energies

MDPI

Article

Novel Cloud-Edge Collaborative Detection Technique for Detecting Defects in PV Components, Based on Transfer Learning

Hongxi Wang [1], Fei Li [1], Wenhao Mo [2], Peng Tao [1], Hongtao Shen [1], Yidi Wu [3], Yushuai Zhang [1] and Fangming Deng [4,*]

[1] Marketing Service Center, State Grid Hebei Electric Power Co., Ltd., Shijiazhuang 050035, China
[2] China Electric Power Research Institute, Beijing 100192, China
[3] State Grid Hebei Electric Power Co., Ltd., Shijiazhuang 050021, China
[4] School of Electrical and Automation Engineering, East China Jiaotong University, Nanchang 330013, China
* Correspondence: 2464@ecjtu.edu.cn

Abstract: The existing techniques for detecting defects in photovoltaic (PV) components have some drawbacks, such as few samples, low detection accuracy, and poor real-time performance. This paper presents a cloud-edge collaborative technique for detecting the defects in PV components, based on transfer learning. The proposed cloud model is based on the YOLO v3-tiny algorithm. To increase the detection effect of small targets, we produced a third prediction layer by fusing the shallow feature information with the stitching layer in the second detection scale and introducing a residual module to achieve improvement of the YOLO v3-tiny algorithm. In order to further increase the ability of the network model to extract target features, the residual module was introduced in the YOLO v3-tiny backbone network to increase network depth and learning ability. Finally, through the model's transfer learning and edge collaboration, the adaptability of the defect-detection algorithm to personalized applications and real-time defect detection was enhanced. The experimental results showed that the average accuracy and recall rates of the improved YOLO v3-tiny for detecting defects in PV components were 95.5% and 93.7%, respectively. The time-consumption of single panoramic image detection is 6.3 ms, whereas the consumption of the model's memory is 64 MB. After cloud-edge learning migration, the training time for a local sample model was improved by 66%, and the accuracy reached 99.78%.

Keywords: cloud-edge collaboration; defect recognition; transfer learning

check for updates

Citation: Wang, H.; Li, F.; Mo, W.; Tao, P.; Shen, H.; Wu, Y.; Zhang, Y.; Deng, F. Novel Cloud-Edge Collaborative Detection Technique for Detecting Defects in PV Components, Based on Transfer Learning. *Energies* **2022**, *15*, 7924. https://doi.org/10.3390/en15217924

Academic Editor: Antonino Laudani

Received: 21 September 2022
Accepted: 23 October 2022
Published: 25 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Solar power generation has great advantages in solving the energy crisis problem and the environmental pollution problem. However, the traditional methods of manual inspection may pose a serious threat to the safety of inspectors when they carry out operational and maintenance inspections on a large number of photovoltaic (PV) components with various defects [1–3]. Therefore, automatic inspection techniques, such as image capture by unmanned ariel vehicles (UAVs) that carry visible light cameras on PV panel arrays and image recognition based on deep learning, have been gradually applied to the detection of defects in PV modules, due to their advantages of high efficiency and accuracy [4,5].

The detection of defects in PV components is a target-detection task in the field of machine vision. In this field, target-detection algorithms based on convolutional neural networks have emerged in recent years, and have achieved wide application due to their high accuracy and robustness [6]. Among the current typical deep-learning target-detection algorithms, one class of target-detection algorithms is based on region extraction, which divides detection process into two types: localization and classification. Representative algorithms are R-CNN [7], Fast R -CNN [8], and Faster R-CNN [9]. Another class of

target-detection algorithms is based on an end-to-end approach, which directly completes localization and classification in one step; representative algorithms are SSD [10] and YOLO [11].

Although the region extraction-based target-detection algorithm has an advantage in accuracy compared with the end-to-end based target-detection algorithm, the process of generating and extracting the candidate regions of an object is computationally intensive and slow, which makes this type of algorithm unable to meet the demand for real-time detection of surface defects in PV modules. The YOLO series of target-detection algorithms can maintain high accuracy with high detection speed, and the feature extraction process is generally more focused, resulting in accurate and fast detection of PV module surface defects [12–14]. Based on the study mentioned above, when UAVs were applied to PV module surface-defect detection, the SSD-based algorithm of Zhong et al. [15] detected PV module defects, but required a longer time to process the results and had poor real-time performance when the detection accuracy of the target was high; when network models such as YOLO v3 and YOLO v4 were deployed to detect the targets, the real-time performance was poor and the models were difficult to apply to mobile workstations. Therefore, this paper aims to improve the network architecture via YOLO v3-tiny [16], a lightweight network that is more suitable for deployment to the edge.

With the rapid development of network communication technology and the increasing number of user-owned businesses, the powerful computing power of the cloud computing center is used increasingly to store and analyze visual data [17,18]. Using the powerful computing power of cloud computing centers for visual data storage and analysis, visual images are identified and sensed by deep-learning techniques, such as target detection, semantic segmentation, and target tracking [19]. Hu et al. [20] proposed a fast-feature-embedding (Caffe) deep-learning method using a fast regional convolutional neural network (RCNN) structure for the large data of power grid operation, which can effectively extract the image features of towers and improve the speed of identifying transmission-line pole-tower images. Luo et al. [21] studied a target-detection algorithm using neural network self-learning for the defect detection of devices in UAV images with different backgrounds, lighting, and scales, and the algorithm improved by nearly 60%, compared with traditional deep-learning target-detection algorithms. For PV plant fault detection, Carletti et al. [22] devised a model-based panel detection method that used the structural rules of PV strings and a new local hotspot detection technique to find local maxima in PV panel regions and, finally, achieve global detection, which improved fault detection accuracy and was reliable.

Traditional cloud computing can no longer meet clients' requirements for rapid responses to network services, data protection, and network flexibility. Edge computing makes up for some of the shortcomings of cloud computing by providing computing services at the device or data end, but cloud computing is superior to edge computing in terms of construction cost and functional scalability. Therefore, the combination of edge computing and cloud computing can better meet the needs of various application scenarios, and the development of cloud-edge collaboration is inevitable.

At present, there is a lack of collaboration between clear cloud-edge architecture and cloud-edge methods for the detection of surface defects in photovoltaic modules. Therefore, this paper refers to the cloud-edge collaboration framework in [23,24] in the electrical Internet of Things and the power distribution fault-analysis service, builds a clear cloud-edge-system framework for the characteristics and application requirements of photovoltaic modules, and introduces migration learning via improved methods. Transfer learning provides the network model with sufficient knowledge by applying the acquired capabilities to the relevant domain models, so that this paper's model can reduce training time, improve recognition accuracy, and complete the target tasks with greater accuracy.

Accordingly, in this paper, we focus on the characteristics and application requirements of PV modules to study the application mode of cloud-edge collaboration architecture in the detection of defects in PV modules. In summary, this paper proposes an improved

deep-learning model, based on cloud-edge collaboration, for the detection of surface defects in PV modules. The contributions of this paper are as follows.

(1) A collaborative cloud-edge-side PV module defect-detection architecture is proposed to solve the contradiction between the real-time requirements of PV module defect detection and the complex computational burden, as well as the problem of sparse PV module defect samples, and to improve the adaptability of the edge side to different applications. This approach makes it possible to apply high-performance deep-learning algorithms at the edge side to complete PV module defect detection accurately in real time.

(2) To solve the problems of low detection accuracy for small targets and long processing times for the detection process during PV module defect detection, a target-detection model based on an improved YOLO v3-tiny network is proposed as the third prediction layer, fusing shallow features with the splicing layer before the second YOLO prediction layer.

(3) To further increase the network model's ability to extract target features, the residual module is introduced on the YOLO v3-tiny backbone network, drawing on the idea of residual networks, to increase the network's depth and learning ability, thus improving the detection capability of the network.

The remainder of the paper is organized as follows: Section 2 provides a detailed description of the scheme proposed in this paper; and Section 3 presents the experimental results and discusses the scheme presented in this paper. Finally, the complete work and the advantages of this paper's approach are summarized in Section 4.

Overall Framework

In this paper, a collaborative cloud-edge PV module defect-detection model is constructed, as shown in Figure 1. The network architecture of the whole model unites the edge server and the cloud for data processing and consists of a drone side, an edge side, and a cloud side. Figure 1 presents the exact relationship and specific functions of the three port facets.

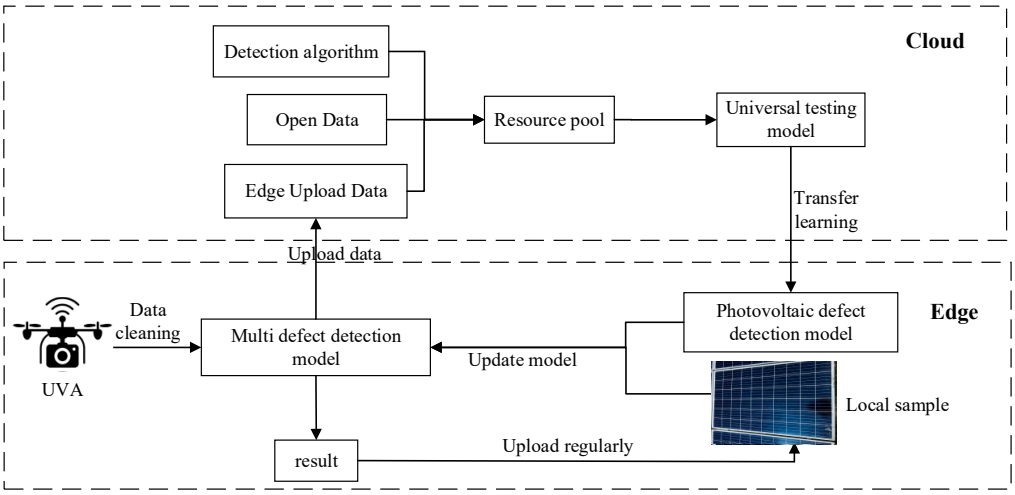


Figure 1. Overall framework design.

The first side is the UAV side, consisting of N UAVs. Given the geographic information of the PV system and the predefined inspection commands, the onboard processor plans the patrol route and sends control commands and image acquisition commands to the UAV flight controller. The UAVs carry infrared cameras and image stores to collect visible and thermal images of the PV modules, and GPS is used to provide accurate PV module detection of defective geographic locations. The captured PV module images and the GPS status information are transmitted to the ground edge server in real time via a wireless communication network.

The second side is the edge-server side. This port consists of M edge servers. The edge server can realize fast real-time responses as well as providing personalized service; therefore, each edge server has two tasks: the first task is the localization of the generic model, whereby the generic detection model generated by the cloud is migrated to the edge side, combined with local PV module sample training to generate a localized PV module defect-detection model; the second task is the data processing and calculation, receiving the real-time data uploaded by the UAV and outputting the results after testing by the local model. The results are fed back to the client, and the real-time data and detection results are uploaded to the cloud for storage to achieve real-time detection of PV module defects.

The third side is the cloud-server side, which serves as a server with sufficient storage resources and abundant computing resources that can be used to generate and optimize a generic detection model. The collected data processed by each edge server and the public data set of PV module defect detection constitute the cloud-side resource base. Based on the cloud-side resource, a generic model for PV module defect detection is generated. Due to the continuous uploading of data at the edge, the model in the cloud is continuously updated to generate better migration decisions, and the optimized neural network parameters are returned to the edge segment to improve the processing efficiency of the edge for subsequent tasks.

As can be seen from Figure 1, the data interaction between the cloud and the edge in the PV component-defect-detection task mainly consists of two tasks: (i) the edge regularly uploads the personalized samples that are stored locally to the cloud storage to enrich the training and validation sets in the cloud; and (ii) as the data set stored in the cloud is updated, the optimization of the generic detection model parameters is completed via training at regular intervals, and the optimized model parameters are migrated to the edge to perform model correction.

2. Design and Implementation of the Defect-Detection Algorithm

2.1. Cloud-Based Algorithm Construction and Training

2.1.1. YOLO v3-Tiny Target-Detection Algorithm

YOLO v3-tiny is a lightweight real-time target-detection algorithm, which is more in line with the above-mentioned design criteria. The lightweight network structure is more suitable for deployment to the edge; v3-tiny is simplified from the YOLO v3 network structure, which can reduce memory consumption during model training, making target detection faster and better able to meet the demand for real-time detection in complex environments. The network structure of YOLO v3-tiny consists of 11 CBL modules, six maximum pooling layers (Maxpool), two convolutional layers, two routing layers (route), one up-sampling layer, and two YOLO prediction layers, totaling 24 network layers.

The center of the camera is approximately 5 m from the vertical height of the PV panel when the UAV is shooting, the proportion of broken grid and spot pixels on the PV module is small, and the module cracks take up even fewer pixels, resulting in a high-miss detection rate of small targets, such as the tiny cracks caused by YOLO v3-tiny. To solve this problem, this study constructed a third scale prediction layer by fusing shallow features and introducing a residual module to achieve improvement of the YOLO v3-tiny algorithm.

2.1.2. Fusion of Shallow Features to Construct a Third Prediction Layer

The features extracted by the shallow network are good for small target localization, but weak in semantic representation. The features extracted by the deep network contain rich semantic information, but easily lose the localization information of small targets [25]. When the input network image size is 416 pixels × 416 pixels, YOLO v3-tiny has two target-detection scales, 13 × 13 and 26 × 26. The 13 × 13 detection layer is suitable for large targets and the 26 × 26 detection layer is suitable for medium targets. The camera is mounted on the bottom of the UAV to collect information; the common resolution of the image is 2048 pixels × 2448 pixels, so the component defects will show up small in the image. When the YOLO v3-tiny model detects these small targets in the image, it is

easy to miss the detection phenomenon. To address these problems of the model and to overcome the shallow low-dimensional features of YOLO v3-tiny that are lost in the deeper layers of the network, and thus result in missed detection of small targets, we added a 52 × 52 detection scale layer as the third prediction layer by fusing the shallow feature information with the stitching layer in the second detection scale. The feature information of the third prediction layer comes from the feature map after the same dimensional splicing of the shallow layer and the second layer, which contains more low-dimensional feature information of the image and helps the model to improve the detection ability of small targets, thus minimizing the missed detection of targets.

2.1.3. Introduction of Residual Modules

This study draws on the idea of residual networks (ResNET) [26] to introduce a deep residual framework to solve the gradient vanishing problem, i.e., allow the convolutional network to learn the residual mappings, instead of expecting each stacked layer of the network to completely fit the potential mappings. The residual network consists of residual modules; each residual module contains two branches: a constant mapping branch and a residual branch, as shown in Figure 2. Assume that in the residual learning module, when the input is x its base mapping is noted as $H(x)$, and it is hoped that its learned residual function is $F(x) = H(x) - x$, such that the output learning feature is $F(x) + x$. When the residual is 0, at this point the building block only performs constant mapping, and the detection accuracy of the network model is at least not lost. In fact, a residual function of 0 cannot exist, which allows the residual learning unit to learn new feature information based on the constant mapping, thereby improving the network learning efficiency. The residual network is formulated as follows:

$$y = F(x, \{W_i\}) + x \tag{1}$$

$$y = F(x, \{W_i\}) + W_s x \tag{2}$$

where y represents the output in the residual module, $F(x, \{W_i\})$ represent the post-training residual mapping function, and W_i, W_s represent the input parameters.

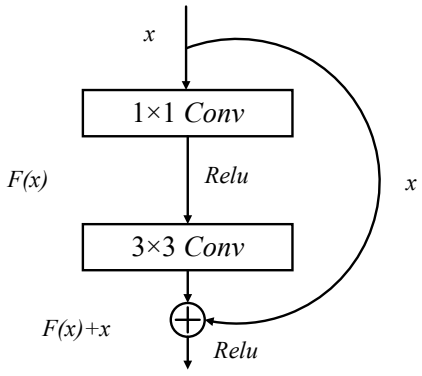


Figure 2. Residual module.

The addition of the residual module means that the features of small targets do not disappear with the gradient as the network deepens, and it can better improve the problem of a high-miss detection rate for small targets. By introducing 1 × 1 and 3 × 3 convolutional kernels to form a residual module, the 1 × 1 convolutional pair of convolutional kernel channels not only plays a role in dimensionality reduction, which will reduce the computational complexity when the computation is large, but also introduces more nonlinear transformations to the neural network. This process obviously increases the depth of the residual block and improves the representation ability of the residual network. The final improved YOLO v3-tiny network model is shown in Figure 3.

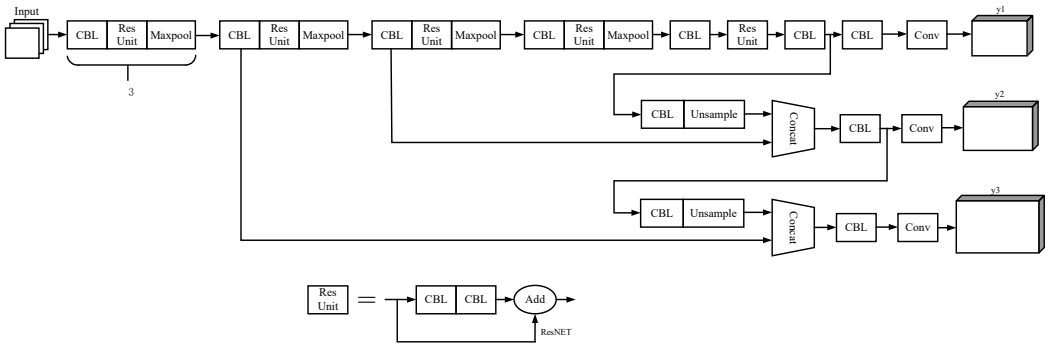


Figure 3. Improved YOLO v3-tiny network structure diagram.

In Figure 4, at the beginning of the task, the publicly available PV component dataset is mainly used for generic model training. As the task proceeds, there will be edge-side applications as locally generated samples are added to the database for continuous update training. The training samples are proportionally divided into training and validation sets during training, and the model parameters are updated by back-propagation for each training session in order to provide a continuous and fast generic model parameter update service for different task scenarios during the whole process of detection, by using the rich computing resources in the cloud.

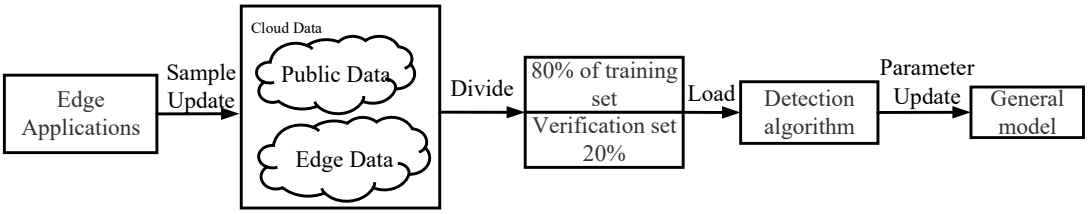


Figure 4. Cloud-based detection model training and update process.

2.2. Edge (Server) Side Algorithms and Training

The task at the edge side is mainly the real-time detection of defects in PV modules, and its algorithm follows the basic framework of the algorithm in the cloud, so that the generic model parameters from the cloud can be migrated to the edge side via transfer learning. However, at the same time, it needs to be adjusted according to different edge-side characteristics. The main tuning strategies are as follows: (1) adjusting the input image size according to the different image characteristics at the edge; and (2) choosing the matching neural network configuration according to the computing resources and local samples at the edge.

The update and application of the edge-side algorithm are shown in Figure 5.

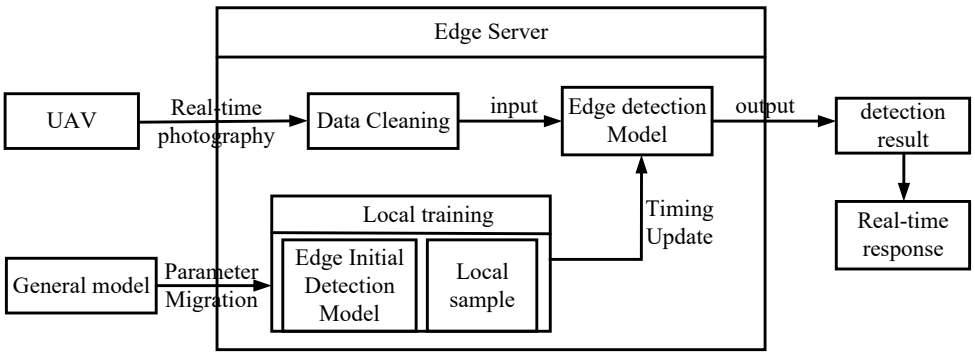


Figure 5. Flow of edge-side algorithm detection and update.

In Figure 4, the detection model initialized at the edge side is obtained from the cloud after parameter migration. The detection model is localized and trained on the edge controller using local samples to finally generate a localized detection model that can be applied to the edge side. At this point, the detection results can be generated in real time by uploading the real-time PV module images on the localized detection model, and the corresponding responses can be made, based on the detection results.

3. Experiments and Conclusions

3.1. Experimental Platform

The software and hardware environments for this paper are shown in Table 1. The experiments were based on the deep-learning open source frameworks. Tensorflow and Keras, and were programmed in Python language.

Table 1. Experimental platform.

Categories	Version
Operating system	Windows 10
CPU	Intel Core i9-10900k
GPU	NVIDIA GeForce GTX 3080
RAM	32 Gb
Tensorflow-gpu	Tensorflow-gpu1.13.2
Keras	Keras 2.1.5
Cuda	Cuda 10.0
Cudnn	Cudnn 7.4.1.5

3.2. Analytical Comparison of Different Models' Detection Results

In order to improve the detection accuracy of small targets in images and reduce the missed detection phenomenon, the input network image size was changed to 608 pixels × 608 pixels and the training parameters were batch size(4). The initial learning rate of 0.001 and the loU threshold of 0.6 YOLO v2 used K-means clustering to obtain the size of the prior frame [27], and YOLO v3 followed this idea [28]. A priori, boxes were initially candidate boxes with fixed widths and heights determined for different scale network layers and were used for logistic regression to detect bounding boxes. Therefore, the choice of the prior frame had an impact on the performance of the detector and on the final detection results. Before the model training started, the K-means algorithm was used to cluster nine anchor box (anchor) sizes to make the model more closely fit the morphological sizes of hot spots and cracks in the dataset, and the final 19 × 19 scale prior frames obtained were (32, 38), (62, 87), and (397, 413) to detect large targets; 38 × 38 scale prior frames were (9,43), (10,16), (13,52) to detect medium targets; and 76 × 76 scales with prior frames of (3,17), (5,27), (7,38) to detect small targets. All network models were trained from scratch, with training rounds (epochs) of 500 until the end of the model convergence.

In Figure 6, the training results of the model are presented using the loss function. It can be seen that the loss curve gradually tends to be stable, and the loss value finally fluctuates around 0.3, achieving the desired convergence of the model.

After training was completed, 220 test set images from the dataset were used to test the YOLO v3-tiny, YOLO v3, and improved YOLO v3-tiny models, and the results are shown in Table 2. The improved YOLO v3-tiny model for PV module defect detection improved 5.2% in recall, 5.6% in mean accuracy, and 4.6% in harmonic mean F1 over the original YOLO v3-tiny, and the improved model F1 was closer to 100%. Compared with YOLO v3, the improved model had 2.6% higher recall, 1.5% higher mean accuracy, and 2.1% higher summed mean F1. The improved model had about one-fourth of the memory of YOLO v3, took one-half less time to detect than YOLO v3, and achieved an average detection frame rate of 84.2 f/s for video streams, which was 49.0% faster than YOLO v3. Although the improved model memory was about 1.84 times that of the YOLO v3-tiny model and the detection elapsed time was 1.4 times that of YOLO v3—tiny, the average accuracy mean

and recall were improved significantly, and met the demand for real-time defect detection by UAVs in motion, which better balances the detection accuracy and detection speed.

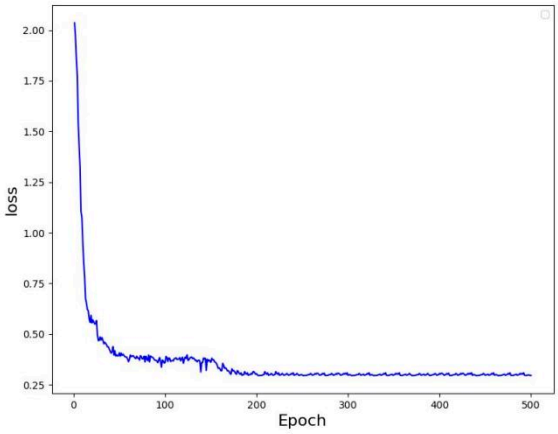


Figure 6. Model training results graph.

Table 2. Comparison of detection results of different models.

Models	R/%	F1/%	mAP/%	Single Image Detection Elapsed Time/ms	Model Memory/MB
YOLO v3-tiny	88.9	89.7	90.3	4.7	35.3
YOLO v3	91.4	92.3	94.2	13.1	251.7
This paper's model	94.2	94.6	95.8	6.2	63.4

3.3. Analysis of the Effect of Different Input Sizes on Model Performance

The experiments were conducted for different input sizes and the results are shown in Table 3. It can be seen that compared with the YOLO v3-tiny model, the model proposed in this study maintained a high detection capability at a smaller input size of 416 pixels × 416 pixels and had the largest improvement over YOLO v3-tiny at the 416 pixels × 416 pixels size, indicating that this paper's model had a strong robustness.

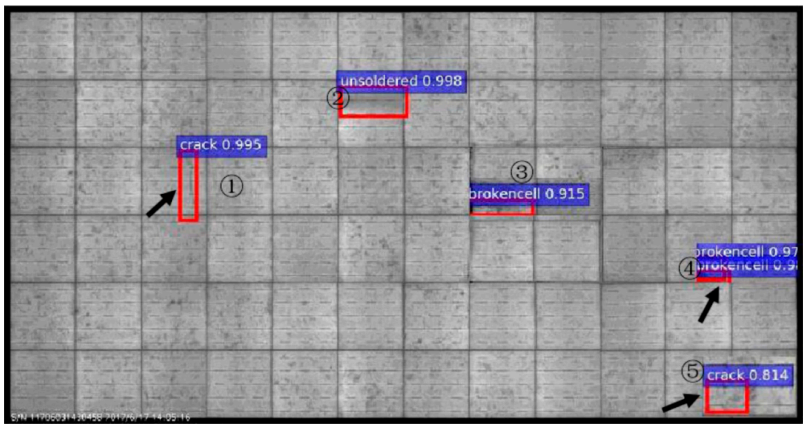
Table 3. Comparison of model detection results for different input sizes.

Input Size/ (Pixels × Pixels)	Models	R/%	F1/%	mAP/%	Single Image Detection Elapsed Time/ms	Model Memory/MB
512 × 512	YOLO v3-tiny	67.1	76.6	75.8	4.4	35.3
	YOLO v3	90.9	92.6	93.9	13.1	251.7
	This paper's model	91.7	92.9	94.5	6.1	63.4
416 × 416	YOLO v3-tiny	62.3	67.2	68.8	4.2	35.3
	YOLO v3	90.1	85.3	91.3	13.3	251.7
	This paper's model	90.4	88.9	93.7	6.2	63.4

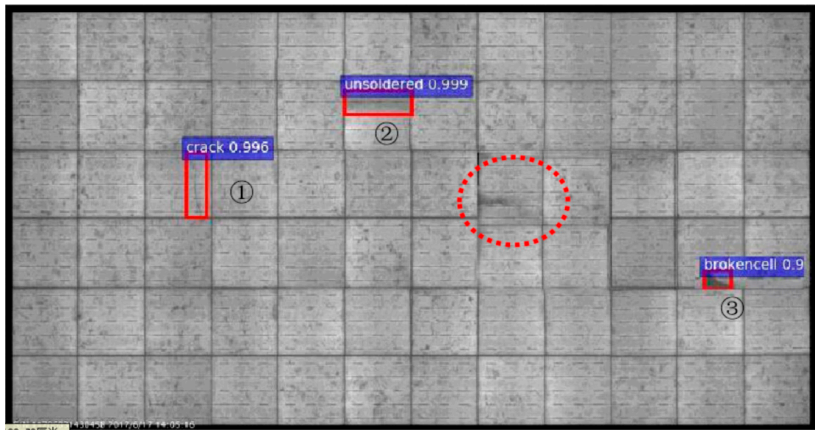
3.4. Comparison of Detection Effects of Different Models

Figure 7 shows the overall schematic diagram of different defect detection results using three algorithms: Yolo v3-tiny, Yolo v3, and this paper's algorithm model. It can be seen that the three algorithms can detect three kinds of defects: cracks, crack defects, and false welding (different defects detected are labeled in the figure). Specifically, Figure 7a shows the detection result of the Yolo v3-tiny algorithm, wherein, ① is the detected crack defect. Obviously, the crack defect is elongated, and the marked position of the boundary box is not sufficiently accurate. Compared with the defect size, the boundary box is large. ④ In order to detect the crack defects, there is a problem of repeated boundary boxes. ⑤ For the detected crack defect, because the gray level of the defect area and the background area are very close, the detection result indicates a false alarm (false detection) area. Figure 7b

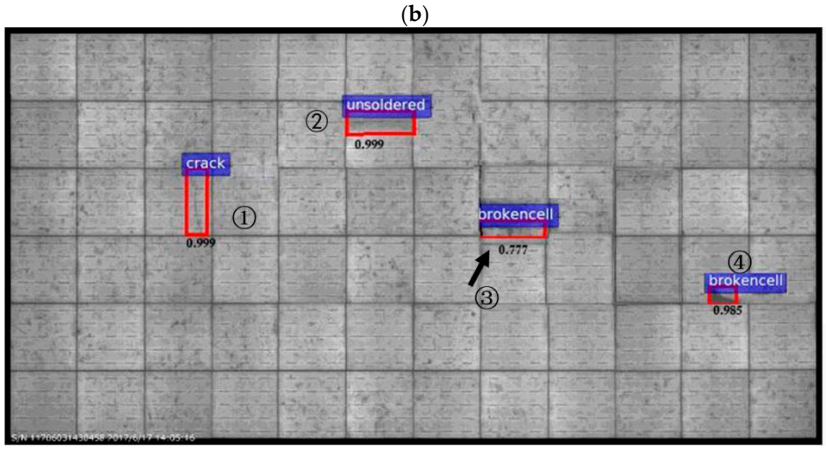
shows the detection results of the Yolo v3 algorithm. The dotted circular frame indicates a crack defect. Because the defect in this area is hidden, the feature is not obvious, and the defect area is very similar to the background area; therefore, the detection results are under reported (under detected) in this area. Figure 7c shows the detection results of this paper's algorithm. It can be seen that cracks, false welds, and crack defects are accurately detected at ①, ② and ④, with very high confidence. ③ is a detected crack defect, although the confidence is a little low; it also exceeds 0.75. Overall, the detection results of this paper's algorithm are better than the other two algorithms, with high detection accuracy, a low false alarm rate (false detection rate),), and high confidence value.



(**a**)



(**b**)



(**c**)

Figure 7. The YOLO v3-tiny, YOLO v3, and this paper's algorithm are used to detect different defects: (**a**) the YOLO v3-tiny algorithm detection result; (**b**) theYOLO v3 algorithm detection result. As shown in (**c**), this paper improves the algorithm defect-detection result.

3.5. Collaborative Cloud-Edge-Side Transfer-Learning Comparison Experiments

In this experiment, a generic model for PV module defect detection was obtained by fully training the public dataset of PV module defects obtained from the web, based on the improved YOLO v3-tiny algorithm on a cloud-mounted GPU server.

At the edge side, some defect data from a PV module dataset of a PV power station in Jiangxi [29], which were similar to the defect types of the transfer-learning source domain dataset, were used to simulate the personalized PV module data collected at the edge side. In this experiment, four types of defects—cracks, broken grids, hot spots, and shadows—were selected to simulate the surface defects of PV modules collected at the edge side.

Through the migration of generic model parameters in the cloud and personalized tuning and training at the edge, the model at the edge detected and identified defects in PV modules.

3.5.1. Exploration of the Selection of Transfer-Learning Strategies and Correspondence Effects

Generally, there are two strategies to perform transfer learning: (i) fine tuning, which involves using a pre-trained network on the base dataset and training all layers in the target dataset; and (ii) freeze-and-train, which involves keeping all layers except the last layer (or layers) frozen (without updating the weights) and training the last layer. In this transfer-learning experiment, the two approaches were applied separately to transfer learning, and the suitable transfer-learning approach was selected by comparing the accuracy of model prediction. The gradient design of the number of training set samples in the target domain in the experiments is shown in Table 4. In each experiment, 140 sets of untrained data with labels were selected as the validation sets for each type of defect, to verify the accuracy of defect detection.

Table 4. Setting of each training sample size in the experiment's target domain.

Training Set	Contains the Number of Samples for Each Type of Defect	Total Number of Defective Samples per Category	Training Set: Number Set Totals
Training set 1	20	1 240	1:62
Training set 2	60	1 240	3:62
Training set 3	120	1 240	3:31
Training set 4	240	1 240	6:31
Training set 5	620	1 240	1:2
Training set 6	1000	1 240	25:31

In this experiment, the above-mentioned PV module defect-detection algorithm was first constructed; then, the constructed neural network was trained in the training process using training sets l to 6 in turn, as shown in Table 4. The experimental results are shown in Table 5 and Figure 8.

Table 5. Results of experiments without transfer learning.

Number of Samples	Number of Training Sessions										
	1	10	20	30	40	50	60	70	80	90	100
20	0.0624	0.1724	0.2499	0.3662	0.4436	0.4765	0.4873	0.4855	0.5016	0.4936	0.4994
60	0.0624	0.4048	0.6505	0.6865	0.7263	0.7484	0.7622	0.7464	0.7464	0.7595	0.7638
120	0.0636	0.7028	0.7975	0.8676	0.8442	0.8723	0.8955	0.9028	0.9056	0.9002	0.9003
240	0.2260	0.7728	0.8384	0.8677	0.8857	0.8742	0.9064	0.9169	0.9251	0.9213	0.9306
620	0.3840	0.9478	0.9611	0.9764	0.9852	0.9847	0.9823	0.9850	0.9844	0.9914	0.9899
1000	0.5606	0.9565	0.9772	0.9933	0.9928	0.9932	0.9951	0.9963	0.9963	0.9973	0.9966

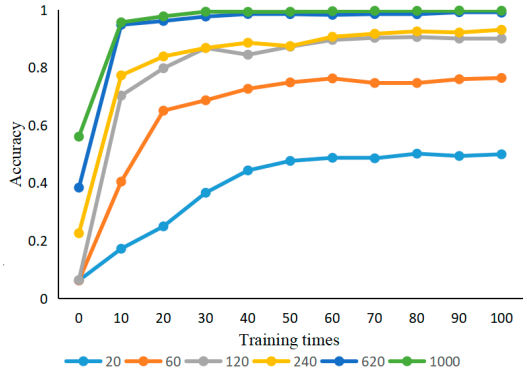


Figure 8. Results of experiments without transfer learning.

The optimal improved algorithm derived in the previous section was first fully trained in the cloud using the source domain samples and the transfer-learning approach to finally obtain a model with a correct defect recognition rate that was close to 95%. All of the parameters in this model were transferred to the edge server side.

First, fine tuning was used to perform transfer learning, i.e., the model parameters transferred from the cloud were directly loaded at the edge side, and the initial model at the edge side was trained using the training sample gradients, as shown in Table 4. The parameters of all layers were updated in the training, where each experiment was repeated 10 times. The final results were averaged over the 10 experimental results, as shown in Table 6 and Figure 9.

Table 6. Experimental results of fine tuning's transfer-learning approach.

Number of Samples	Number of Training Sessions										
	1	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
20	0.1603	0.6050	0.6570	0.6564	0.6699	0.6753	0.6735	0.6751	0.6719	0.6722	0.6782
60	0.3162	0.7680	0.8144	0.8228	0.8320	0.8259	0.8266	0.8295	0.8314	0.8344	0.8373
120	0.5095	0.8478	0.8830	0.8991	0.9033	0.9046	0.9138	0.9101	0.9095	0.9141	0.9198
240	0.6679	0.9026	0.9320	0.9389	0.9479	0.9479	0.9552	0.9503	0.9504	0.9554	0.9534
620	0.7964	0.9630	0.9792	0.9811	0.9875	0.9882	0.9875	0.9878	0.9911	0.9905	0.9914
1000	0.8754	0.9803	0.9927	0.9929	0.9952	0.9957	0.9961	0.9973	0.9976	0.9970	0.9982

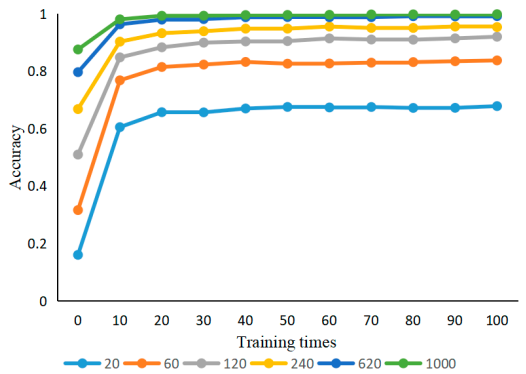


Figure 9. Experimental results of fine tuning's transfer-learning approach.

Next, the freeze-and train-transfer-learning approach was used for training, i.e., the model parameters transmitted from the cloud were loaded directly at the edge, and the model was trained using the training sample gradient, as shown in Table 4. Only the parameters of the fully connected layer were updated during the training. In this experiment, one fully connected layer and two fully connected layer were used, respectively, to achieve the final classification. Each experiment was repeated 10 times and the final results

were averaged over the 10 experimental results, as shown in Tables 7–9, and Figure 10. The experimental results of the above four experiments, with a small number of training samples, were intercepted separately and aggregated for comparative analysis; a summary of their results is shown in Table 9.

Table 7. Effect of using one fully connected layer.

Number of Samples	Number of Training Sessions										
	1	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
20	0.1499	0.5049	0.5502	0.5615	0.5653	0.5614	0.5683	0.5692	0.5677	0.5707	0.5710
60	0.3497	0.5899	0.6168	0.6188	0.6241	0.6283	0.6287	0.6295	0.6333	0.6323	0.6301
120	0.4482	0.6192	0.6414	0.6488	0.6482	0.6503	0.6534	0.6628	0.6619	0.6638	0.6630
240	0.5391	0.6451	0.6533	0.6555	0.6650	0.6670	0.6686	0.6683	0.6728	0.6734	0.6728
620	0.6149	0.6778	0.6884	0.6863	0.6906	0.6889	0.6925	0.6866	0.6793	0.6879	0.6899
1000	0.6244	0.6839	0.6920	0.6902	0.6954	0.6974	0.7005	0.7006	0.6960	0.6954	0.7001

Table 8. Effect of using two fully connected layers.

Number of Samples	Number of Training Sessions										
	1	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
20	0.1186	0.4106	0.4946	0.5430	0.5612	0.5692	0.5710	0.5746	0.5726	0.5781	0.5775
60	0.2808	0.5818	0.6207	0.6297	0.6350	0.6336	0.6356	0.6345	0.6368	0.6396	0.6368
120	0.3567	0.6329	0.6497	0.6543	0.6592	0.6657	0.6634	0.6627	0.6672	0.6655	0.6642
240	0.4277	0.6534	0.6691	0.6762	0.6698	0.6740	0.6790	0.6794	0.6765	0.6788	0.6781
620	0.5924	0.6817	0.6929	0.6954	0.7014	0.6961	0.6997	0.7021	0.7019	0.7019	0.7031
1000	0.6247	0.6902	0.6947	0.6981	0.7066	0.7055	0.7088	0.7141	0.7146	0.7159	0.7192

Table 9. Effect of using two fully connected layers.

Testing Method and Number of Samples	Number of Training Sessions								
	1	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Non-transferable learning20	0.0625	0.1725	0.2499	0.3660	0.4437	0.4766	0.4873	0.4855	0.5616
Non-transferable learning 60	0.0625	0.4048	0.6505	0.6863	0.7263	0.7485	0.7622	0.7469	0.7464
Non-transferable learning 120	0.0636	0.7028	0.7976	0.8377	0.8641	0.8753	0.8955	0.9028	0.9056
Fine tuning 20	0.1603	0.6050	0.6570	0.6564	0.6699	0.6753	0.6735	0.6751	0.6719
Fine tuning 60	0.3162	0.7680	0.8144	0.8228	0.8320	0.8259	0.8266	0.8295	0.8314
Fine tuning 120	0.5095	0.8478	0.8830	0.8991	0.9033	0.9046	0.9138	0.9101	0.9095
Freeze-and-train1 20	0.1499	0.5049	0.5502	0.5615	0.5653	0.5614	0.5683	0.5692	0.5677
Freeze-and-train1 60	0.3497	0.5899	0.6168	0.6188	0.6241	0.6283	0.6287	0.6295	0.6333
Freeze-and-train1 120	0.4482	0.6192	0.6414	0.6488	0.6482	0.6503	0.6534	0.6628	0.6619
Freeze-and-train2 20	0.1186	0.4106	0.4946	0.5430	0.5612	0.5692	0.5710	0.5746	0.5726
Freeze-and-train2 60	0.2808	0.5818	0.6207	0.6297	0.6350	0.6336	0.6356	0.6345	0.6368
Freeze-and-train2 120	0.3657	0.6329	0.6497	0.6543	0.6592	0.6657	0.6634	0.6627	0.6672

Table 9 shows the experimental results when using two fully connected layers.

The following conclusions were drawn from this experiment.

(1) The improvement in detection accuracy using transfer learning with a small number of training samples was significant.

Using transfer learning when the number of training samples per class was less than 120 significantly improved the prediction accuracy of the model. For example, when the number of personalized samples per class was only 20 and 60, prediction accuracies of 67% and 83% were obtained by using transfer learning; in contrast, without transfer learning, only 50% and 74% prediction accuracies, respectively, were achieved in the same cases.

(2) There was a significant reduction in training time using transfer learning with a small number of training samples.

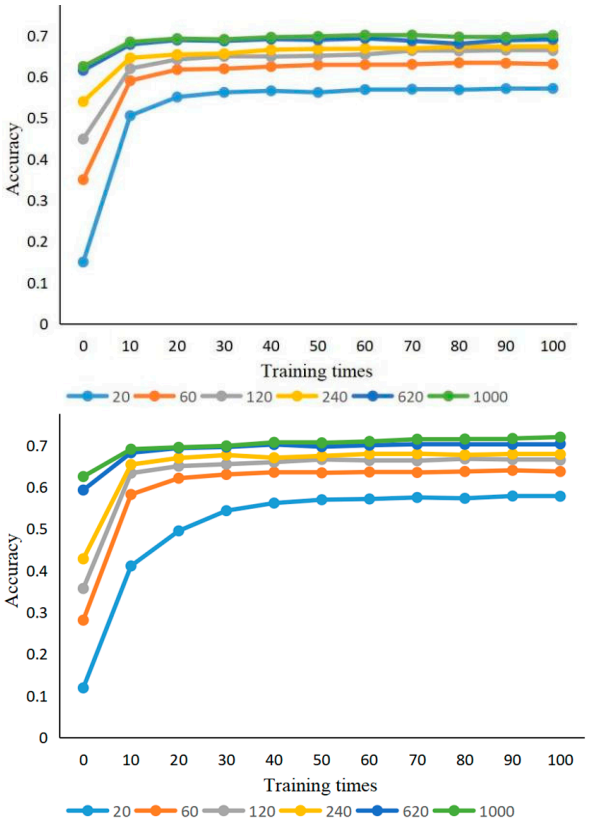


Figure 10. Experimental results when using one fully connected layer.

From the experiments, it can be seen that using transfer learning significantly reduced the number of training sessions of the model when the number of training samples per class did not exceed 240. In this experiment, when the number of samples per class at the edge was 20, 60, 120, and 240, the average time required for each training was 1.265 s, 2.075 s, 3.285 s, and 5.724 s, respectively. When the number of samples per class was only 20, it took less than 10 training sessions to achieve 90% of the optimal solution with transfer learning, while it took at least 40 sessions without transfer learning. This was a savings of more e than 75% of the training time. When the number of samples per class was only 60, it took less than 10 training sessions to achieve 90% of the optimal solution with transfer learning, while it took at least 30 sessions without transfer learning. This was a savings of more than 66% of the training time. The training times required to reach 90% of the optimal solution for sample sizes of l20 and 240 were also significantly reduced.

(3) The fine tuning transfer-learning approach in this task was more accurate than that of freeze–and-train.

The fine tuning approach in this defect recognition task always outperformed freeze-and-train by more than l0% in defect recognition accuracy when there were 20 or more labeled training images for each class of defects.

3.5.2. Exploring the Effect of Transfer-Learning Samples on Transfer Learning

The source domain for transfer learning in the above experiments was selected from a publicly available PV module defect dataset obtained on the web. The dataset contained samples of healthy PV modules, samples with cracks on the PV module surface, samples with broken grids on the PV module surface, samples with shadows on the PV module surface, and samples with hot spots on the PV module surface. However, the target domain that was migrated in this task contained only four types of samples: healthy PV modules, cracked PV modules, broken PV modules, and shaded PV modules. There were no samples of hot spot PV modules. In this experiment, we wanted to investigate the effect of the source domain sample categories on transfer learning. The complete training was performed with

the samples in the source domain in which the hot spots were removed from the PV modules and with the complete source domain samples. In order to ensure the reliability of the model's prediction accuracy, each experiment was repeated 10 times and the final results were averaged over the 10 experimental results, as shown in Figure 11.

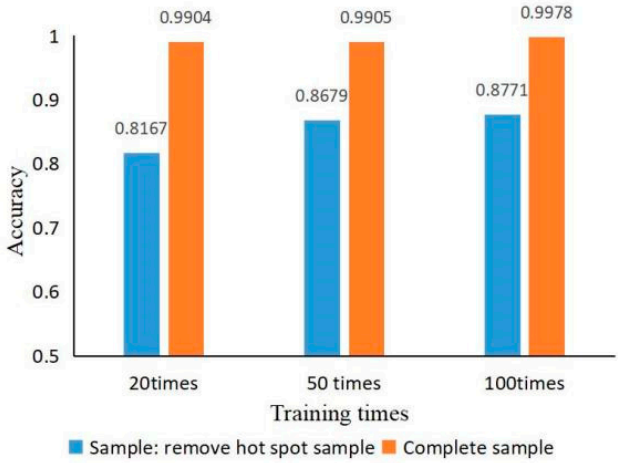


Figure 11. Relationship between source domain samples on the effect of transfer learning.

It is concluded from this experiment that in order to achieve a good transfer-learning effect, the aim in selecting transfer-learning samples should be to have as many samples with different defect performances in the source domain as possible, and it is desirable to make a fine and perfect classification of the samples. Therefore, the dataset of PV modules with different working conditions, stored in the cloud in this task, can enhance the effect of transfer learning.

4. Conclusions

In this paper, a cloud-edge collaborative method for detecting defects in PV modules, based on the improved YOLO v3-tiny technique, was proposed to achieve real-time detection of defects in PV modules. The defect data were produced for the main surface defects in PV modules and used in YOLO v3-tiny, YOLO v3, and improved YOLO v3-tiny models. The experimental results showed that the models used in this paper had higher detection accuracy and lower detection time, and they maintained high detection accuracy under different input sizes, indicating that they had high robustness. The detection model was continuously trained using a large amount of sample data that was stored in the cloud to generate a generic detection model that can be applied to the task of detecting defects in PV modules. The trained generic model was then migrated to the cloud edge by means of transfer learning, where only a small number of local samples need to be fine-tuned to obtain a personalized model for detecting PV module defects. This model can be used for the real-time detection of PV module defects. At the same time, the personalized samples at the cloud edge were periodically uploaded to the cloud as training data for updating the generic detection model. Through experimental validation and comparative analysis, we showed that the training of the detection model can be realized in the cloud using this method of PV module defect detection, and only a small amount of personalized adjustment training is needed at the edge, which can save a great deal of training time. At the same time, as the training samples were insufficient, the accuracy of defect detection can be significantly improved. Finally, this method showed good results in terms of real-time application, accuracy, and sample limitation.

Author Contributions: Conceptualization, H.S.; data curation, Y.W.; formal analysis, H.W.; funding acquisition, P.T. and F.D.; investigation, F.L.; methodology, H.W.; project administration, F.D.; resources, F.L.; software, W.M.; validation, W.M. and Y.Z.; writing—original draft, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Project of the Natural State Grid Corporation of China (SGHEDK00DYJS2000044), the Natural Science Foundation of China (52167008, 51867010), the Outstanding Youth Fund Project of the Jiangxi Natural Science Foundation (20202ACBL214021), the Key Research and Development Plan of Jiangxi Province (20202BBGL73098), and the Science and Technology Project of the Education Department of Jiangxi Province (GJJ210650).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Polman, A.; Knight, M.; Garnett, E.C.; Ehrler, B.; Sinke, W.C. Photovoltaic materials: Present efficiencies and future challenges. *Science* **2016**, *352*, aad4424. [CrossRef]
2. Dhimish, M.; Mather, P.; Holmes, V. Novel photovoltaic hot-spotting fault detection algorithm. *IEEE Trans. Device Mater. Reliab.* **2019**, *19*, 378–386. [CrossRef]
3. Lamb, W.P.; Asnes, D.E.; Kupfer, J.; Lickey, E.; Bakken, J.; Haskell, R.C.; Saeta, P.N.; Yang, Q. Real-Time Anticipation and Prevention of Hot Spots by Monitoring the Dynamic Conductance of Photovoltaic Panels. *IEEE J. Photovolt.* **2022**, *12*, 1051–1057. [CrossRef]
4. Wang, J. Research on the Application of Machine Learning in Photovoltaic Power Station. In Proceedings of the 2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST), Guangzhou, China, 10–12 December 2021; pp. 1927–1930.
5. Sizkouhi, A.M.M.; Esmailifar, S.M.; Aghaei, M.; De Oliveira, A.K.V.; Rüther, R. Autonomous path planning by unmanned aerial vehicle (UAV) for precise monitoring of large-scale PV plants. In Proceedings of the 2019 IEEE 46th Photovoltaic Specialists Conference (PVSC), Chicago, IL, USA, 16–21 June 2019; pp. 1398–1402.
6. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
7. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 142–158. [CrossRef] [PubMed]
8. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
10. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Gong, H.; Li, H.; Xu, K.; Zhang, Y. Object detection based on improved YOLOv3-tiny. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; pp. 3240–3245.
13. Jiang, Z.; Zhao, L.; Li, S.; Jia, Y. Real-time object detection method based on improved YOLO v4-tiny. *arXiv* **2020**, arXiv:2011.04244.
14. Mujahid, A.; Awan, M.J.; Yasin, A.; Mohammed, M.A.; Damaševičius, R.; Maskeliūnas, R.; Abdulkareem, K.H. Real-time hand gesture recognition based on deep learning YOLOv3 model. *Appl. Sci.* **2021**, *11*, 4164. [CrossRef]
15. Zhong, Y.-S.; Xu, L.-H.; Zhou, K. Research on defect detection of photovoltaic modules based on improved SSD algorithm. *Microprocessor* **2022**, *1*, 22–25.
16. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 687–694.
17. Gao, L.; Liu, T.; Cao, T.; Hwang, Y.; Radermacher, R. Comparing deep learning models for multi energy vectors prediction on multiple types of building. *Appl. Energy* **2021**, *301*, 117486. [CrossRef]
18. Lin, L.; Gao, L.; Kedzierski, M.A.; Hwang, Y. A general model for flow boiling heat transfer in microfin tubes based on a new neural network architecture. *Energy AI* **2022**, *8*, 100151. [CrossRef]
19. Wang, L.; Chen, Z.; Hua, D.; Zheng, Z. Semantic segmentation of transmission lines and their accessories based on UAV-taken images. *IEEE Access* **2019**, *7*, 80829–80839. [CrossRef]
20. Hu, Z.; He, T.; Zeng, Y.; Luo, X.; Wang, J.; Huang, S.; Lin, B. Fast image recognition of transmission tower based on big data. *Prot. Control. Mod. Power Syst.* **2018**, *3*, 1–10. [CrossRef]
21. Luo, H.; Wu, Q.; Chen, K.; Peng, Z.; Fan, P.; Hu, J. Power Equipment Defect Detection Algorithm Based on Deep Learning. In *Artificial Intelligence in China*; Springer: Singapore, 2021; pp. 211–219.
22. Carletti, V.; Greco, A.; Saggese, A.; Vento, M. An intelligent flying system for automatic detection of faults in photovoltaic plants. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 2027–2040. [CrossRef]
23. Tao, J.; Tang, J.; Ji, Y.; Zhang, N. Application of Cloud Edge Collaboration Architecture in Power IoT. In Proceedings of the 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 6–8 November 2020; Volume 1, pp. 18–22.

24. Song, D.; Yuan, L.; Zhao, W.; Yu, Q.; Du, J.; Pan, K. Cloud-Edge Computing Resource Collaborative Optimization Method for Power Distribution Fault Analysis Service. In Proceedings of the 2021 China International Conference on Electricity Distribution (CICED), Shanghai, China, 7–9 April 2021; pp. 627–632.
25. Chang, L.; Chen, Y.T.; Hung, M.H.; Wang, J.H.; Chang, Y.L. Yolov3 based ship detection in visible and infrared images. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 3549–3552.
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
27. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
28. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
29. Liu, Z.-D.; Chen, Z.-Y.; Zhou, J.; Yang, Z.-K.; Shi, H.-R. Deep confidence network-based defect detection method for solar photovoltaic panels. *Mech. Des. Manuf. Eng.* **2021**, *5*, 61–64.