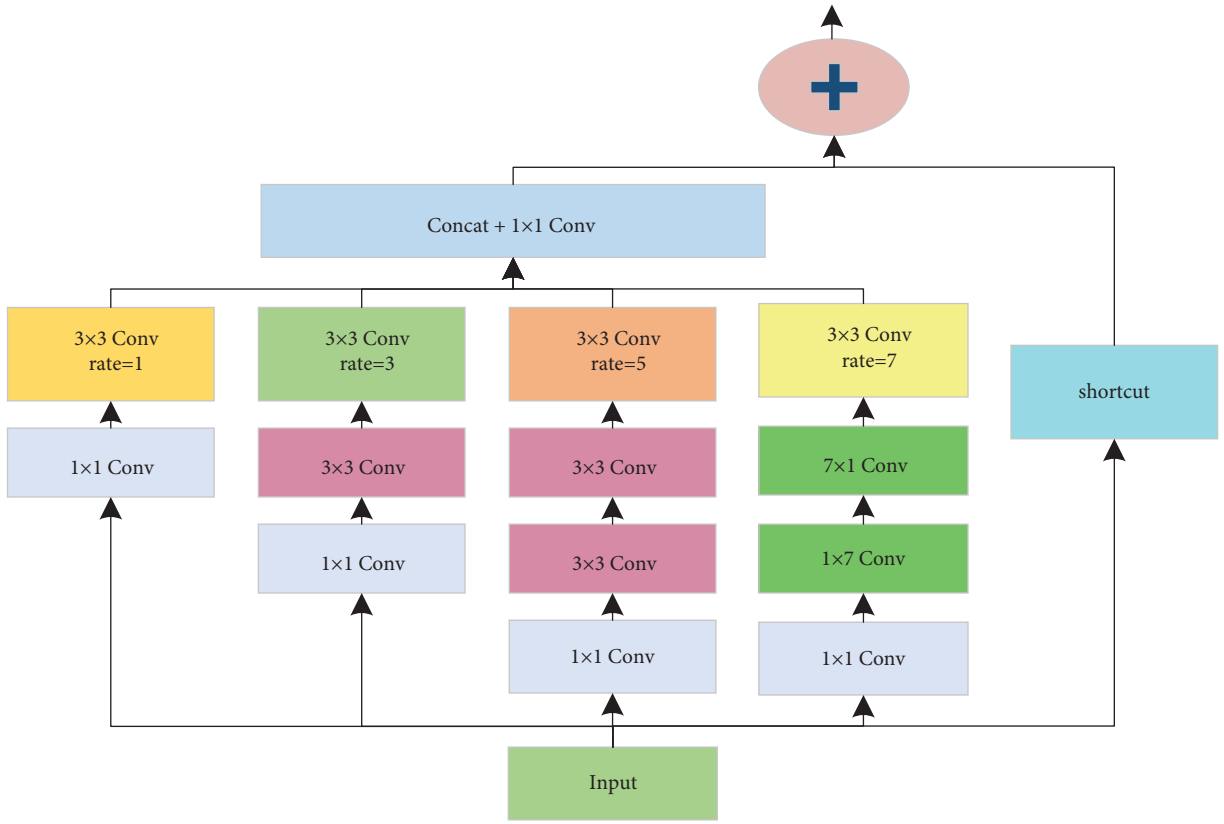


Figure 5: The feature enhancement network module structure.

three parts, the overlapping area loss L_{IOU}, the center points distance loss L_{dis}, and the length and width difference loss L_{aps}. Among them, L_{aps} can directly minimise the length-width gap between the prediction box and the ground truth so that the convergence speed and the location performance of the network are excellent.

3. Results and Discussion

3.1. Experimental Environment and Parameters' Setting. Based on the Ubuntu 18.04 operating system, this study employs the PyTorch deep learning framework and *Python* compiler language to carry out all the experiments. The specific experimental environment and training parameter settings of the improved algorithm are shown in Table 1.

3.2. Algorithm Evaluation Index. In order to verify the effectiveness of the innovative improved method in this study, the Average Precision (AP), the mean Average Precision (mAP), the Frames Per Second (FPS), and the Model Size (MS) are selected to evaluate model performance according to the positioning accuracy of the prediction boxes and the missed and false detection of the targets. The AP measures the precision and recall of a certain class, and its value is the area of the *P-R* curve. The larger the value is, the better the detection performance of the network for this kind of target. The mAP is the mean value of the AP of various categories, which is used to evaluate the overall detection accuracy of the model. The FPS reflects the detection speed; the greater the value is, the better the real time of the algorithm is. The MS refers to the amount of memory occupied by the algorithm, which represents the requirement for storage space.

3.3. Experimental Results and Comparative Analysis

3.3.1. Experimental Result of the PWR-YOLOV5. The self-made dataset of corrosion components is divided into a training set, a validation set, and a test set in a ratio of 8 : 1 : 1. In order to avoid the influence of video frame extraction on the distribution of the dataset, all images are shuffled before partitioning to advance the generalization of the model. In this study, the improved algorithm is trained and tested based on the above partitioning method. Figure 6 shows the *P-R* curve of the PWR-YOLOV5 network during testing, which is drawn from the precision and recall values under all confidence levels of various targets. The area below the curve indicates the average precision of each class, so the AP of FangRust, Fang_NoRust, Jue_Rust, and Jue_NoRust, respectively, reaches up to 96.88%, 95.02%, 95.61%, and 93.96%.

3.3.2. Comparison with YOLOV5. In this study, the YOLOV5 and the modified PWR-YOLOV5 are compared in the following four aspects: average precision, mean average accuracy, frames per second, and model size. The comparative results are recorded in Table 2.

Compared with YOLOV5, the AP of the proposed algorithm on FangRust, Fang_NoRust, Jue_Rust, and

Table 1: Experimental environment and parameter settings.

Experimental environment		Model parameter	
Parameters	Configuration	Parameters	Value
CPU	Intel(*R*) xeon(*R*) *E*5-2680 v4	Optimizer	SGD
GPU	NVIDIA GeForce GTX 3080	Initial learning rate	0.001
CUDA	11.0	Size of images	640 × 640
Python	3.8	Batch size	16
PyTorch	1.9.0	Epoch	350

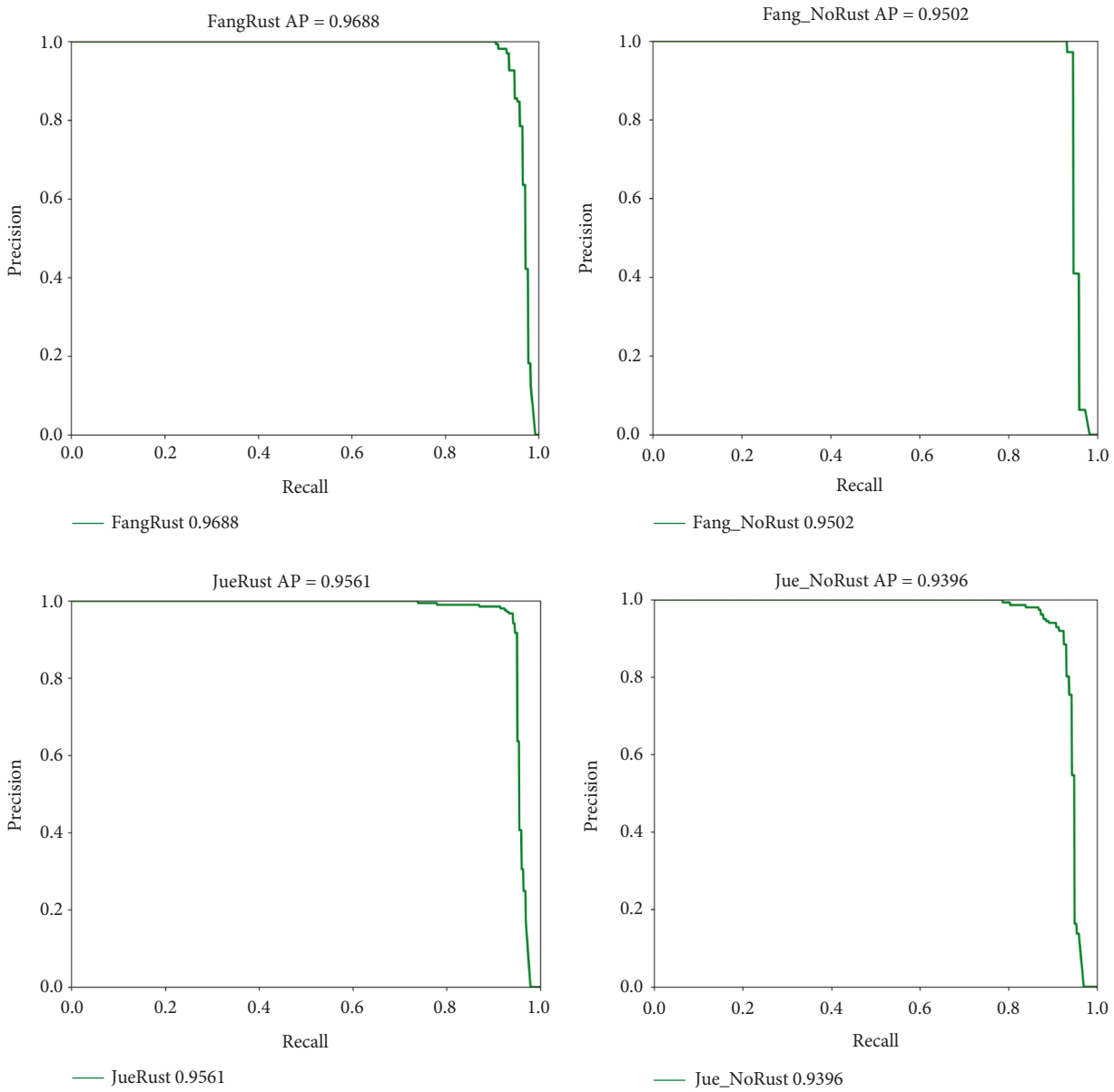


Figure 6: The *P*-*R* curve for four classes of the PWR-YOLOV5 algorithm.

Jue_NoRust is improved by 2.52%, 5.3%, 2.11%, and 10.93%, respectively, and the mAP is increased by 5.22%. The experimental data given above show that the improved method presented in this study can effectively advance the detection accuracy of various targets. However, the introduced PSA modules and RFENet in the PWR-YOLOV5 all increase the number of parameters, resulting in the model being inferior to the YOLOV5 in terms of detection speed and memory overhead.

The detection effect of the algorithm in real complex scenes is shown in Figure 7, in which the original images are on the left, and the YOLOV5 detection results are in the

Table 2: Performance comparison between the YOLOV5 and the improved PWR-YOLOV5.

Models	AP (%)				mAP (%)	FPS	MS (MB)
	FangRust	Fang_NoRust	JueRust	Jue_NoRust			
YOLOV5	94.36	89.72	93.50	83.03	90.15	**109.9**	**14.40**
Ours	**96.88**	**95.02**	**95.61**	**93.96**	**95.37**	64.9	26.65

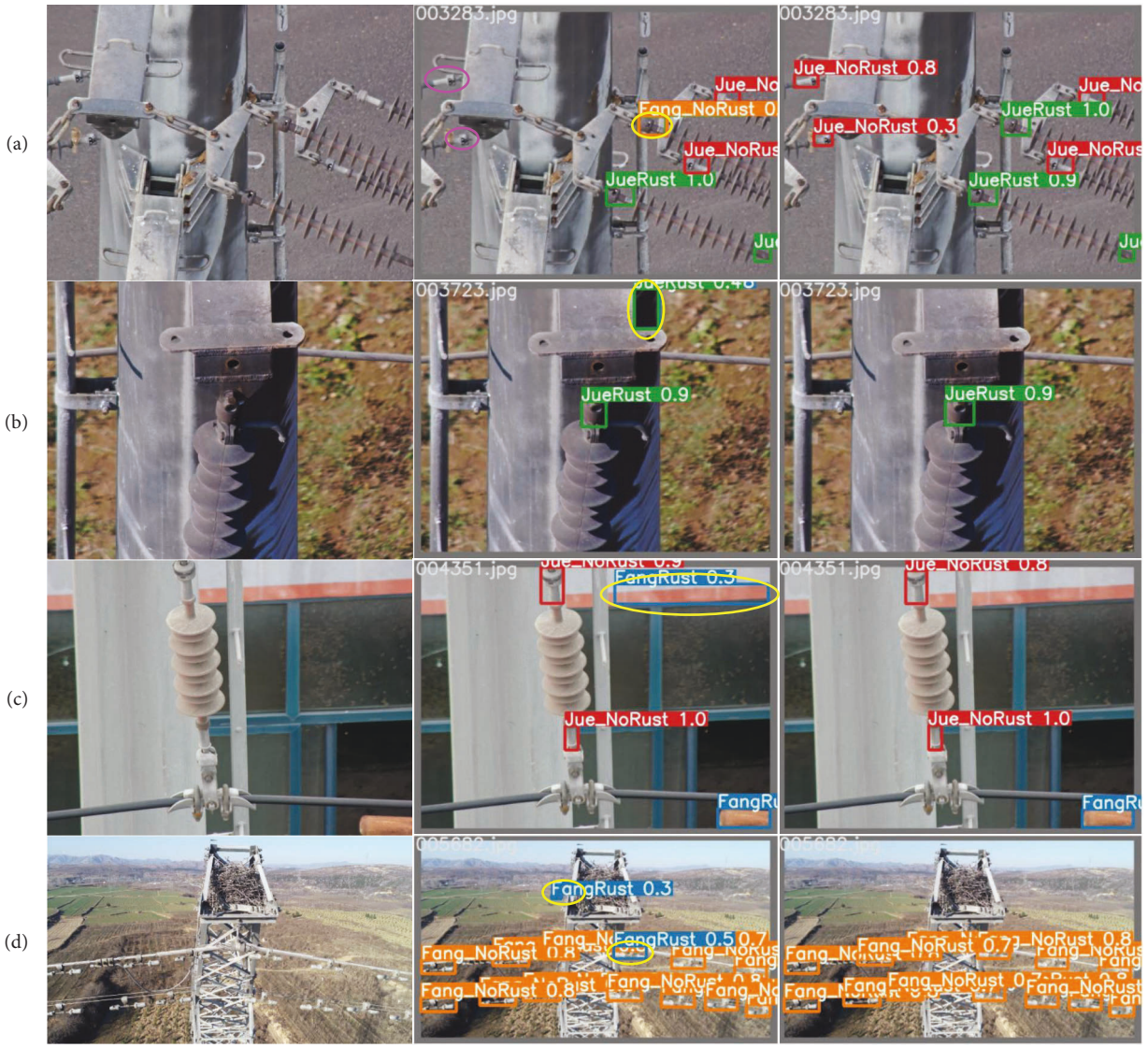


Figure 7: Detection results of the YOLOV5 and the PWR-YOLOV5 in real scenes.

middle. Meanwhile, the PWR-YOLOV5 detection outcomes are on the right. In the figure, the rectangular label boxes in blue, orange, green, and red, respectively, indicate FangRust, Fang_NoRust, JueRust, and Jue_NoRust, and the purple ellipse boxes denote missed detection, and the yellow ellipse boxes represent false detection. The YOLOV5 has missed detections of small objects in (a) and false detections in (a) and (d). In the case of serious background interference in (b), (c), and (d), some background items are mistakenly identified as targets. In contrast, the optimized PWR-YOLOV5 algorithm can significantly improve missed detections, false detections, and object identifications in complicated background.

3.3.3. Comparison with Other Classical Algorithms. For the purpose of validating the performance of the PWR-YOLOV5 algorithm, we chose representative one-stage algorithms SSD [23], RetinaNet [24], YOLOV3 [25], YOLOV4 [26], and two-stage algorithms Faster *R*-CNN [27], and on

Table 3: Performance comparison of different algorithms.

Models	AP (%)				mAP (%)	FPS	MS (MB)
	FangRust	Fang_NoRust	JueRust	Jue_NoRust			
SSD	83.35	91.84	78.45	51.78	76.36	44.7	96.61
RetinaNet	79.29	65.35	74.71	56.18	68.89	16.5	145.92
YOLOV3	75.43	93.83	72.50	45.12	71.72	31.4	93.83
YOLOV4	89.85	**98.42**	90.94	67.51	86.68	21.2	256.34
Faster *R*-CNN	83.54	92.17	66.54	47.22	72.37	22.4	547.02
YOLOX	90.33	96.38	92.60	71.21	87.63	25.5	36.03
CerterNet	85.94	94.41	85.45	63.76	82.39	37.8	131.01
YOLOV5	94.36	89.72	93.50	83.03	90.15	**109.9**	**14.40**
Ours	**96.88**	95.02	**95.61**	**93.96**	**95.37**	64.9	26.65

Table 4: Results of the ablation experiment.

Base	WA-PANet	PSA	RFEnet	EIoU loss	AP (%)				mAP/%	FPS	MS (MB)
					FangRust	Fang_NoRust	JueRust	Jue_NoRust			
✓					94.36	89.72	93.50	83.03	90.15	109.9	14.40
✓	✓				94.52	93.77	94.64	87.89	92.70	83.3	17.52
✓	✓	✓			94.80	93.80	97.90	89.33	93.96	68.5	22.09
✓	✓	✓	✓		95.79	93.93	97.30	93.83	95.21	64.9	26.65
✓	✓	✓	✓	✓	96.88	95.02	95.61	93.96	95.37	64.9	26.65

the basis of anchor-free algorithms CenterNet [28] and YOLOX [29], we developed contrast experiments. The test results of each model are shown in Table 3.

It can be seen from Table 3 that the detection accuracy of the proposed algorithm is optimal for the other three types of targets, except that the AP of Fang_NoRust is lower than YOLOV4 and YOLOX. Compared with the one-stage algorithm YOLOV4 with better detection performance, the AP of PWR-YOLOV5 on Fang_Rust, Jue_Rust, and Jue_NoRust is improved by 7.03%, 4.67%, and 26.45%, respectively. In comparison with the two-stage algorithm Faster *R*-CNN, the average precision of the improved algorithm for four classes was increased by 13.34%, 2.85%, 29.07%, and 46.74%, respectively. In contrast to YOLOX based on the anchor-free algorithm, the detection AP of the optimized YOLOV5 for Fang_Rust, Jue_Rust, and Jue_NoRust is enhanced by 6.55%, 3.01%, and 22.75%, respectively. Contrasted with the mAP, which measures the overall detection accuracy of the model, the PWR-YOLOV5 algorithm is the best, reaching 95.37%. In terms of detection speed and memory occupancy, the proposed algorithm is inferior to the YOLOV5, but better than other common detection models.

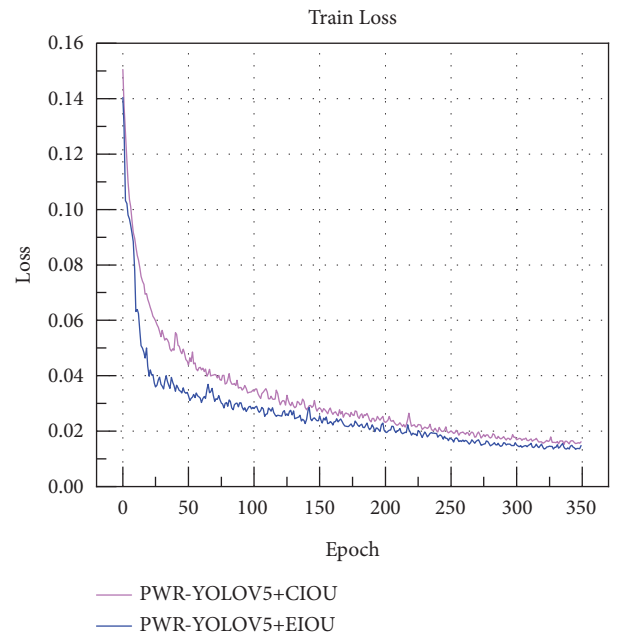


Figure 8: The training loss curves of CIOU Loss and EIOU Loss.

3.3.4. Ablation Experiment. In order to prove the influence of the four improved methods on the detection results, an ablation experiment is designed in this study, which chooses the YOLOV5 algorithm as the base. The experimental results are shown in Table 4.

All of the four proposed methods can improve the detection accuracy of various targets to a certain extent according to Table 4. The WA-PANet feature fusion network constructed by deepening the feature fusion process and introducing skip layer connections and adaptive feature fusion factors can not only ensure the integrity of fusion information but also take into account the influence of features from different stages on the fusion results. Compared with the YOLOV5 algorithm, the mAP advances by 2.55%. The PSA mechanism introduces channel attention on the feature maps of different receptive fields and adaptively associates spatial information and channel weights with $Softmax$ to produce pixel-level attention on objects, and the mAP is increased from 92.70% to 93.96%. The RFENet adopts bottleneck layers and dilated convolutions of different scales to strengthen the feature extraction for the

fusion output of the WA-PANet, which can make full use of features of different receptive fields and enhance the feature expression ability of various scale targets. The mAP of the algorithm advances to 95.21%. The three improvements referred to above to the model structure increase the number of parameters to different degrees, so the detection speed decreases somewhat, but it can still meet the requirement of real-time detection. In addition, the training loss curves of the improved algorithm under the CIOU Loss and the EIOU Loss are respectively drawn in Figure 8. By comparing the convergence of the two curves, it can be seen that the optimization method of the loss function proposed in this study makes the convergence rate faster and the positioning accuracy more superior.

4. Conclusions

Aiming at the problems of poor detection accuracy, serious missed detection, and false detection of corrosion components in distant view inspection by UAV, we propose a PWR-YOLOV5 detection method for corrosion components based on the YOLOV5 algorithm. This method firstly constructs a new feature fusion network, WA-PANet, which makes full use of the features at different stages and improves the detection accuracy of the corroded targets in remote view. Secondly, the PSA mechanism is led into the deep layers of the network to effectively suppress the background information and highlight the feature expression of the pixel-level targets. And then, the bottleneck structures and dilated convolutions of different scales are used to establish the feature enhancement network RFENet to alleviate the problem of feature expression ability weakening caused by feature fusion at different semantic levels. Finally, we apply the EIoU Loss to optimize the loss function of bounding box regression, so as to improve the positioning accuracy for the rusted targets.

The experimental results show that the proposed algorithm has more accurate location performance and higher detection accuracy. Compared with the original network, the average precision on FangRust, Fang_NoRust, JueRust, and Jue_NoRust are heightened by 2.52%, 5.3%, 2.11%, and 10.93%, respectively. The mAP can reach up to 95.37%, and the detection speed is 64.9FPS. Considering the accuracy and speed comprehensively, the proposed algorithm has higher application value, which provides a new idea for the corrosion component inspection of the overhead transmission line by UAV in the mining area.

Data Availability

The code data used to support the findings of the study can be obtained from the corresponding author upon request. But the image data involve in project privace, which is not publicly available.

Conflicts of Interest

The authors declare that the publication of this study does not refer to any conflicts of interest.

Acknowledgments

The authors are thankful to the staff of the Hemei Group Power Supply Department for their assistance in the process of this study. This study was funded by the Training Plan of Young Backbone Teachers in Colleges and Universities of Henan Province (Grant no. 2019GGJS060).

References

[1] L. Zhang, "Accelerating the transition of global clean energy Mineral security has become a national strategy," *China Economic Review*, vol. 2, no. 1, p. 8, 2021.

[2] J. Zhang, *Design and Implementation of Intelligent Remote Patrol Line System of Transmission Line*, University of Electronic Science and Technology of China, Chengdu, Sichuan, 2015.

[3] D. I. Jones, "An experimental power pick-up mechanism for an electrically driven UAV," in *Proceedings of the 2007 IEEE International Symposium on Industrial Electronics*, pp. 2033–2038, IEEE, Vigo, Spain, June 2007.

[4] O. Araar, N. Aouf, and J. L. Vallejo Dietz, "Power pylon detection and monocular depth estimation from inspection UAVs," *Industrial Robot: International Journal*, vol. 42, no. 3, pp. 200–213, 2015.

[5] S. Zhang, Z. Lei, and Y. Liu, "Research and analysis on intelligent inspection scheme of unmanned aerial vehicle based on artificial intelligence," *Chinese Journal of Sichuan Electric Power Technology*, vol. 42, no. 3, pp. 90–94, 2019.

[6] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010.

[7] M. Recky and F. Leberl, "Windows detection using K-means in CIE-lab color space," in *Proceedings of the 2010 20th International Conference on Pattern Recognition*, pp. 356–359, IEEE, Istanbul, Turkey, August 2010.

[8] W. Song, D. Zuo, B. Deng, and H. Zhang, "Corrosion defect detection of earthquake hammer for high voltage transmission line," *Chinese Journal of Scientific Instrument*, vol. 37, no. 1, pp. 113–117, 2016.

[9] Y. Dai, D. Lv, and S. Guo, "Transmission line rusted area detection scheme based on color and texture features," *Chinese Journal of Industrial Control Computer*, vol. 31, no. 9, pp. 39-40, 2018.

[10] X. Huang, X. Zhang, Y. Zhang, and L. Zhao, "A method of identifying rust status of dampers based on image processing," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 8, pp. 5407–5417, 2020.

[11] H. K. Shen, P. H. Chen, and L. M. Chang, "Automated steel bridge coating rust defect recognition method based on color and texture feature," *Automation in Construction*, vol. 31, pp. 338–356, 2013.

[12] C. Sun, X. Qiu, X. Luo, Q. Huang, and C. Cao, "Rust defect recognition method for anti-vibration hammer based on the features of aggregated channels," *Chinese Journal of Computing Technology and Automation*, vol. 39, no. 2, pp. 128–132, 2020.

[13] L. Petricca, T. Moss, G. Figueroa, and S. Broen, "Corrosion detection using AI: a comparison of standard computer vision techniques and deep learning model," in *Proceedings of the Sixth International Conference on Computer Science,*

Engineering and Information Technology, pp. 91–99, IEEE, New Jersey, NJ, USA, May 2016.

[14] K. Zhang, Q. Hou, and W. Huang, "Defect detection of anti-vibration hammer based on improved faster R-CNN," in *Proceedings of the 2020 7th International Forum on Electrical Engineering and Automation IFEEA*, pp. 889–893, IEEE, Hefei, China, September 2020.

[15] Z. Wu, W. Xiong, J. Ren, and M. Jiang, "Corrosion object detection of power equipment based on lightweight SSD," *Chinese Journal of Computer Systems & Applications*, vol. 29, no. 2, pp. 262–267, 2020.

[16] H. Zhang, L. Wu, Y. Chen, R. Chen, and S. Kong, "Attention-Guided Multitask Convolutional Neural Network for Power Line Parts Detection," *IEEE Transaction on Instrumentation and Measurement*, vol. 71, Article ID 5008213, 2022.

[17] Yolov5, https://github.com/ultralytics/yolov5, 2021.

[18] M. Wu, Z. Zhang, Y. Song, Z. Shu, and B. Li, "Small object detection network based on adaptive feature enhancement," *Chinese Journal of Laser & Optoelectronics Progress*, vol. 59, pp. 1–14, 2022.

[19] H. Zhang, K. Zu, J. Lu, Y. Zou, and D. Meng, "EPSANet: An Efficient Pyramid Squeeze Attention Block on Convolutional Neural Network," 2021, https://arxiv.org/abs/2105.14447.

[20] J. Hu, S. Li, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pp. 7132–7141, IEEE, Salt Lake City, Utah, USA, June 2018.

[21] S. Liu, D. Huang, and Y. Wang, "Receptive field block net for accurate and fast object detection," *Proceedings of the European Conference on Computer Vision ECCV September*, pp. 385–400, Munich, Germany, 2018.

[22] Y. F. Zhang, W. Ren, Z. Zhang, J. Zhen, W. Liang, and T. Tieniu, "Focal and Efficient IOU Loss for Accurate Bounding Box Regression," 2021, https://arxiv.org/abs/2101.08158.

[23] W. Liu, D. Anguelov, D. Erhan, E. Dumitru, S. Christian, and R. Scott, "SSD: single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, pp. 21–37, Springer International Publishing, Amsterdam, Netherlands, October 2016.

[24] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision ICCV*, pp. 2980–2988, IEEE, Venice Italy, October 2017.

[25] J. Redmon and A. Farhadi, "YOLOV3: An Incremental Improvement," 2018, https://arxiv.org/abs/1804.02767.

[26] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOV4: Optimal Speed and Accuracy of Object Detection," 2020, https://arxiv.org/abs/2004.10934.

[27] S. Ren, K. He, R. Girshick, J. Sun, and R.-C. N. N. Faster, "Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[28] K. Duan, S. Bai, L. Xie, Q. Honggang, H. Qingming, and Q. Tian, "Centernet: keypoint triplets for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision ICCV*, pp. 6569–6578, IEEE, Seoul, Korea, October 2019.

[29] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding yolo Series in 2021," 2021, https://arxiv.org/abs/2107.08430.