Article

ResnetCPS for Power Equipment and Defect Detection

Xingyu Yan [1,2], Lixin Jia [1,2], Xiao Liao [3], Wei Cui [3], Shuangsi Xue [1,2], Dapeng Yan [1,2,]* and Hui Cao [1,2]

1 The Shaanxi Key Laboratory of Smart Grid, Xi'an Jiaotong University, Xi'an 710049, China; yxyxjtu@stu.xjtu.edu.cn (X.Y.); lxjia@mail.xjtu.edu.cn (L.J.); xssxjtu@stu.xjtu.edu.cn (S.X.); huicao@mail.xjtu.edu.cn (H.C.)

2 The State Key Laboratory of Electrical Insulation and Pocross-Layerwer Equipment, School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China

3 State Grid Information & Telecommunication Group Co., Ltd., Beijing 100031, China; liaoxiao@sgitg.sgcc.com.cn (X.L.); cuiwei@sgitg.sgcc.com.cn (W.C.)

* Correspondence: dapengyan@xjtu.edu.cn

Abstract: Routine visual inspection is fundamental to the preventive maintenance of power equipment. Convolutional neural networks (CNNs) substantially reduce the number of parameters and efficiently extract image features for classification tasks. In the actual production and operation process of substations, due to the limitation of safety distance, camera monitoring, inspection robots, etc., cannot be very close to the target. The operational environment of power equipment leads to scale variations in the main target and thus compromises the performance of conventional models. To address the challenges posed by scale fluctuations in power equipment image datasets, while adhering to the requirements for model efficiency and enhanced inter-channel communication, this paper proposed the ResNet Cross-Layer Parameter Sharing (ResNetCPS) framework. The core idea is that the network output should remain consistent for the same object at different scales. The proposed framework facilitates weight sharing across different layers within the convolutional network, establishing connections between pertinent channels across layers and leveraging the scale invariance inherent in image datasets. Additionally, for substation image processing mainly based on edge devices, smaller models must be used to reduce the expenditure of computing power. The Cross-Layer Parameter Sharing framework not only reduces the overall number of model parameters but also decreases training time. To further enhance the representation of critical features while suppressing less important or redundant ones, an Inserting and Adjacency Attention (IAA) module is designed. This mechanism improves the model's overall performance by dynamically adjusting the importance of different channels. Experimental results demonstrate that the proposed method significantly enhances network efficiency, reduces the total parameter storage space, and improves training efficiency without sacrificing accuracy. Specifically, models incorporating the Cross-Layer Parameter Sharing module achieved a reduction in the number of parameters and model size by 10% to 30% compared to the baseline models.

Keywords: power system; substation; ResNet; cross-layer parameter sharing

Citation: Yan, X.; Jia, L.; Liao, X.; Cui, W.; Xue, S.; Yan, D.; Cao, H. ResnetCPS for Power Equipment and Defect Detection. *Appl. Sci.* **2024**, *14*, 10578. https://doi.org/10.3390/app142210578

Academic Editor: João M. F. Rodrigues

Received: 30 September 2024
Revised: 9 November 2024
Accepted: 12 November 2024
Published: 16 November 2024

1. Introduction

During the long-term operation of power equipment, various factors such as environmental and load changes may cause scratches, deformations, cracks, rusting, overheating, discoloration, cracking, and other problems on the appearance of the equipment. These issues may be external manifestations of internal equipment failures. By inspecting the appearance of the equipment, these problems can be detected. The faulty equipment could be repaired or replaced in a timely manner to avoid production safety losses caused by the expansion of the fault. Relying on various monitoring and inspection equipment instead of artificial vision to identify safety hazards is the key to intelligent substations [1–5].

In recent years, with the rapid development of computer vision technology, deep learning has made significant progress in visual applications in the electrical engineering industry. Deep learning can automatically learn and extract feature information from images by constructing deep neural network models, thereby achieving efficient processing and analysis of images. In the electrical engineering industry, deep learning visual applications are mainly focused on equipment fault diagnosis, intelligent inspection, safety monitoring, and other aspects, providing strong support for improving the safety and reliability of power systems, such as insulators [6,7], bird nests [8], and dials [9,10]. Ref. [11] proposed an effective multi-task defect classifier. By constructing a CNN model, automatic extraction and recognition of fault features can be achieved. Compared with traditional fault diagnosis methods, deep learning-based image recognition methods have higher accuracy and efficiency, and can quickly locate the fault location and provide corresponding repair suggestions.

Reviewing early developments of image recognition, we found that cleverly utilizing image properties for parameter sharing is the core of the development of convolutional neural networks. With the continuous improvement of computing power, the convolutional neural network gradually develops into a super-extensive network with deeper layers and more parameters. The ViT model based on attention mechanism transformer architecture provides a solution to the depth vision problem other than convolutional neural networks [12]. Aside from that, vision models based on transformer have emerged endlessly, including Swin Transformer [13], Deit [14], crossvit [15], DaViT [16], PVT [17], CaiT [18], Focal transformer [19], etc. Swin Transformer is a modified version of Vision Transformer using shifted windows (SWs) [13]. It divides the fixed-size sampling block in Vision Transformer into different-sized blocks (windows) according to the hierarchy, and the information between each block is not shared. DaViT proposed a dual attention architecture, which includes spatial attention, and proposes channel tokens to introduce channel attention [16]. The Multi-Layer Perceptron (MLP) Mixer model, which uses MLP to replace Conv in traditional CNN and self-attention in the transformer, is proposed [20]. It uses the most traditional MLP as the underlying architecture and obtains results without input from the transformer. Recently, networks based on MLP architecture have been proposed, including gMLP [21], ResMLP [22], and cycleMLP [23]. Compared with CNN, the ViT model ignores two basic properties of the image: translation invariance and rotation invariance. It learns and trains through more data to obtain the correlation between the blocks in the image, which is called self-attention.

The essence of the convolutional neural network is a parameter-sharing strategy based on image properties. The principle of parameter sharing is based on the assumptions of neural networks for specific problems. The training efficiency is improved by changing the model structure. A large number of parameters are reduced without restricting the generalization ability of the model. Effective parameter sharing limits the number of parameters and improves the training efficiency of the model. Therefore, convolutional neural networks can efficiently train to obtain good accuracy in the case of limited datasets and computing power. These parameters are often shared within the same layer. According to the development track of deep learning, the fewer restrictions on neural network structure, the more accurate the model. For example, the transformer model only assumes that there is self-attention between windows. The MLP model will further improve the model's accuracy, and it does not use parameter sharing. However, these models often require more accurate datasets and higher computing power.

For most public datasets, objects in the image are clear and distinguishable, occupying the center of the image with their boundaries exactly at the edges of the image. For images of power equipment, due to limitations such as working environment and shooting angle, the captured object may not be accurately located at the center of the image. It may only occupy a small part of the image. For such images, the model should perform the same operation on objects of different scales. This is the scale invariance of datasets. This article proposes a ResNet variant named ResNet with Cross-Layer Parameter Sharing (ResnetCPS) based on the scale invariance of image datasets. For parameter modules with the same number of channels and high repetition rate, they share some parameters to reduce parameter coupling and parameter count. Cross-Layer Parameter Sharing utilizes information from multiple convolutional neural network layers to optimize model performance. The difference between this method and traditional cellular neural networks is that weights are shared between different network layers. It encourages layers to learn from each other instead of focusing on optimizing individual layers, resulting in more efficient models. It also makes the network more robust to changes in training data, as it is more resilient to the effects of random perturbations. Cross-Layer Parameter Sharing can improve the accuracy of processing power equipment datasets. Meanwhile, parameter sharing without changing the structure reduces the total number of parameters and decreases the storage space for parameters. Shared parameters during training are trained multiple times in the same backpropagation process, resulting in higher training efficiency. In addition, increasing the information exchange between shared and non-shared parameters, enhancing the ability to represent important features, and using channel self-attention mechanism could improve the overall performance of the model are important.

The main contributions of this paper are as follows:

(1) A Cross-Layer Parameter Sharing mechanism was designed to address the issue of non-prominent data subjects in power equipment image data.

(2) An Inserting and Adjacency Attention Module was designed based on the sharing mechanism. The "inserting" action augments the communication among fragmented channels, and simultaneously, this action curtails the size of the model and enhances the efficiency.

(3) The overall architecture of the ResNetCPS model was designed and its effectiveness on an original substation dataset was validated.

The rest of this article is organized as follows. Section 2 introduces related work. The details of the proposed methods are presented in Section 3. The execution details of the experiment, analysis of the results, and the design and results of the ablation experiment are given in Section 4. Finally, Section 5 concludes this article.

2. Related Work

2.1. ResNet

Residual Neural Network (ResNet) solves the problem of gradient vanishing and exploding during the training process of deep neural networks by introducing the concept of residual learning, enabling deeper networks to be trained and optimized [24]. Its ideas have been applied to more network designs. Based on ResNet, there are many improved networks. ResNet-B/C/D replaces some original convolutional kernels, reducing the computational and parameter burden of models while maintaining high accuracy [25]. Among them, ResNet-b shifts the downsampling in the residual difference to avoid information loss. ResNet-c replaces the original 7×7 convolutional kernel with three 3×3 convolutional kernels, reducing the number of parameters. ResNet-d adopts group convolution based on ResNet-b. Wide ResNet (WRN) is a variant that enhances network width by increasing the number of ResNet channels (i.e., increasing the number of output channels per convolutional layer) [26]. It improves the feature extraction ability of the network. ResNeXt introduced the concept of "cardinality", using group convolution to control the number of packets by introducing cardinality [27]. In the Split Transform Merge strategy, 1×1 convolution achieves low-dimensional embedding, such as by converting 256 channels into 4 channels, with 32 branches (cardinality = 4). This grouping convolution can effectively reduce the number of model parameters and accelerate model training. This structure can replace the method of increasing network depth to a certain extent. NFNet is a ResNet-based network that does not require Batch Normalization, and its core is AGC (Adaptive Gradient Clipping) technology [28]. AGC technology dynamically adjusts thresholds based on gradient information during network training that can train non-normalized networks with larger batches and large-scale data augmentation to achieve more effective gradient clipping. It can avoid the limitation of fixed thresholds in Batch Normalization and improve the performance and training speed of the model. Res2Net has modified the block in ResNet to extract multiscale features further [29]. Resnetrs has improved its expansion strategy by expanding depth in settings where overfitting occurs and gradually increasing image resolution [30]. This strategy increases training speed.

2.2. Channel Self-Attention Module

SENet (Squeeze and Extraction Network) introduces an attention mechanism on the output of each ResNet block, explicitly modeling the interdependence between feature channels [31]. The Squeeze operation compresses each two-dimensional feature channel into an actual number with a global receptive field. The Extraction operation adjusts the importance of each feature channel through a learnable weight. By learning the weights of different channels, the network's attention to essential features has been improved. ECANet (Efficient Channel Attention Network) adopts a more efficient method to learn channel attention [32]. It uses a 1D convolutional layer to convolve the features of each channel and then uses the sigmoid function to learn channel attention. It can significantly reduce computational complexity and improve model performance to a certain extent.

2.3. Lightweight Model

Through more robust sharing and connection mechanisms, networks can be cleverly constructed and lightweight models can be designed, such as MobileNets [33–35], ShuffleNets [36,37], ChannelNet [38], FBNet [39], GhostNet [40], MicroNet [41], TinyNet [42], TVConv [43], and Fasternet [44]. The core of Mobilenet is depthwise-separable convolution, which decomposes traditional convolution by first using a depthwise convolution for spatial convolution and then a pointwise convolution for channel convolution [33]. The core of ShuffleNet is channel shuffling [36]. The feature maps of different groups are rearranged, thereby increasing the information interaction between different groups. ChannelNets include channelwise group convolutions, channelwise-based depthwise-separable convolutions, and unweighted convolution classification layers [38]. The three components are used to compress different parts of the network. FBNet adopts the JointNAS method, using both hyperparameters and training strategies as the search space, making the network design more comprehensive and optimized [39]. In GhostNet [40], "Ghost Module" uses a small number of convolution kernels to generate some original feature maps and then causes more "ghost" feature maps through simple linear transformations. These ghost feature maps have similar characteristics to the original ones, but the computational complexity is significantly reduced. MicroNet redesigns the network structure of the convolutional layer, reducing the connectivity between network nodes without reducing the width of the network. Micro-factorized convolution combines pointwise convolution in MobileNet with depthwise convolution, decomposed into a low-rank matrix, thereby achieving a good balance between the number of channels and the connectivity of input and output [41]. TVConv proposes Affinity Maps to distinguish multiple different local features [43]. It can implicitly capture the semantic features of different spatial regions. Fasternet proposes a new partial convolution (PConv) that can extract spatial features more efficiently by simultaneously reducing redundant computation and memory access [44]. It runs significantly faster than other networks on a variety of devices. Ref. [45] proposed an improved lightweight convolutional neural network based on U-Net.

3. Method

3.1. Overview

The structure of ResnetCPS152 with a ratio of 1/4 is illustrated in Figure 1. The model initially employs multiple 3×3 convolution processes, transforming 3-channel images into 64-channel feature maps. Subsequently, the model comprises four stages. Within each stage, there are two paths. One path traverses a bottleneck comprising 1×1, 3×3, and 1×1 convolutional layers, followed by an Inserting and Adjacency Attention (IAA) module. The first convolutional layer features a stride of 2. The other path includes a 2×2 average pooling layer and a 1×1 convolutional layer. The values from both paths are summed before being fed into the repeating blocks. Each block includes both a shortcut and a bottleneck. Unlike the bottleneck in earlier stages, all convolutional layers within this block have a stride of 1. In stages two and three, the repeating blocks incorporate shared layers. The shared and normal layers process the preceding feature maps, which are then concatenated. The sharing mechanism is centered around repeating blocks as its fundamental unit.

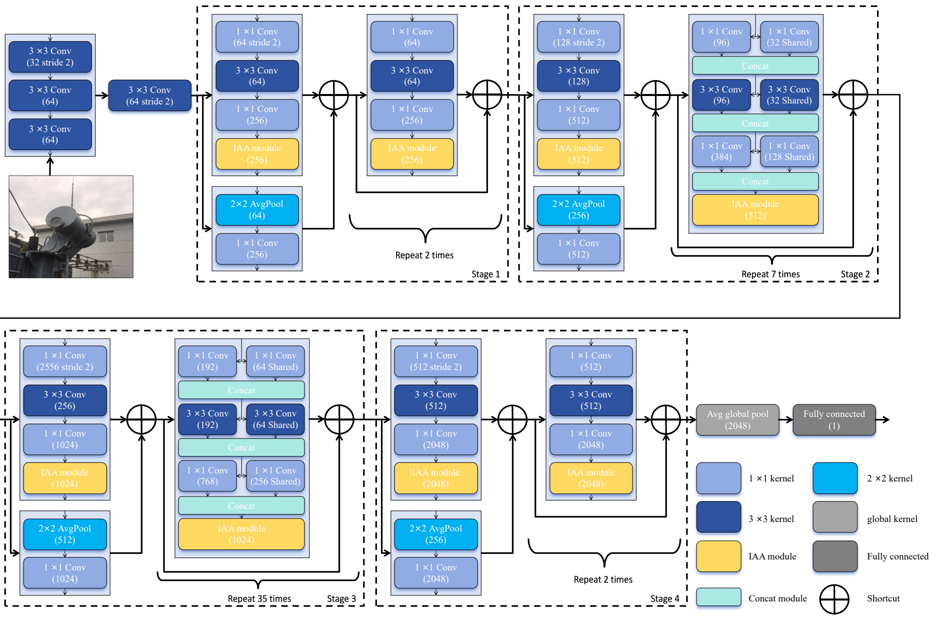


Figure 1. The structure of ResnetCPS152 with ratio = 1/4.

3.2. *Convolution Operations with Cross-Layer Parameter Sharing*

The equation for the convolution operation can be expressed as follows:

$$c_{i,j,k}^{(l)} = \sum_{m=1}^{F} \sum_{n=1}^{F} w_{m,n,k}^{(l)} a_{(i-\frac{F+1}{2})\times s+m,(j-\frac{F+1}{2})\times s+n}^{(l-1)} \tag{1}$$

where $c_{i,j,k}^{(l)}$ is the convolution result of the i-th row and j-th column and k-th kernel in the l-th layer, $w_{m,n,k}^{(l)}$ is the weight of the k-th convolution kernel's m-th row and n-th column, and $a_{(i-\frac{F+1}{2})\times s+m,(j-\frac{F+1}{2})\times s+n,k}^{(l-1)}$ is the value of the k-th feature map at position $((i-\frac{F+1}{2})\times s+m,(j-\frac{F+1}{2})\times s+n)$ in the $(l-1)$-th layer. F is the size of the convolutional kernel, and s is the stride size, representing the step size at which the convolutional kernel slides on the input feature map. Consider bias and concatenate the outputs to obtain the output of that layer.

$$z_{i,j}^{(l)} = b^{(l)} + \sum_{k=1}^{K^{(l)}} c_{i,j,k}^{(l)} \tag{2}$$

where $z_{i,j}^{(l)}$ is the convolution result of the i-th row and j-th column in the l-th layer, $b^{(l)}$ is the bias term, and $K^{(l)}$ is the number of convolution kernels in the l-th layer.

We define shared convolution as the partial convolution kernels between duplicate layers or blocks being the same, with the same parameters.

The resulting output is also concatenated from the individual outputs of each convolutional kernel. Equation (1) has been rewritten as follows:

$$\begin{aligned} z_{i,j}^{(l)} &= b^{(l)} \\ &+ \sum_{k=1}^{K^{(l)}} \sum_{m=1}^{F} \sum_{n=1}^{F} w_{m,n,k}^{(l)} a_{(i-\frac{F+1}{2})\times s+m,(j-\frac{F+1}{2})\times s+n}^{(l-1)} \\ &+ \sum_{k=K^{(l)}}^{K^{(d)}} \sum_{m=1}^{F} \sum_{n=1}^{F} w_{m,n,k}^{(d)} a_{(i-\frac{F+1}{2})\times s+m,(j-\frac{F+1}{2})\times s+n}^{(l-1)} \end{aligned} \tag{3}$$

where $K^{(d)}$ is the number of convolution kernels in the d-th layer of the duplicate blocks, and $w_{m,n,k}^{(d)}$ is the weight of the k-th convolution kernel's m-th row and n-th column in the d-th layer of the duplicate blocks.

Figure 2 shows the repetitive structure of the 4th stage of the 152-layer ResnetCPS. It contains a bottleneck block repeated 35 times, marked as a black dashed rounded box. Each block is composed of a repeated three-layer network, labeled as a gray solid line rounded box, consisting of a 1×1 convolutional layer for 256 channels, a 3×3 convolutional layer for 256 channels, and a 1×1 convolutional layer for 1024 channels. When the sharing ratio is $1/4$, each layer corresponds to 64, 64, and 256 channel convolutions as shared convolutions, marked with green rounded boxes. These convolution parameters distributed in different blocks correspond to the same channel, and they calculate a common gradient during training while being modified. The remaining convolution parameters are labeled with rounded boxes of different colors, and their respective gradients are computed during training. In forward calculation, the convolutional kernels of each layer are combined and calculated forward on a layer-by-layer basis. Each layer of its repeating structure has a portion of the shared parameter part that shares parameters. In forward propagation, the shared parameter part can participate in the calculation repeatedly. Shared parameter parts distributed in different layers will be simultaneously calculated and updated when calculating gradients in reverse. The proportion of this part is set as hyperparameter r in the article. The remaining components are unique convolutional kernels for each structure, which generally participate in gradient calculations.

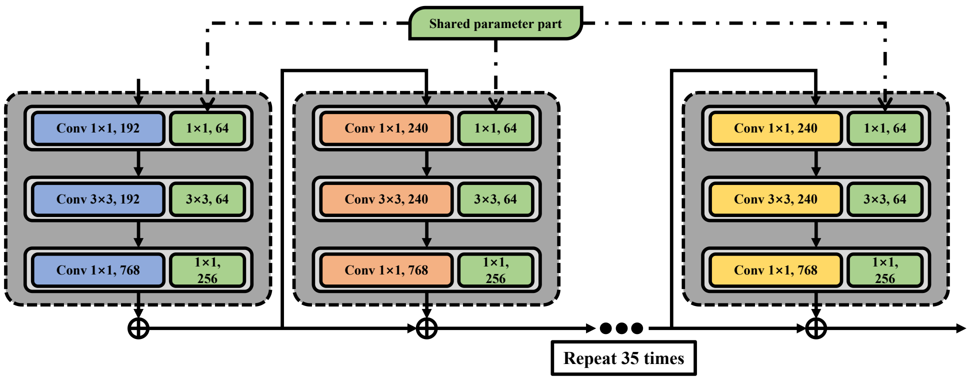


Figure 2. The 4th stage of the 152-layer ResnetCPS.

3.3. Shared Layer Complexity

The time complexity of repeating blocks can be represented as follows:

$$TIME \sim \mathcal{O}(N \times M^2 \times K^2 \times C^2) \tag{4}$$

where N is the number of repetitions in the convolution blocks, M is the size of the feature map, K is the size of the convolution kernel, and C is the number of convolution kernels in each layer, which is also the number of channels in the feature map. This formula represents the number of convolution operations in each layer.

If no shared units are added to the repeated blocks, its spatial complexity can be represented as follows:

$$SPACE \sim \mathcal{O}(N \times K^2 \times C^2) \tag{5}$$

This equation represents the number of parameters for each convolutional kernel layer.

We use time complexity divided by spatial complexity to represent the number of calculations a single parameter participates in during each forward propagation of the model. The number of calculations without adding shared units is as follows:

If a shared unit is added to a repeated block, its spatial complexity can be represented by the following equation:

$$SPACE^* \sim \mathcal{O}(N \times K^2 \times (C - C_{share})^2 + K^2 \times C_{share}^2) \tag{6}$$

where C_{share} is the number of shared convolution kernels in each layer.

$$C_{share} = \alpha C \tag{7}$$

In repeated blocks, regardless of whether shared units are added or not, the time complexity remains unchanged, that is, the number of convolution operations in each layer remains unchanged. According to the definition, the time complexity of the convolutional layer divided by the spatial complexity can calculate the average number of times the convolutional kernel participates in convolution operations during forward propagation.

$$\begin{aligned} Count &= \frac{TIME}{SPACE} \\ &\sim \mathcal{O}(\frac{N \times M^2 \times K^2 \times C^2}{N \times K^2 \times C^2}) \sim \mathcal{O}(M^2) \end{aligned} \tag{8}$$

If a shared unit is added to a repeated block, the number of times a single parameter participates in the forward propagation of the model is as follows:

$$\begin{aligned} Count^* &= \frac{TIME}{SPACE^*} \\ &\sim \mathcal{O}(\frac{N \times M^2 \times K^2 \times C^2}{N \times K^2 \times C^2 \times (1-\alpha)^2 + K^2 \times \alpha^2 C^2}) \\ &\sim \mathcal{O}(\frac{N \times M^2}{(1-\alpha)^2 N + \alpha^2}) \end{aligned} \tag{9}$$

Parameters can affect the size of memory occupied by the model. The more parameters there are, the larger the final model will be. It can be seen that adding shared units reduces the complexity of the model space. The number of times α parameter participates in forward propagation of the model will affect the updating intensity of the parameter during error backpropagation. In other words, the more calculations it participates in, the stronger the error backpropagation. Through comparison, it can be seen that adding a shared unit will increase the number of calculations that a single parameter of the model participates in. Therefore, the average number of updates for each parameter will be more frequent. This is different from simply increasing the training step size, which can effectively improve training efficiency. The formula indicates that two parameters will have an impact on the scale and efficiency of the model. The impact of normalization layer, activation layer, short links, etc., on model performance is not coupled with shared units, so they are ignored when analyzing complexity. They also participate in training and computation, which weakens the effectiveness of parameter participation in training.

3.4. Inserting and Adjacency Attention Module

The IAA module is a channel self-attention module, as shown in Figure 3. The blue part in the feature map represents the output of the normal convolution kernel, and the red part represents the output of the shared convolution kernel. After a global average pooling, the feature map is compressed into a $1 \times 1 \times C$ channel feature vector. The shared channel feature vector is inserted into the normal channel feature vector with an interval of the reciprocal of the sharing ratio α. If $\alpha = 1/4$, then a shared feature value is inserted into every three normal channel feature vectors. Then, adjacency convolution is performed, which is a one-dimensional convolution with a kernel length of $1/\alpha + 1$. After a layer of convolution, the inserted channel is extracted to the initial position. During this process, keeping the dimension of the channel feature vector unchanged, channel-based self-attention is formed. Finally, the self-attention weight is multiplied by the initial feature map to scale the feature map and form the final output. The smaller the sharing ratio, the longer the kernel length. In extreme cases, the sharing ratio is zero, the one-dimensional convolution layer becomes a fully connected layer, and the IAA module degenerates into an SE module.

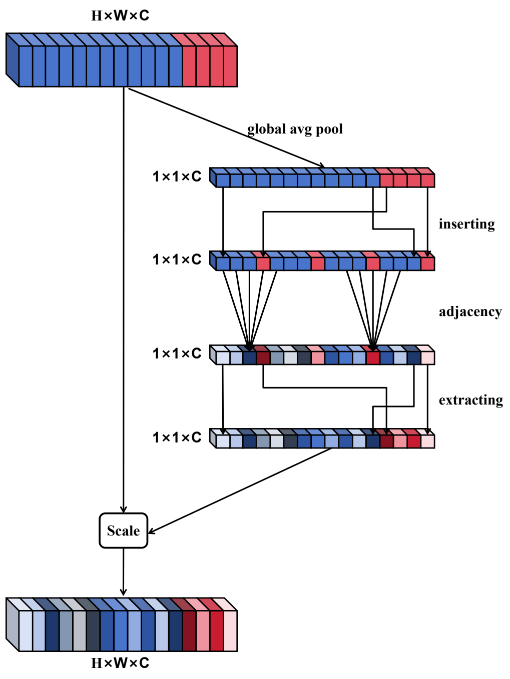


Figure 3. IAA module structure and operation flowchart.

4. Experiment

4.1. Implementation Details

The experiment was performed on a server with Intel® Xeon® gold 6136 CPU and NVIDIA Corporation GP102 [TITAN Xp]. The equipment was sourced from Synology, Shanghai, China.The experiment was validated on the embedded development platform, NVIDIA Jetson TX2, with an NVIDIA Denver dual-core 64-bit CPU and 256 NVIDIA® CUDA® core Pascal™ Architecture. The equipment was sourced from NVIDIA, Santa Clara, U.S.A. The system version is Ubuntu 18.04, the CUDA version is 11.1, the Python version is 3.8.1, and the Pytorch version is 1.7.0.

The optimizer is set to Momentum with a 0.9 coefficient. There are 200 training epochs with 10 warm-up epochs. During the experiment, when the number of training epochs was set to 200, the model could reach a relatively stable convergence state within a reasonable time in most cases, that is, the loss value no longer decreased significantly and the accuracy also tended to be stable. The choice of batch size was mainly based on two principles. First, the batch size should be as large as possible. Second, it should not exceed the video memory capacity. For example, for smaller models (such as resnetcps50 and resnetcps101), we set the batch size to 64, which could fully utilize the computing resources and improve the training efficiency. For larger models (such as resnetcps200 and resnetcps270), in order to avoid memory problems, we adjusted the batch size to 32. In this way, we could find a better balance between memory usage and computing efficiency in the training of different models, ensuring that the models could be trained smoothly and achieve good performance. And according to the different batch sizes, the learning rate was adjusted. The larger the batch size, the larger the learning rate. For the coefficient of the Momentum optimizer, we adjusted it through experiments. The larger the coefficient value, the more information from the previous gradients can be retained, but it may also lead to overly aggressive updates and the optimal solution being missed. We tried different coefficient values from 0.1 to 0.99. When the coefficient was set to 0.1, the model converged slowly and required more training epochs to achieve good performance. When the coefficient increased to 0.9, the model could quickly reduce the loss value in the early stage of training and showed good stability throughout the training process.

4.2. Dataset

As shown in Figure 4, the dataset consists of pictures of redundant objects or equipment states commonly found in substations. This article refers to it as a substation dataset. The dataset contains various objects, including 2787 dials, 728 nests, 941 oil pollution, 5209 press plates, 7290 insulators, 1717 silica gels, and 988 tanks. Dials are helpful for monitoring the operating parameters of equipment, and any abnormality can affect fault judgment. Press plates are related to the reliability of electrical connections. The failure of silica gel affects the humidity inside the equipment. Problems with tanks affect insulation and heat dissipation. Oil pollution reduces insulation and corrodes equipment. Nests are prone to causing short-circuit faults. Object classification in substations is essential for the smart grid because it allows faster and more efficient detection of faults in the power grid. It helps reduce downtime on the grid and helps increase the reliability of the energy supply. It can help maximize energy efficiency by detecting potential inefficiencies before they become problematic. Currently, there is no similar public dataset for substation or power equipment defect classification dataset.

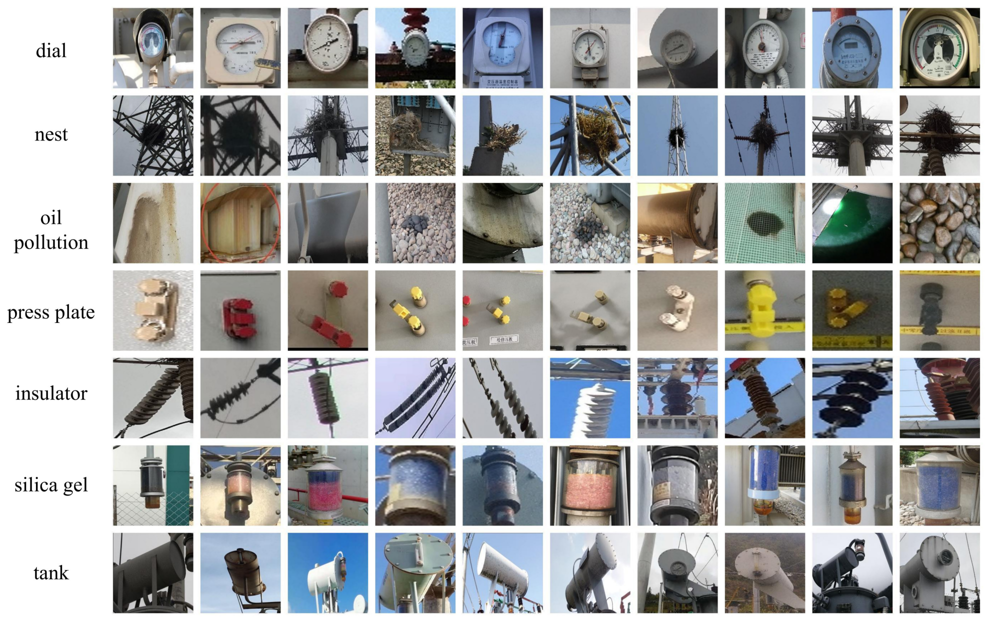


Figure 4. Substation object dataset.

Target labeling calculates the proportion of the corresponding target to the image. The ratio is divided into 10 levels. Each 10% is a gear, and the ratio from 0 to 100% is used for statistics. The statistical histogram of the proportion of the target object is shown in

Figure 5. For some high outdoor targets, such as bird's nests and some dials, most of the pictures are taken from a distance, and the proportion is often lower. In contrast, for some oversized or indoor equipment, most pictures are taken at close range, and the proportion is often higher. Overall, most of the image targets are concentrated in 20–80%. There are even some images with a target ratio of 10–15%. Most of the target objects are not the main body of the pictures.

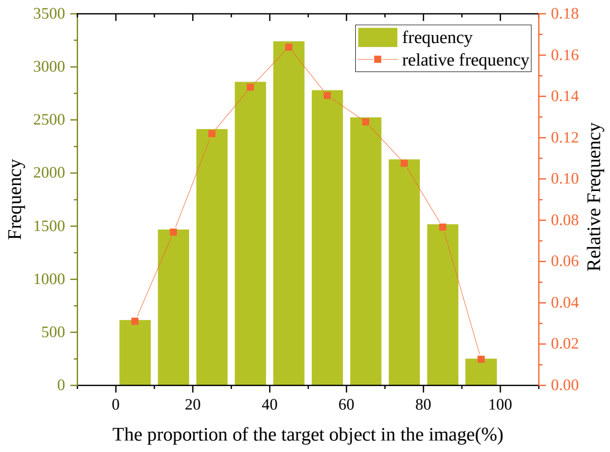
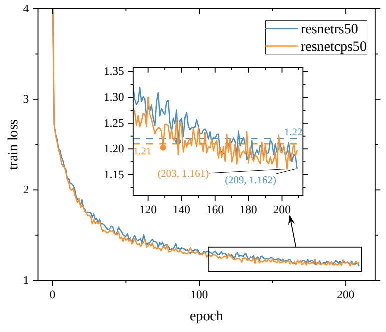


Figure 5. Histogram of the proportion of the target object in the image.

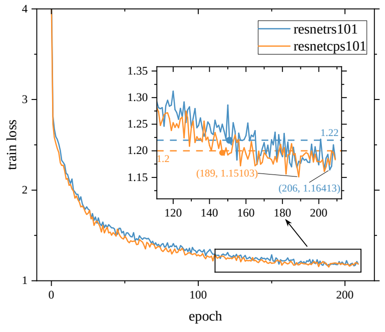
4.3. Training Process

The training process of the resnetcps model is set as the experimental subject to evaluate the Cross-Layer Parameter Sharing module. Five models with increasing layers of resnetcps are trained, including resnetcps50, resnetcps101, resnetcps152, resnetcps200, and resnetcps270. As shown in Figure 6, the fluctuation line represents the training curve of the model, with blue representing the resnetrs model and orange representing the resnetcps. The sharing ratio in this experiment was set to 1/4. We mainly measure whether the model training has a faster convergence speed and the lowest loss value. The last training epoch is zoomed in for easy observation. The lowest value is marked. Due to the high volatility of the training process, a value 5% higher than the minimum value is set as the sign of the training entering the final steady state, represented by the dashed line in the enlarged image. The position is recorded below the dashed line for the first time, indicating that the training process has entered a steady state.

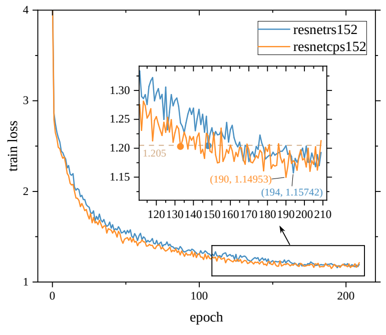
In the experiment, the resnetcps series models (50, 101, 152, 200, 270) are compared with the resnetrs series models. In terms of convergence speed, the resnetcps series is faster. For example, resnetcps50 reaches a minimum value of 1.161 at 203 training epochs, while resnetrs50 reaches a minimum value of 1.162 at 209 training epochs. Resnetcps200 reaches a minimum value of 1.123 at 177 training epochs, while resnetrs200 reaches a minimum value of 1.144 at 198 training epochs. In terms of the number of training epochs required to enter the steady state, the resnetcps series is earlier. For instance, resnetcps50 enters the steady state at 128 training epochs, while resnetrs50 enters the steady state at 138 training epochs. Resnetcps270 enters the steady state at 113 training epochs, while resnetrs270 enters the steady state at 150 training epochs. In terms of the minimum loss value, the difference between the two is not significant, but the resnetcps series is generally slightly smaller. For example, the minimum loss value of resnetcps270 is 1.127, while that of resnetrs270 is 1.138. Because the parameters in the shared part of the model are repeatedly trained, the model, with the addition of the Cross-Layer Parameter Sharing module, has a faster convergence speed and enters the steady-state interval faster. The minimum loss value of the resnetcps with the reserve of the Cross-Layer Parameter Sharing module is smaller than the minimum loss value of the resnetrs model. At the end of the training phase, the loss value of the resnetrs model continues to decline.



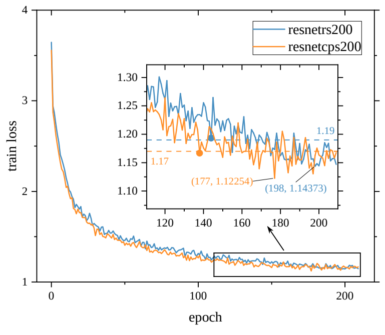
(**a**)



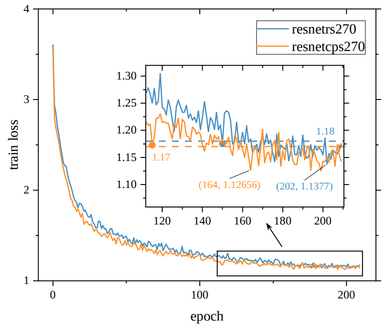
(**b**)



(**c**)



(**d**)



(**e**)

Figure 6. Train loss reduction curve. (**a**) resnetrs50 and resnetcps50. (**b**) resnetrs101 and resnetcps101. (**c**) resnetrs152 and resnetcps152. (**d**) resnetrs200 and resnetcps200. (**e**) resnetrs270 and resnetcps270.

4.4. Comparison Between Models

As shown in Table 1, some important ResNet variant frameworks were tested, including WRN, ResNeXt, ECAResNets, NFNet, Res2Net, ResNetv2, SENet, and RexNet model architectures. The number and accuracy of parameters are used as evaluation indicators. Representative models from various frameworks with the addition of Cross-Layer Parameter Sharing modules are compared. Resnet200 and Resnetrs200, with a sharing ratio of 1/4 and 1/8, were added as representatives for comparison in this article. The selected models are similar to the model in this paper regarding parameter quantity or accuracy. A comparison model with an equal number of parameters to the model in this article will have lower accuracy than the model in this article. The comparison model with similar accuracy to the model in this article has a much larger number of parameters than the model in this article. Therefore, while the model accuracy maintains SOTA, the parameters can remain relatively small.

Table 1. Comparison of parameter quantity and accuracy of ResNet variant models.

Model	Params (M)	Top-1 Acc (%)
wide_resnet50 [26]	68.88	81.46
wide_resnet101 [26]	126.89	78.85
resnetv2_101×1_bitm [46]	44.54	82.33
resnetv2_101×3_bitm [46]	387.93	84.44
nfnet_l0 [28]	35.07	82.75
seresnet152 [31]	66.82	78.65
res2net50 [29]	48.4	78.95
ecaresnet101 [32]	44.57	82.17
ecaresnet269 [32]	102.09	87.97
resnest200 [47]	70.2	83.83
resnest101 [47]	48.28	82.89
resnetcps200 $_{ratio=1/16}$	125.84	88.36
resnetcps350 $_{ratio=1/8}$	153.02	89.33

Table 2 shows some classification models with parameters similar to resnetcps200. resnetcps200 can achieve state-of-the-art performance with similar model sizes.

Table 2. Comparison of parameter quantity and accuracy of each model.

Model	Params (M)	Top-1 Acc (%)
RepVGG-B2 [48]	80.31	81.71
Multiscale DEQ [49]	81	82.25
PVTv2-B4 [50]	82	84.65
RevBiFPN-S5 [51]	82	84.14
Conformer-B [52]	83.3	85.65
XCiT-M24 [53]	84	86.75
DeiT-B [14]	86	84.82
ConViT-B [54]	86	84.66
ViT-B @224 [55]	86.6	84.42
ViT-B @384 [55]	87	85.35
SwinV2-B [56]	88	86.68
Swin-B [56]	88	85.79
Mega [57]	90	83.36
DiNAT-Base [58]	90	85.14
Pyramid ViG-B [59]	92.6	84.84
CoE-Large [60]	95.3	83.47
resnetcps200 $_{ratio=1/4}$	82.21	87.16
resnetcps200 $_{ratio=1/8}$	87.71	87.69

4.5. Ablation Study

A holistic ablation experiment was used to verify the relationship between the model architecture, sharing ratio, parameter amount, memory read and write amount, model size, and accuracy. The number of memory reads and writes, model size, and accuracy are compared with the parameters under different sharing ratios. The model architecture and sharing ratio are analyzed. The experimental data are substation data. Table 3 shows the experimental results of different models of resnetcps architecture. Because this parameter is the denominator to participate in the shared convolution calculation, the more significant the number, the smaller the proportion of parameters participating in the sharing. Params represents the number of parameters. The number of parameters determines the model's size and the training's complexity. The lower the number of parameters, the faster the model training. Memr + w represents the amount of memory read and write. The amount of memory read and write is enough to affect the response speed of the model. The smaller the memory read and write, the faster the model responds. Model size represents the space occupied by the storage model. Top-1 ACC represents the model's classification accuracy at the maximum probability value. The optimal parameters in the table are bolded. Figure 7 is the corresponding diagram of parameters and accuracy of each model of resnetcps architecture. In addition, we also compared the accuracy impact of different channel self-attention modules on resnetcps.

Table 3. Resnetcps model scale with different sharing ratios.

Model	Sharing Ratio	Params	Mem R + W	Model Size	Top-1 Acc
resnetrs50		35.69 M	273.64 M	279 M	85.52
ResNet-cps50	1/16	35.41 M	273.02 M	277 M	85.65
	1/8	35.13 M	271.81 M	275 M	86.39
	1/4	34.58 M	269.39 M	271 M	**86.95**
	1/2	**33.46 M**	**264.56 M**	**262 M**	86.79
resnetrs101		63.62 M	526.85 M	498 M	85.65
ResNet-cps101	1/16	62.16 M	521.01 M	486 M	86.62
	1/8	60.69 M	514.32 M	475 M	87.36
	1/4	57.77 M	500.95 M	452 M	**87.39**
	1/2	**51.92 M**	**474.21 M**	**406 M**	87.26
resnetrs152		86.62 M	1.0 G	678 M	84.75
ResNet-cps152	1/16	84.15 M	1015.25 M	659 M	86.34
	1/8	81.68 M	1001.51 M	639 M	**87.95**
	1/4	76.73 M	974.03 M	601 M	86.32
	1/2	**66.85 M**	**919.06 M**	**523 M**	85.57
resnetrs200		93.21 M	1.31 G	730 M	86.35
ResNet-cps200	1/16	90.46 M	1.29 G	708 M	87.26
	1/8	87.71 M	1.28 G	687 M	**87.69**
	1/4	82.21 M	1.24 G	644 M	87.16
	1/2	**71.21 M**	**1.17 G**	**558 M**	87.08
resnetrs270		129.86 M	1.72 G	1017 M	87.76
ResNet-cps270	1/16	125.84 M	1.7 G	985 M	**88.36**
	1/8	121.82 M	1.68 G	954 M	88.20
	1/4	113.78 M	1.63 G	891 M	88.15
	1/2	**97.69 M**	**1.53 G**	**765 M**	87.64
resnetrs350		163.96 M	2.54 G	1.284 G	87.71
ResNet-cps350	1/16	158.49 M	2.51 G	1.241 G	88.60
	1/8	153.02 M	2.48 G	1.198 G	**89.33**
	1/4	142.09 M	2.4 G	1.113 G	88.39
	1/2	**120.23 M**	**2.26 G**	**942 M**	88.33
resnetrs400		191.89 M	3.53 G	1.502 G	84.76
ResNet-cps400	1/16	185.24 M	3.48 G	1.450 G	85.01
	1/8	178.59 M	3.43 G	1.398 G	85.86
	1/4	165.29 M	3.34 G	1.295 G	**86.32**
	1/2	**138.69 M**	**3.14 G**	**1.087 G**	84.13

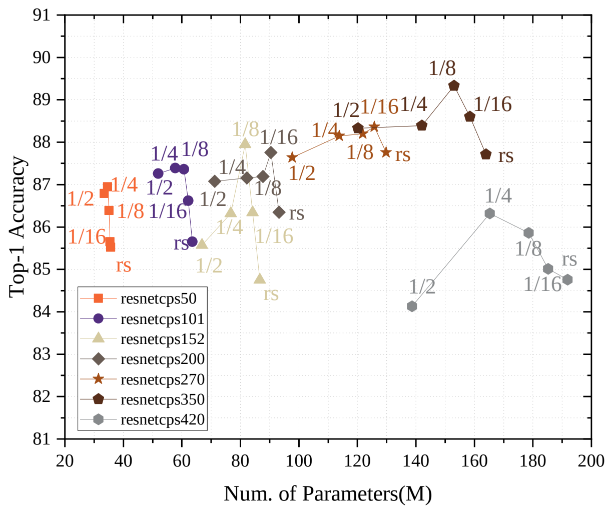


Figure 7. Model parameter quantity and accuracy under different sharing ratios.

4.5.1. Effects of Sharing Ratio on Model Scales

As shown in Table 3, resnetcps models follow the same pattern. As the sharing ratio changes, model parameters, memory reads and writes, and model size gradually decrease. For the largest model, resnetcps420, the reduction in model size is even more significant. At a ratio of 1/16, compared to the resnetcps model, the parameter count decreased by 6.65 M, accounting for 3.47%, the memory read and write count decreased by 43.1 MB, accounting for 1.19%, and the model size decreased by 52 M, accounting for 3.46%. At a ratio of 1/2, compared to the resnetcps model, the parameter count decreased by 53.2 MB, accounting for 27.72%, the memory read and write count decreased by 393.96 MB, accounting for 10.91%, and the model size decreased by 415 M, accounting for 27.63%. It can be seen that the higher the proportion of cross-layer shared parameters, the smaller the model size, which is also in line with the design of Cross-Layer Parameter Sharing. When the proportion of cross-layer shared parameters is half, the model exhibits a significant decrease in scale.

4.5.2. Channel Self-Attention Module

The most outstanding models in resnetcps are selected as the basis. We compare different channel self-attention models. After analyzing the models and comparing their parameters, it was found that the IAA model, which is based on Cross-Layer Parameter Sharing, can better fit the data and achieve higher accuracy. As shown in Table 4, the first column represents not using the channel self-attention model, SE represents the Squeeze and Excitation Module, and ECA represents the Efficient Channel Attention Module.

Table 4. The variation in accuracy with different channel self-attention modules.

Attention / Model		SE	ECA	IAA
resnetcps50 $_{ratio=1/4}$	84.12	85.35	86.12	86.95
resnetcps101 $_{ratio=1/4}$	85.24	86.57	86.58	87.39
resnetcps152 $_{ratio=1/8}$	85.41	87.40	86.66	87.95
resnetcps200 $_{ratio=1/8}$	85.37	86.87	86.81	87.69
resnetcps270 $_{ratio=1/16}$	86.12	88.24	87.78	88.36
resnetcps350 $_{ratio=1/8}$	87.30	88.92	88.50	89.33
resnetcps400 $_{ratio=1/4}$	84.83	85.16	85.67	86.32

4.5.3. Effects of Sharing Ratio on Accuracy

As shown in Figure 7, the Cross-Layer Parameter Sharing module compares the model parameter quantity and accuracy under different sharing ratios. Each line represents a type of model. The horizontal axis represents the number of model parameters. The vertical axis represents the accuracy of the model. The sharing ratio is indicated next to the corresponding points in the graph. When the Cross-Layer Parameter Sharing module is not added, the corresponding point of the model is at the rightmost end of the line, with the most parameters. As the sharing ratio increases, the number of parameters decreases, and the corresponding points move more to the left.

The resnetcps architecture includes seven types of models. As the scale of the resnetcps model increases, the accuracy of the model also increases. The resnetcps420 series has relatively low accuracy. It can be seen that the accuracy fluctuations of different sharing ratios within the same type of model are relatively small. The accuracy of the resnetcps model determines the accuracy range of the model added to the parameter-sharing module. On this basis, the reduction in parameters is significant.

5. Discussion

Considering the starting point of the model design, ResnetCPS emerges as a unified model capable of addressing a class of problems. Looking ahead, future research directions could involve exploring its application in other analogous scenarios, such as anomaly detection in nanotechnology [61,62], thereby potentially expanding its utility and impact in the field.

6. Conclusions

This study proposes a novel approach, namely ResnetCPS, to tackle the problem of scale fluctuations in datasets and ensure the lightweight characteristics of the model. This paper uses images of power equipment as validation, which can be considered for use in other fields with similar features. ResnetCPS mainly includes the following advantages:

Firstly, by leveraging the scale-invariant design of image datasets, ResNetCPS demonstrates enhanced effectiveness in extracting features from the data. This design enables it to better capture the inherent patterns and characteristics within the power equipment images, regardless of variations in scale.

Secondly, the parameter-sharing mechanism of ResNetCPS plays a crucial role. It can significantly reduce both the parameter count and the model size. This reduction not only alleviates the computational burden during training and inference but also holds great promise for practical industrial applications, where resources are often limited and efficiency is of utmost importance.

Thirdly, ResNetCPS empowers the model to achieve faster convergence within a constrained training time frame, simultaneously attaining higher accuracy.

In the experimental phase, when compared with the Resnet variant network, ResnetCPS achieved a remarkable 10–30% reduction in both the number of parameters and the model size. Moreover, when contrasted with networks of similar scale, its accuracy reaches the state-of-the-art level, highlighting its superiority in performance. Codes are available at https://github.com/wdzjs/ResnetCPS/tree/main accessed on 16 November 2024.

Author Contributions: Conceptualization, X.Y. and D.Y.; methodology, X.Y.; software, X.Y. and S.X.; validation, X.L.; formal analysis, W.C.; investigation, D.Y.; resources, L.J.; data curation, H.C.; writing—original draft preparation, X.Y.; writing—review and editing, D.Y. and S.X.; visualization, X.L.; supervision, W.C.; project administration, X.Y.; funding acquisition, D.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (62306232), the Natural Science Basic Research Program of Shaanxi Province (2023-JC-QN-0662), the State Key Laboratory of Electrical Insulation and Power Equipment (EIPE23416), and Fundamental Research Funds for the Central University and HPC Platform, Xi'an Jiaotong University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: 3rd Party Data Restrictions apply to the availability of these data. Data were obtained from State Grid and are available from the authors with the permission of State Grid.

Conflicts of Interest: Authors Xiao Liao and Wei Cui were employed by the company State Grid Information & Telecommunication Group Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Guan, X.; Gao, W.; Peng, H.; Shu, N.; Gao, D.W. Image-based incipient fault classification of electrical substation equipment by transfer learning of deep convolutional neural network. *IEEE Can. J. Electr. Comput. Eng.* **2021**, *45*, 1–8. [CrossRef]
2. Gao, F.; Li, Q.; Ji, Y.; Ji, S.; Guo, J.; Sun, H.; Liu, Y.; Feng, S.; Wei, H.; Wang, N.; et al. EWNet: An early warning classification framework for smart grid based on local-to-global perception. *Neurocomputing* **2021**, *443*, 199–212. [CrossRef]
3. Zhao, Z.; Feng, S.; Zhai, Y.; Zhao, W.; Li, G. Infrared Thermal Image Instance Segmentation Method for Power Substation Equipment Based on Visual Feature Reasoning. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–13. [CrossRef]
4. Nassu, B.T.; Marchesi, B.; Wagner, R.; Gomes, V.B.; Zarnicinski, V.; Lippmann, L. A computer vision system for monitoring disconnect switches in distribution substations. *IEEE Trans. Power Deliv.* **2021**, *37*, 833–841. [CrossRef]
5. Zhang, F.; Bales, C.; Fleyeh, H. From time series to image analysis: A transfer learning approach for night setback identification of district heating substations. *J. Build. Eng.* **2021**, *43*, 102537. [CrossRef]
6. Shuang, F.; Wei, S.; Li, Y.; Gu, X.; Lu, Z. Detail R-CNN: Insulator Detection based on Detail Feature Enhancement and Metric Learning. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–14. [CrossRef]
7. Shiozaki, K. The classification of surface states of topological insulators and superconductors with magnetic point group symmetry. *Prog. Theor. Exp. Phys.* **2022**, *2022*, 04A104. [CrossRef]
8. Li, W.; Wang, S.; Ullah, I.; Zhang, X.; Duan, J. Multiple attention-based encoder–decoder networks for gas meter character recognition. *Sci. Rep.* **2022**, *12*, 10371. [CrossRef]
9. Zhang, Z.; He, G. Recognition of bird nests on power transmission lines in aerial images based on improved YOLOv4. *Front. Energy Res.* **2022**, *10*, 870253. [CrossRef]
10. Liao, J.; Xu, H.; Fang, X.; Miao, Q.; Zhu, G. Quantitative Assessment Framework for Non-Structural Bird's Nest Risk Information of Transmission Tower in High-Resolution UAV Images. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–12. [CrossRef]
11. Dong, X.; Taylor, C.J.; Cootes, T.F. Defect classification and detection using a multitask deep one-class CNN. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 1719–1730. [CrossRef]
12. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
13. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 10012–10022.
14. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning, PMLR, Online, 18–24 July 2021; pp. 10347–10357.
15. Chen, C.F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 357–366.
16. Ding, M.; Xiao, B.; Codella, N.; Luo, P.; Wang, J.; Yuan, L. Davit: Dual attention vision transformers. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 74–92.
17. Wang, W.; Xie, E.; Li, X.; Fan, D.P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 568–578.
18. Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; Jégou, H. Going deeper with image transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 32–42.
19. Yang, J.; Li, C.; Zhang, P.; Dai, X.; Xiao, B.; Yuan, L.; Gao, J. Focal self-attention for local-global interactions in vision transformers. *arXiv* **2021**, arXiv:2107.00641.
20. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
21. Liu, H.; Dai, Z.; So, D.; Le, Q.V. Pay attention to mlps. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 9204–9215.
22. Touvron, H.; Bojanowski, P.; Caron, M.; Cord, M.; El-Nouby, A.; Grave, E.; Izacard, G.; Joulin, A.; Synnaeve, G.; Verbeek, J.; et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 5314–5321. [CrossRef]
23. Chen, S.; Xie, E.; Ge, C.; Chen, R.; Liang, D.; Luo, P. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv* **2021**, arXiv:2107.10224. [CrossRef]

24. 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.
25. He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; Li, M. Bag of tricks for image classification with convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 558–567.
26. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146.
27. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
28. Brock, A.; De, S.; Smith, S.L.; Simonyan, K. High-performance large-scale image recognition without normalization. In Proceedings of the International Conference on Machine Learning, PMLR, Online, 18–24 July 2021; pp. 1059–1071.
29. Gao, S.H.; Cheng, M.M.; Zhao, K.; Zhang, X.Y.; Yang, M.H.; Torr, P. Res2net: A new multi-scale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 652–662. [CrossRef]
30. Bello, I.; Fedus, W.; Du, X.; Cubuk, E.D.; Srinivas, A.; Lin, T.Y.; Shlens, J.; Zoph, B. Revisiting resnets: Improved training and scaling strategies. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22614–22627.
31. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
32. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient channel attention for deep convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 11534–11542.
33. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
34. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
35. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
36. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
37. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
38. Gao, H.; Wang, Z.; Ji, S. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 2570–2581. [CrossRef]
39. Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 10734–10742.
40. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 1580–1589.
41. Li, Y.; Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Yuan, L.; Liu, Z.; Zhang, L.; Vasconcelos, N. Micronet: Improving image recognition with extremely low flops. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 468–477.
42. Dong, W.; Lv, J.; Chen, G.; Wang, Y.; Li, H.; Gao, Y.; Bharadia, D. TinyNet: A lightweight, modular, and unified network architecture for the internet of things. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services, Portland, OR, USA, 27 June–1 July 2022; pp. 248–260.
43. Chen, J.; He, T.; Zhuo, W.; Ma, L.; Ha, S.; Chan, S.H.G. Tvconv: Efficient translation variant convolution for layout-aware visual processing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12548–12558.
44. Chen, J.; Kao, S.h.; He, H.; Zhuo, W.; Wen, S.; Lee, C.H.; Chan, S.H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 12021–12031.
45. Deo, A.J.; Behera, S.K.; Das, D.P. Online Monitoring of Iron Ore Pellet Size Distribution Using Lightweight Convolutional Neural Network. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 1974–1985. [CrossRef]
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part IV 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 630–645.
47. Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Lin, H.; Zhang, Z.; Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; et al. Resnest: Split-attention networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 2736–2746.

48. Ding, X.; Zhang, X.; Ma, N.; Han, J.; Ding, G.; Sun, J. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 13733–13742.
49. Bai, S.; Koltun, V.; Kolter, J.Z. Multiscale deep equilibrium models. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 5238–5250.
50. Wang, W.; Xie, E.; Li, X.; Fan, D.P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pvt v2: Improved baselines with pyramid vision transformer. *Comput. Vis. Media* **2022**, *8*, 415–424. [CrossRef]
51. Chiley, V.; Thangarasa, V.; Gupta, A.; Samar, A.; Hestness, J.; DeCoste, D. RevBiFPN: The fully reversible bidirectional feature pyramid network. *Proc. Mach. Learn. Syst.* **2023**, *5*, 625–645.
52. Peng, Z.; Huang, W.; Gu, S.; Xie, L.; Wang, Y.; Jiao, J.; Ye, Q. Conformer: Local features coupling global representations for visual recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 367–376.
53. Ali, A.; Touvron, H.; Caron, M.; Bojanowski, P.; Douze, M.; Joulin, A.; Laptev, I.; Neverova, N.; Synnaeve, G.; Verbeek, J.; et al. Xcit: Cross-covariance image transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 20014–20027.
54. d'Ascoli, S.; Touvron, H.; Leavitt, M.L.; Morcos, A.S.; Biroli, G.; Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 2286–2296.
55. Touvron, H.; Cord, M.; Jégou, H. Deit iii: Revenge of the vit. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 516–533.
56. Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y.; Ning, J.; Cao, Y.; Zhang, Z.; Dong, L.; et al. Swin transformer v2: Scaling up capacity and resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12009–12019.
57. Ma, X.; Zhou, C.; Kong, X.; He, J.; Gui, L.; Neubig, G.; May, J.; Zettlemoyer, L. Mega: Moving average equipped gated attention. *arXiv* **2022**, arXiv:2209.10655.
58. Hassani, A.; Shi, H. Dilated neighborhood attention transformer. *arXiv* **2022**, arXiv:2209.15001.
59. Han, K.; Wang, Y.; Guo, J.; Tang, Y.; Wu, E. Vision gnn: An image is worth graph of nodes. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 8291–8303.
60. Zhang, Y.; Chen, Z.; Zhong, Z. Collaboration of experts: Achieving 80% top-1 accuracy on imagenet with 100m flops. *arXiv* **2021**, arXiv:2107.03815.
61. Ieracitano, C.; Mammone, N.; Paviglianiti, A.; Morabito, F.C. A conditional generative adversarial network and transfer learning-oriented anomaly classification system for electrospun nanofibers. *Int. J. Neural Syst.* **2022**, *32*, 2250054. [CrossRef] [PubMed]
62. Napoletano, P.; Piccoli, F.; Schettini, R. Anomaly detection in nanofibrous materials by CNN-based self-similarity. *Sensors* **2018**, *18*, 209. [CrossRef] [PubMed]