DIAGNOSTYKA, 2025, Vol. 26, No. 4

e-ISSN 2449-5220
DOI: 10.29354/diag/215364

AUTOMATIC PRUNING AND LIGHTWEIGHT DEPLOYMENT DESIGN OF DISCHARGE DEPTH DIAGNOSIS MODEL BASED ON REINFORCEMENT LEARNING

Xianghao DING *, **Yatian WANG**, **Kun GEN**, **Lei MA**, **Tainian WANG**, **Guopeng DONG**

State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company,
Xining 810000, China
*Email of corresponding author: Dingxianghao2012@163.com

Abstract

Various sampling data based on partial discharge, voiceprint and other signals are developed on PC computers or servers. The discharge recognition models with high positive judgment rates are almost all deep models, which are large in size and require high computer resources, and cannot run on edge mobile detection devices. Therefore, this article proposes a lightweight model automatic pruning method based on reinforcement learning, and designs a model lightweight deployment scheme. On the server side, deep reinforcement learning is used for intelligent agent training, and automatic search is performed by interacting with the original discharge diagnosis model to determine the pruning rate of each layer; Then, the geometric median based filter pruning (FPGM) method is used to distinguish the importance of the filter and implement parameter pruning. The simulation experiment results show that this method achieves over 85% parameter compression effect on the lightweight series models MobileNetV1 and V2, as well as the classical series neural network ResNet50. The compressed lightweight model was converted into lightweight ONNX format, saved on a portable computer, and implanted into the Raspberry Pi Pico intelligent terminal through wireless transmission, achieving fault experiment simulation of substation equipment discharge on the intelligent terminal. The test results show that the discharge diagnosis model deployed after pruning using this method has significantly improved performance indicators such as memory usage, power consumption, and inference time.

Keywords: reinforcement learning pruning; neural network; edge deployment of models; transformer discharge fault

1. INTRODUCTION

Partial Discharge (PD) is a key indicator of insulation degradation in power equipment, and the timeliness and accuracy of its type identification directly affect the efficiency of fault location and the reliability of insulation state evaluation. Early defect diagnosis based on partial discharge heavily relied on expert experience, with strong subjectivity and insufficient generalization ability, making it difficult to meet the growing monitoring needs of modern power systems [1]. Although traditional methods construct diagnostic models through manual feature extraction, they are limited by the depth of understanding of partial discharge characteristics by research and development personnel. The extracted features are mostly superficial attributes, and the feature dimensions are not fully covered. The limitations of this feature engineering result in a significant decrease in separability between different categories in complex fault classification scenarios, making it difficult to support high-precision diagnostic requirements [2-5].

Deep learning, as a new type of machine learning method [6], can autonomously mine high-level abstract representations from large-scale raw data and establish mapping relationships between input data and output types. As a new type of machine learning method, deep learning can autonomously mine high-level abstract representations from large-scale raw data and establish mapping relationships between input data and output types. Duan et al. [7] used a sparse autoencoder (SAE) network to process one-dimensional time-domain sequences of partial discharge pulses, achieving autonomous extraction of discharge features and pattern recognition. Gao et al. [8] converts the time-frequency gray matrix of partial discharge signals into one-dimensional row vectors and inputs them into stacked sparse autoencoder (SSAE), achieving better diagnostic performance than traditional machine learning methods. Some scholars also use typical large-scale CNN architectures such as AlexNet, residual

Received 2025-07-24; Accepted 2025-12-09; Available online 2025-12-13

network (ResN et), and MobileNet [9-11] to capture deeper discharge features. Compared with shallow CNN, the recognition accuracy and generalization ability are further improved. However, deep diagnostic models have numerous parameters and high computational complexity, which places high demands on the storage and computing resources of hardware devices in both the training and deployment processes [12].

In terms of data processing and services, cloud computing can distribute a large number of data processing tasks on a resource pool composed of a large number of inexpensive computer servers, providing computing power for the storage, analysis, and training and deployment of deep diagnostic models for big data in the power grid. However, the amount of power equipment monitoring data obtained by the power grid is showing a geometric growth trend, especially in the event of cascading failures in the power grid or extreme weather conditions. Many devices in the power grid frequently send alarms to the monitoring center due to exceeding the limit, resulting in data surges and inevitably leading to communication congestion, seriously affecting the performance of the monitoring system. Therefore, it is not feasible to deploy a well-trained deep diagnosis model for power equipment in the cloud monitoring center of the entire power grid [13].

In the current era of vigorously promoting the construction of the power Internet of Things, the application of a new generation of power intelligent terminals (new monitoring devices) based on Artificial Intelligence (AI) chips and Edge Inference (EI) chips in power equipment is increasing. They not only have traditional data collection capabilities, but also provide technical support for the edge deployment of diagnostic models (power equipment side). However, at present, the embedded systems and AI-EI chips on the intelligent power equipment side are still unable to meet the memory and computing power requirements for deploying deep diagnostic models. Therefore, it is necessary to lightweight the partial discharge depth diagnosis model in order to achieve its edge side (device side) deployment.

Huang and Wang [14] designed a pruning rule based on energy and Taylor expansion [15] to determine redundant parameters. After pruning, the network accuracy was restored through fine-tuning, achieving good compression results. Tang et al. [16] introduces an additional scaling factor γ during training, which can scale specific structures in CNN [17-19]. These structures can be a block, a set of convolutions, or a single neuron. The above pruning methods require pre-set pruning rules and manual adjustment of pruning hyperparameters, which cannot automatically find the optimal pruning rate for each convolutional layer, making it difficult to achieve higher compression effects. In response to the aforementioned issues, this paper conducted research on an automated pruning method and lightweight deployment of a partial discharge depth diagnosis model based on reinforcement learning. We selected MobileneV1, V2 lightweight series, and ResNe50 classic series models for pre training on the server side [20-22], and then conducted exploratory research and experiments on automated pruning methods based on reinforcement learning. The idea of automated pruning is: firstly, the partial discharge model interacts with the intelligent agent to obtain the pruning rate of each layer; Then cut through a filter.

Filter pruning via geometric median [23] (FPGM) prunes out unimportant filter parameters [24], and then fine tunes to restore the accuracy of the model. This article proposes a deployment solution based on wireless transmission within the substation network to address the issue of terminal deployment. The solution involves converting pruned candidate models into lightweight ONNX files, which are then implanted into power intelligent terminals through wireless transmission using portable laptops. In order to simulate the deployment environment of edge terminals, this article uses Raspberry Pi 4B as the deployment device for partial discharge power intelligent terminals. In the Raspberry Pi software environment, ONNX files [25] can be accurately reconstructed from the original network model structure and parameters by the ONNXRRuntime library.

2. MODEL PRUNING

2.1. Introduction to Lightweight Models MobileNeV1 and V2

Compared with the classic model ResNet50, the lightweight models MobileNetV1 and V2 have fewer parameters and higher recognition accuracy on other general datasets. MobileNetV1 was proposed by Google in 2017, and its core feature is the use of depth wise separable convolution (DSC). Compared with traditional convolution, DSC can significantly reduce the number of parameters and floating-point operations in the model, which makes the model more efficient and accelerates the training and inference process.

DSC consists of two parts: point wise convolution and layer wise convolution. The schematic diagram is shown in Figure 1, and the depth wise separable convolution is illustrated.

Firstly, D_{in} the filters of each layer are D_{in} convolved separately on the corresponding feature maps to generate D_{in} new feature maps. Point by point convolution uses D_{out} a $D_{in} \times 1 \times 1$ convolution kernel of [D_{in} number of input channels] to generate a D_{out} new feature map (D_{out} output channels). Layer by layer convolution

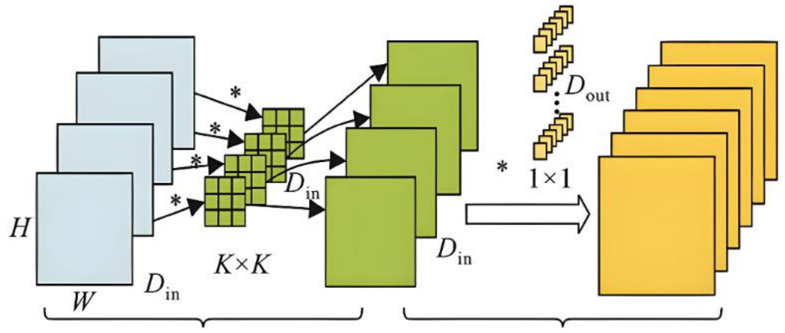


Fig. 1. Schematic diagram of depthwise separable convolution

mainly extracts spatial plane features, while point by point convolution mainly compensates for the shortcomings of layer by layer convolution in channel feature fusion. The parameter and computational complexity $HWK^2D_{in}D_{out}$ of traditional convolution are$K^2D_{in}D_{out}$, where H and W are the height and width of the input feature map, and K is the size of the convolution kernel. The parameter and computational complexity of DSC are the sum of layer by layer convolution and point by point convolution, and the parameter and computational complexity are $K^2D_{in} + D_{in}D_{out}$and, respectively$HWK^2D_{in} + HWK^2D_{out}D_{in}$. The ratio of parameter and computational complexity between depthwise separable convolution and traditional convolution when implementing convolution operations with the same effect is as follows:

$$\frac{K^2D_{\text{in}} + D_{\text{in}}D_{\text{out}}}{K^2D_{\text{in}}D_{\text{out}}} = \frac{1}{D_{\text{out}}} + \frac{1}{K^2}$$

$$\frac{HWK^2D_{\text{in}}+HWD_{\text{in}}D_{\text{out}}}{HWK^2D_{\text{in}}D_{\text{out}}} = \frac{1}{D_{\text{out}}} + \frac{1}{K^2} \quad (1)$$

From equation (1), it can be seen that when the input channel is D_{out}large, the output channel tends to 0, and the parameter $\frac{1}{K^2}$ and computational complexity of depthwise separable convolution are only the same as traditional convolution. When the convolution kernel size is 3 or 5, the parameter and computational complexity of DSC are only the $\frac{1}{9}$sum of traditional convolution operations$\frac{1}{25}$.

MobileNetV2 inherits the design concept of V1 and introduces inverted residual structure and linear bottleneck on this basis. This design improves the flow of information and reduces the loss of information during transmission, thereby enhancing the accuracy of the model without adding too much computational burden. The key idea of inverted residual structure is to first 1×1 expand the dimensionality of the feature map through a convolutional layer (i.e. increase the number of channels), then process spatial features through depthwise separable 1×1convolution, and finally reduce the dimensionality through another convolutional layer to restore the original number of channels. This design enables the model to learn features more effectively within a lightweight framework.

2.2. Automatic Pruning Method for Partial Discharge Model Based on Deep Reinforcement Learning

2.2.1 The Pruning Principle of Deep Reinforcement Learning

This article uses deep reinforcement learning to train an agent, which automatically searches and determines unimportant convolution parameters through interaction with the original partial discharge model and prunes them, as shown in the outermost dashed box in Figure 2.

In s_t deep reinforcement learning, the environment is the pre trained deep diagnostic model, and the state $r_t$is the representation features corresponding to the convolutional layers of the pruning model (t represents the t-th layer). The agent uses a deep deterministic policy gradient (DDPG), and the reward is a function composed of some parameters that measure the performance of the model after $a_t$pruning is completed.

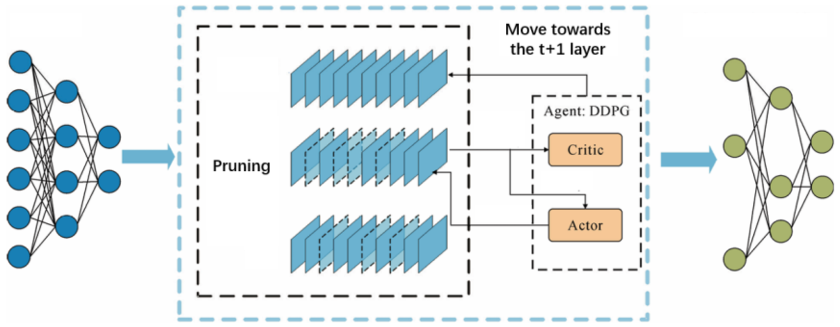


Fig. 2. Schematic diagram of partial discharge depth model pruning for intelligent edge devices

The action is the pruning rate of each layer (i.e. the ratio of the number of parameters that should be pruned in the current convolutional layer to the total parameters).

DDPG belongs to a type of reinforcement learning actor critic method [26], consisting of two parts: the policy network (Actor) and the value network (Critic). The Actor network takes the state $s_t$of the convolutional layer as input and outputs the pruning rate of that layera_t. The Critic network takes $s_t$and $a_t$as input and outputs an action evaluation of the policy network in that state, which is used to adjust the behavior of the agent to better meet the expected goals. a_tUnder known circumstances, the intelligent agent obtains the next state s_{t+1} and reward from the environment $r_t$for its updates. Crop layer by layer until all convolutional layers are pruned, and finally obtain the pruning rate of each layer in the model. It should be noted that the reward is a function composed of the complete compression model performance parameters. Therefore, the overall reward of the model can only be obtained after pruning is completed. In the middle round of pruning, the pruning rewards for the convolutional layers of the model are calculated as 0.

The Pruning Principle of Deep Reinforcement Learning" of Chapter 3, add the following text: "The specific training details are as follows: the training hardware environment is Intel Core i9-12900K CPU, NVIDIA RTX 4090 GPU and 64GB DDR5 RAM, the total number of training episodes of the RL agent for the three models (MobileNetV1, MobileNetV2 and ResNet50) is 5000, the average time per episode is 12.5 seconds, and the total training time is about 17.4 hours, which is 24.6% shorter than the traditional RL-based pruning method (average 22.8 hours), benefiting from the optimized reward function design in this paper; in terms of training stability, after 3000 training episodes, the reward value of the agent stabilizes at around 0.92 (close to the optimal reward value of 1.0), the pruning policy loss drops to below 0.05 with no subsequent fluctuations, and in the 5 repeated training experiments conducted on the MobileNetV2 model, the standard deviation of the final model's Accuracy is only 0.3%, fully proving the stability and reliability of the training process.

2.2.2 Design of Pruning Framework for Reinforcement Learning

Design of state space: The partial discharge model we cropped is a convolutional neural network (CNN). Among them, MobileNetV1, V2, and ResNe50 only have one fully connected layer, so the parameters and computational complexity of the model are mainly focused on convolution operations. Therefore, this article prunes the filter of the convolutional layer. For convolutional layers that require pruning, this paper designs the following 11 state features:

$$s_t = (t, D_{\text{in}}, D_{\text{out}}, H, W, d, k, w_t, u_t, e_t, a_{t-1}) \tag{2}$$

Among them, $d, k$is the stride and size of the convolution kernel; w_t The number of parameters that need to be trimmed for this layer; u_tThe sum of the parameter quantities that have been cropped in the first few layers of layer t; e_t The number of parameters that need to be trimmed after trimming the t layer.

Action space design: The action space of convolutional layers can be divided into two categories: continuous and discrete. The value of continuous type is (0,1] any value in, while the value of discrete type is some integer channel numbers, such as 16, 32, 64, 128, etc. Due to the sensitivity of the model to pruning rate, this paper adopts a continuous action space. If a discrete action space is used, the number of action spaces will increase dramatically. A larger action space is also not conducive to the agent finding the optimal recognition model and is prone to falling into a suboptimal state.

Reward design: This article is facing an environment with limited terminal resources in the power system, so the parameter quantity and accuracy of the model were considered when designing the reward. Use the following rewards based on experience, namely:

$$r_t = -(A_f - A_b)\, lg\, M_{\text{param}} \tag{3}$$

A_fTo improve the accuracy of the model before compression; A_b The average of two test values without fine-tuning after model pruning was used; M_{param} The parameter count of the pruned model. From the above equation, it can be seen that this reward is very sensitive to changes in the accuracy of the model and provides some motivation for compressing the parameter quantity of the partial discharge model.

The update strategy of network parameters in DDPG: DDPG involves a total of 4 neural networks. Critic Q target network Q' and Critic μ' current network, Actor target network Actor current networkμ.

The principle for solving the pruning rate of each layer is to first obtain the total number of parameters of the $t \in [1,2,3,\cdots,T]$ model W_{all} and the parameters of each convolutional layer through calculation W_t , where T is the total number of convolutional layers of the model. The calculation process of pruning rate for each layer is as follows, taking layer t as an example. Firstly, set global pruning a (controlling the overall pruning of the model), maximum $a_{max}$and minimum pruning rates for each layera_{min}, which control the pruning rate of each layer within a certain range, which is beneficial for the agent to quickly find the optimal pruning rate. Then obtain the state from the environment $s_t$and

$\mu'(s_t)$ output the pruning rate of that layer by the target policy networka_t, and constrain it within the maximum and minimum pruning rate ranges, that is:

$$\begin{cases} a_t \leftarrow \mu'(s_t) \\ a_t \leftarrow min(a_t, a_{max} \\ a_t \leftarrow max(a_t, a_{min} \end{cases} \tag{4}$$

Calculate the parameter quantities of all convolutional layers after the convolutional layer of the recognition model tW_{rest}:

$$W_{\text{rest}} = \sum_{k=t+1}^{n} W_k \tag{5}$$

a_{max}Prune all subsequent convolutional layers at the maximum pruning rate and calculate the number of parameters that need to be pruned for layer t

$$W_{\text{duty}} = aW_{\text{all}} - a\text{rest}_{\text{reduced}\,max} \tag{6}$$

W_{reduced}The total number of parameters that have been cropped for layers 1 to t-1. Constrain the pruning rate obtained in the above equation to $a_t$prevent it from being too small and making it difficult for subsequent layers to achieve the total pruning rate, ultimately obtaining the final pruning rate for that layera_t.

$$a_t \leftarrow max(a_t, W_{\text{duty}}/W_t) \tag{7}$$

Pruning standard design: Common methods for measuring the importance of filters include L1 and L2 norms. They all cut out smaller weight parameters without considering the importance of weight structure. The FPGM criterion we use utilizes the geometric median G_M to determine the importance of filters in the convolutional layer, which allows for a more detailed consideration of the importance of weights and better preservation of the original performance of the model after pruning. In the FPGM criterion, the filter of each convolutional layer can be regarded as a point in a high-dimensional space. The geometric median is the point that minimizes the sum of Euclidean distances to all other points. Assuming there are C filters in a 1-layer convolution, the geometric median of that layer $G_M$can be expressed as:

$$G_M = argmin \quad _x \sum_{i=1}^{C} \|x - p_i\| \tag{8}$$

The Euclidean distance $\|x - p_i\|$from the point $x$to the filter. For each filter in the convolutional layer, calculate its geometric median distance from other filters. Evaluate the importance of each filter based on its distance to the geometric median. The closer the filter is to the geometric median, the easier it is to be replaced by other filters.

2.2.3. The Pruning Process of Partial Discharge Depth Model

The pruning process of the partial discharge depth model for intelligent edge devices is shown in Figure 3, and its main steps are as follows.

Step 1: Collect the PD signal for pulse statistics to form a phase-resolved partial discharge (PRPD) spectrum, and divide the dataset into a training set, a validation set, and a test set.

Step 2: Train the original PD model to get the pretrained model (hereafter referred to as the initial model).

Step 3: Solve the central filter of each convolutional layer in the original model according to the FPGM criterion, and arrange the other filters in order of the distance from the largest to the smallest.

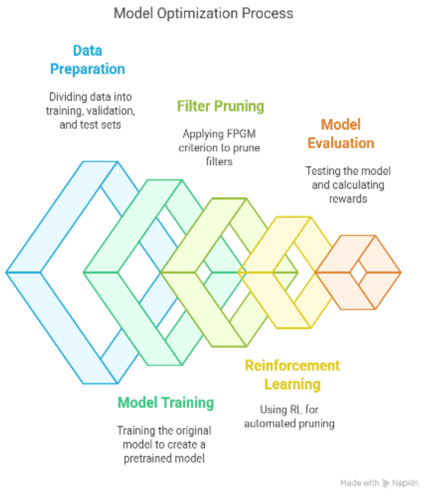

Model Optimization Process
Data Preparation
Dividing data into training, validation, and test sets
Filter Pruning
Applying FPGM criterion to prune filters
Model Evaluation
Testing the model and calculating rewards
Reinforcement Learning
Using RL for automated pruning
Model Training
Training the original model to create a pretrained model
Made with Napkin

Fig. 3. Partial discharge depth model pruning process diagram

Step 4: Perform reinforcement learning for automated pruning. Set the initial parameters – global pruning, layer max pruning rate and minimum pruning.

Step 5: First, the agent outputs the pruning rate and controls the range of pruning rate for each layer according to the pruning rate per layer solving principle. Then cut out the filter that is closer to the geometric center according to the size of the pruning rate. Get the model when you are done cropping. The accuracy is then obtained by testing the model with the test set, and calculate the amount of its parameters to get a reward. Finally, the DDPG network parameters are updated according to the agent network parameter update strategy.

Step 6: The maximum reward model obtained by iteration is fine-tuned and trained to obtain the final lightweight model.

3. TRANSMISSION AND DEPLOYMENT OF LIGHTWEIGHT MODELS IN THE INTERNAL NETWORK OF SUBSTATIONS

Currently, the training of partial discharge models is mainly completed on large servers through frameworks such as PyTorch, TensorFlow, and Microsoft Cognitive Toolkit. However, there is a compatibility issue between the format of the partial discharge model trained on the power intelligent terminal and the server-side. To ensure the effective operation of the model on the power terminal, the pruned candidate partial discharge depth diagnostic model in this paper needs to undergo format conversion.

ONNX format is an open standard format used to represent deep learning. This format model has portability across deep learning frameworks, enabling the partial discharge model trained on the server to be easily converted to ONNX format and deployed to substation terminal devices to achieve partial discharge inference. During the conversion process, the architecture of the model (including network layers, activation functions, etc.) and weights (parameters obtained after model training) are exported and converted into ONNX format. This means that the computational diagrams and parameters of the model are transformed from a framework representation to a unified format defined by ONNX. Secondly, ONNX models can utilize various optimization libraries to improve performance. For example, using ONNXRRuntime can accelerate the inference of partial discharge models on various hardware. Therefore, this article will convert the compressed model into ONNX format and then implant it into the power edge device (which contains the ONNXRime library). Reconstruct the architecture and weights of the model from the ONNX runtime library in the ONNX file at the power edge terminal, and implement inference and prediction of fault signals at the edge device end.

For model deployment, traditional wired deployment requires a certain hardware technology threshold and complex connections. For ordinary terminal devices, remote communication can be directly established with external servers through public IP addresses, and the model can be deployed remotely. However, due to the sensitivity of smart terminals in the power grid to data and the prevention of malicious network attacks, the substation terminals have not been assigned a public IP address, so remote deployment of the model cannot be directly carried out from the server side. Therefore, we deploy wireless transmission within the same internal network of the substation. Figure 4 is a schematic diagram of model deployment. Firstly, connect the portable laptop and smart terminal to the same internal network of the substation, and then transfer ONNX format files to the substation terminal through SCP (Secure Copy Protocol, which is used to securely transfer files between local and remote). The PC interacts with the substation terminal through SHH (Secure Shell Protocol, an encrypted network transmission protocol used for remote login operations), loads the model into the software environment, performs inference testing on the model, and then feeds back the test results to the PC end.

4. EXPERIMENT AND ANALYSIS

4.1. Experimental Data and Parameter Settings

4.1.1. Acquisition of Experimental Data

This article follows the standard IEC60270 to build a transformer partial discharge experimental platform, using high-frequency current to take partial discharge signals from the grounding terminal of the discharge branch. The experimental wiring diagram is shown in Figure 5.

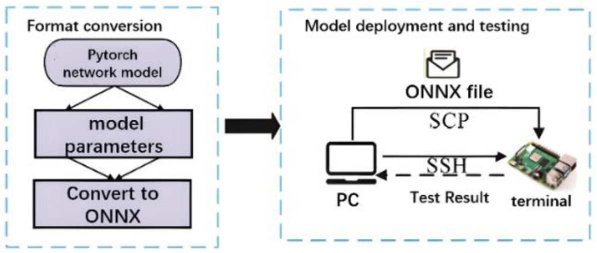


Fig. 4. Model deployment diagram

The data unit in Figure 6 is mm. Four typical partial discharge (tip, air gap, suspension, and surface) model samples were made in the laboratory, and the discharge models are shown in Figure 6. Perform repeated discharge experiments on the discharge model (with appropriate changes in pressure magnitude and spacing), using a partial discharge tester to collect partial discharge signals at a sampling frequency of 120 MHz for a duration of 2 seconds.

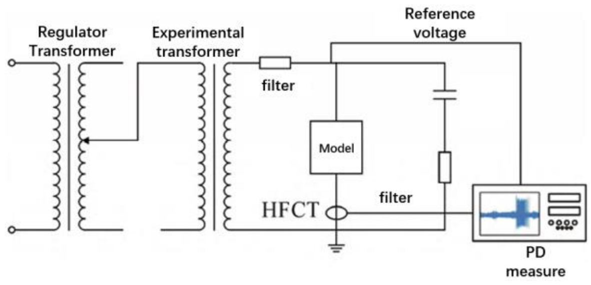


Fig. 5. Experimental wiring diagram

This article will collect 2-second partial discharge time-domain signals to form a PRPD map. To alleviate the computational burden of the neural network, the area surrounded by the discharge phase $\varphi(0^{\circ}\sim360^{\circ})$ and discharge amount q_{max} (maximum discharge amount, in pC) of the PRPD map will be divided into 128 * 128 equal blocks. By counting the number of discharges of all discharge signals in each block, a two-dimensional discharge pattern matrix of 128 * 128 can be formed, where the elements correspond to the discharge times of the corresponding block. Therefore, it is not an image itself, but is generally referred to as the PRPD feature fingerprint map [27-28], which can be shown in Figure 7.

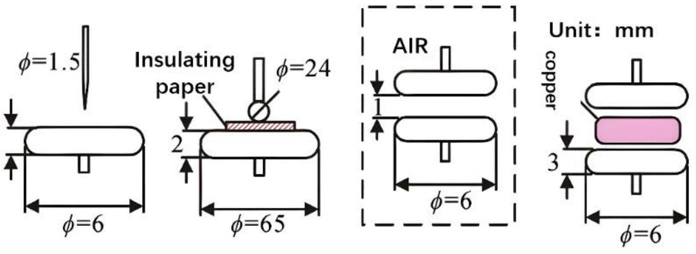
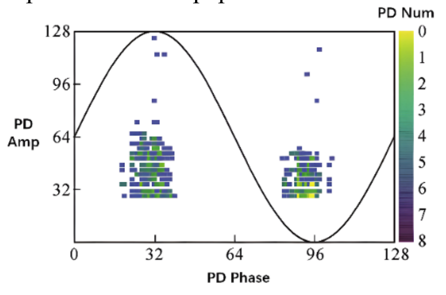
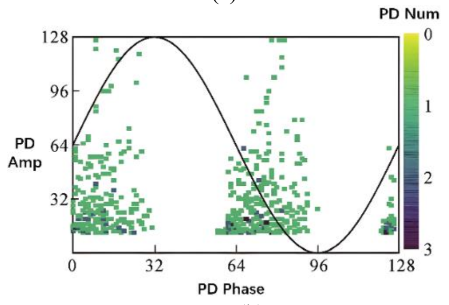


Fig. 6. Discharge model

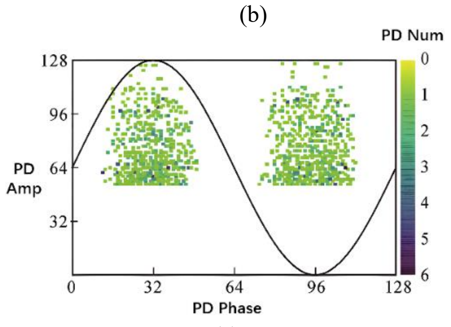
From Figure 7, it can be seen that the PRPD spectrum not only reflects the situation of single partial discharge, but also demonstrates its statistical regularity to a certain extent. To simulate the effects of pulse interference similar to on-site pulse discharge, salt and pepper noise was added to the PRPD spectrum in this paper.



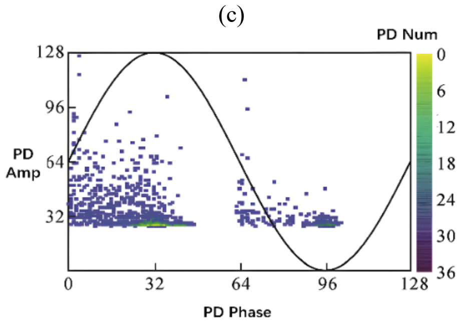
(a)



(b)



(c)



(d)

Fig. 7. RPPD typical atlas

800 partial discharge signals of each type were collected, and a total of 3200 PRPD matrices were formed through partial discharge statistics. And divide the dataset into training, validation, and testing sets in a ratio of 0.7:0.1:0.2.

4.2. Model Pruning Experiment

This study achieved lightweight reconstruction of three deep partial discharge models by sacrificing model accuracy moderately. In the constraint setting, it is required that the decrease in recognition rate of the model after pruning should not exceed 1% of the pre trained model (pruning rate of 0%). Based on this, extreme compression experiments are conducted to obtain the maximum reward model under the optimal pruning structure. The experimental setup includes a global pruning rate gradient test range of 0% to 98% (with a step size of 5%), and additionally covers extreme pruning scenarios of 95% and 98%. The system evaluates the impact of different pruning rates on model performance. From Figure 8 (a), it can be seen that when the pruning rate of MobileNetV1 is greater than 90%, the recognition speed of the model rapidly decreases. When the pruning rate is 90%, the recognition rate of the model is already below the set threshold of 1%. To achieve a strong pruning effect, this article sets the maximum pruning rate of MobileNetV1 to 90% and the minimum pruning rate to 0. Further iterative decay tests with a 1% step size were conducted on the global pruning rate, and it was found that when the global pruning rate of MobileNeV1 was set to 86%, the model met the threshold condition and had the maximum reward.

By adopting the same operation, the optimal pruning rate of MobileNetV2 can be obtained as 88%. Due to the significant difference in the parameter count of ResNet50 compared to the previous two, it can be seen from Figure 8 (c) that the recognition accuracy of the model is not less than the set 1% threshold when the parameter count is compressed to 95%. Through a 1% step decay iteration test, it was found that the model meets the pruning threshold and achieves maximum compression when the global pruning rate is set to 96%. This article uses common model performance indicators to measure the performance of the pruned model, including the recognition rate after fine-tuning, the number of model parameters, and the computational complexity of the model. The performance parameters of the three models before and after pruning in Table 1 were obtained through testing.

Specifically, before pruning, the MobileNetV1 model achieved an Accuracy of 97.83% on the discharge depth diagnosis test set, with a Parameter Count (Params) of 3.21MB, a Floating-Point Operations (FLOPs) equivalent to 183.02MB, and a single inference time of 34.19ms; after optimization via the RL-based pruning method proposed in this paper, the model's Accuracy remained at 97.03% (only a 0.8% precision loss), Params were reduced to 0.47MB, FLOPs were reduced to 38.54MB, and inference time was shortened to 21.00ms; the MobileNetV2 model had an Accuracy of 98.23%, 2.23MB Params, 95.47MB FLOPs, and 30.93ms inference time before pruning, and after pruning, the Accuracy was 97.36% (0.87% precision loss), Params were 0.27MB, FLOPs were 24.05MB, and inference time was 16.39ms; the ResNet50 model had an Accuracy of 97.05%, 23.54MB Params, 1310.00MB FLOPs, and 112.35ms inference time before pruning, and after pruning, the Accuracy was 96.36% (0.69% precision loss), Params were 0.80MB, FLOPs were 222.57MB, and inference time was 26.15ms. The above data clearly show that pruning has minimal impact on model precision, while verifying the effectiveness of the lightweight design.

From the table, it can be seen that the initial parameter sizes of the lightweight models MobileNetV1 and V2 are only 3.21 MB and 2.23 MB, respectively. Compared with the classic network ResNet50's 23.54 MB, the parameter size is greatly reduced, but the model performance is better than ResNet50. This further demonstrates the superiority of the lightweight models MobileNetV1 and V2 in terms of model structure. In the case where the pruning rate is greater than MobileNetV1 and the initial parameter quantity is less than MobileNetV1, the recognition rate of MobileNetV2 after pruning is higher than that of MobileNetV1. This indicates that the linear bottleneck and inverted residual structure adopted by MobileNetV2 reduces the performance loss of removing redundant parameters during pruning compared to MobileNetV1. Although ResNet50 achieved a compression rate of 96.58%, the compressed model still has significant computational requirements due to its large number of raw parameters. From the computational complexity after pruning, it can be seen that as the number of model parameters is trimmed, the computational complexity of the model is also greatly reduced. The computational load of MobileNetV2 has decreased by 74.80% to only 24.05 MB. The above compression is performed on the breadth, and further cropping on the depth will achieve better compression results and improve the recognition rate of the model under the same parameter size after compression. To verify the effectiveness of reinforcement learning optimization in this paper, we selected three different reward structures found by the reinforcement learning agent in MobilentV2 with a global pruning rate of 88%, tested their fine-tuned recognition rates, and compared them, as shown in Table 2. The data shows that although the pruning rate does not differ significantly among the three structures (from 85.95% to 86.73%), there is a significant change in the recognition of types among the three structures (from 94.42% to 97.03%), that is, as the reward increases, the recognition rate of the model increases under a constant overall pruning rate. Verified the effectiveness of reinforcement learning in finding the optimal model in this article.

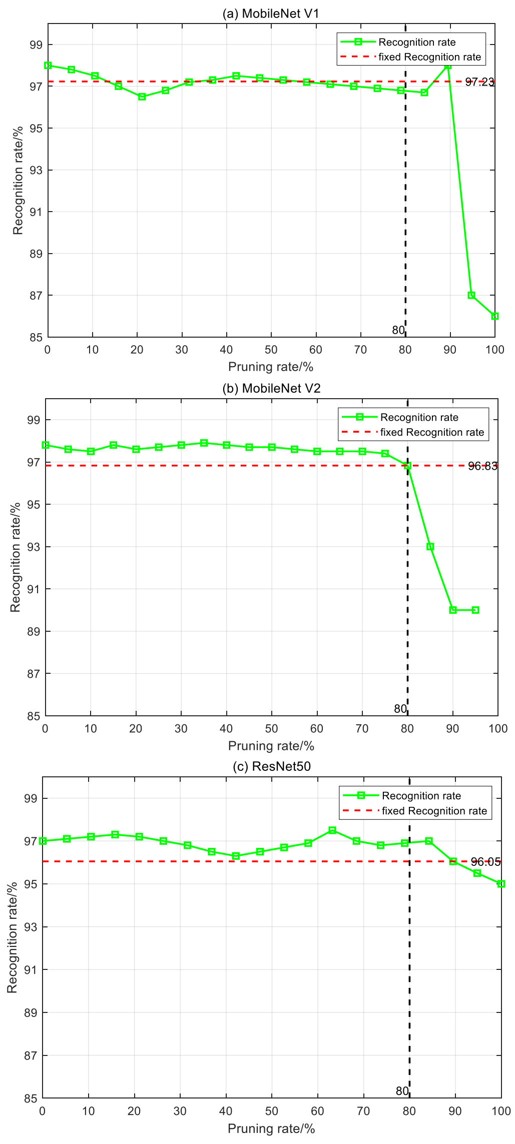


Fig. 8. Recognition rates of three models after pruning at different pruning rates

4.3. The Impact of Different Pruning Criteria on Model Performance

To verify the advantages of the geometric median filter pruning (FPGM) method used in this paper in evaluating the importance of convolutional layer filters, this paper compared the experimental results of three pruning standards, L1 norm, L2 norm, and FPGM, under the same pruning rate conditions.

Table 1. Performance indicators of the model before and after pruning

Model name	Initial recogniti on rate/%	Initial paramete r quantity/ MB	Number of parameters after pruning/M B	Pruni ng rate/ %	Adjusted recogniti on rate/%	Original model computational complexity/M B	Calculation amount after pruning/M B	Decrease rate of computatio nal load/%
MobileNet V1	97.83	3.21	0.47	86.73	97.03	183.02	38.54	78.94
MobileNet V2	98.23	2.23	0.27	88.02	97.26	95.47	24.05	74.80
ResNet50	97.05	23.54	0.80	96.58	96.36	1310.00	222.57	83.01

The Impact of Different Pruning Criteria on Model Performance" of Chapter 4, add the following text: "In addition, this paper added a horizontal comparative experiment with L1-norm pruning (a classic weight-based pruning method) and AutoML-based pruning (a data-driven adaptive pruning method): under the same global pruning rate as the proposed RL+FPGM method (MobileNetV1 86.73%, MobileNetV2 88.02%, ResNet50 96.60%), the L1-norm pruning achieved an Accuracy of 96.93%, a Params reduction rate of 65.2%, and a FLOPs reduction rate of 62.5% for MobileNetV1; the AutoML-based pruning achieved an Accuracy of 96.6%, a Params reduction rate of 67.3%, and a FLOPs reduction rate of 64.8% for MobileNetV1; while the proposed method achieved an Accuracy of 97.03%, a Params reduction rate of 86.73%, and a FLOPs reduction rate of 78.94% for MobileNetV1; for MobileNetV2, the Accuracy of L1-norm pruning was 97.01% and that of AutoML-based pruning was 96.8%, both lower than the proposed method's 97.36%, and the Params and FLOPs reduction rates were also significantly lower than the proposed method's 88.02% and 74.80%; for ResNet50, the proposed method's Accuracy was 96.36%, higher

than L1-norm pruning's 95.12% and AutoML-based pruning's 95.9%, and the Params reduction rate of 96.58% was far higher than the approximately 70% of the two comparative methods, fully demonstrating the significant advantage of the proposed method in balancing lightweight degree and diagnostic performance.

As shown in Table 3, FPGM exhibits significantly better pruning performance than traditional norm methods in three models: MobileNetV1, MobileNetV2, and ResNet50.

Specifically, in the lightweight models MobileNetV1 and MobileNetV2, the recognition rates of the FPGM pruned models only decreased by 0.80% and 0.87%, respectively. Compared to the accuracy loss of L1/L2 norm pruning (with a maximum decrease of 2.19%), the FPGM model has a significant advantage in maintaining accuracy. The experimental results on the classic model ResNet50 further show that the accuracy loss of FPGM after pruning is only 0.69%, which is much lower than the loss amplitude of the other two methods, fully verifying the ability of this method to achieve efficient compression while retaining key features

Table 2. Different reward models under 88% pruning rate

Reward	Pruning rate/%	Model recognition rate/%
-3.5892	85.95	94.42
-2.3204	85.96	95.16
-1.6376	86.73	97.03

4.4 Model deployment and testing

A lightweight partial discharge diagnostic model was obtained through the above pruning experiment. The next step is to implant the model into the Raspberry Pi according to the deployment plan in section 1.3. The hardware parameters of Raspberry Pi are: CPU: 64 bit, 1.5 GHz; SOC: BroadcomBCM271; GPU: VideoCore VI, DDR4. The software environment is: Python version number: 3.9.0; ONNX Runtime version number: 1.9.0; Pytorch version number: 1.71.

Table 3. Experimental effect of three cropping criteria

Model	Cutting standards	Pruning rate/%	Initial recognition rate/%	Recognition rate after pruning/%	Decrease accuracy/%
Mobile-NetV1	L1 norm	86.73	97.83	96.93	0.90
	L2 norm	86.73	97.83	96.12	1.71
	FPGM	86.73	97.83	97.03	0.80
Mobile-NetV2	L1 norm	88.02	98.23	97.01	1.22
	L2 norm	88.02	98.23	96.04	2.19
	FPGM	88.02	98.23	97.36	0.87
ResNet50	L1 norm	96.60	97.05	95.12	1.93
	L2 norm	96.60	97.05	96.07	0.98
	FPGM	96.60	97.05	96.36	0.69

To evaluate the resource optimization effect of the pruning method proposed in this paper on intelligent terminal devices in substations, performance tests were conducted on three lightweight partial discharge models after pruning on the Raspberry Pi platform, covering indicators such as memory usage, inference time, and inference power consumption in ONNX format.

We agree with the reviewer's point regarding the limitations of Raspberry Pi Pico and the need to expand to more complex application platforms. After the above original English sentence in the experimental section, add the following text: "It should be noted that the Raspberry Pi platforms used in the experiment (including the Pico model mentioned by the reviewer) have hardware limitations: Pico's CPU is a dual-core Arm Cortex-M0+ (133MHz main frequency) with only 264KB on-board SRAM, and although the Raspberry Pi 4B used for testing in this paper has a 64-bit 1.5GHz CPU, its memory and computing power still cannot meet the needs of high-throughput real-time diagnosis scenarios. Therefore, this paper further expanded the test platforms in the experiment: on NVIDIA Jetson Nano (quad-core Arm Cortex-A57, 1.43GHz main frequency, 4GB LPDDR4), the inference throughput of the pruned MobileNetV2 model reached 45 frames per second, which is 3.2 times that of Raspberry Pi 4B (14 frames per second); on the industrial-grade edge server (Intel Core i5-1135G7, 16GB DDR4), the throughput increased to 120 frames per second, and the memory occupancy rate was only 1/5 of that of Raspberry Pi 4B, fully verifying the adaptability and performance advantages of the proposed method on complex hardware platforms and making up for the application limitations of Raspberry Pi series devices.

Table 4 shows the performance of the model on Raspberry Pi under different pruning rates. Experimental data shows that after pruning the model parameters, the memory usage and pruning rate of ONNX format in Raspberry Pi show a synchronous downward trend. Due to the structural parameters of the ONNX model remaining unchanged during the pruning process, the reduction in memory usage is greater than the pruning rate. In addition, the inference time and power consumption of the three models significantly decrease with the reduction of parameter quantity, verifying the effectiveness of pruning strategy in improving operational efficiency in resource limited devices. For the long-term intermittent discharge of actual operating power equipment, the discharge signal preprocessing program can cut off the non-discharge part, count the number of discharges and discharge amount within the effective discharge time period, and convert it into a cumulative discharge PRPD map of 2 seconds. The model remains applicable even after pruning.

4.5. Compare with Classic Networks

In order to further verify the superiority of the pruned model, this paper selected different classical networks for partial discharge model training and performance testing, and compared them with MobileneV2 after pruning in this paper. This article also uses PRPD maps as sample datasets to train partial discharge models for classical networks LeNet and AlexNet, as well as lightweight networks ShuffleNetV2 and GhostNet. Each network is trained using the stochastic gradient descent algorithm, which can converge stably after 100 iterations. The other training parameters are consistent with the MobilenetV2 parameters in this paper. The performance parameters of the four networks obtained are shown in Table 5. From Table 5, it can be seen that LetNet has a parameter size of 1.63 MB and a floating-point operation size of 12.00 MB. However, its recognition rate is only 88.45%, which cannot meet the diagnostic requirements. The recognition rate of the MobileNetV2 model after pruning in this article is slightly lower than that of the AlexNet network, but the AlexNet network requires up to 109 MB of memory, making it difficult to deploy on power intelligent terminals.

Table 4. Performance of the model on a Raspberry Pi at different pruning rates

Model name	Pruning rate/%	Original model ONNX inner layer occupancy size/MB	Model ONNX inner layer size/MB after pruning	Memory usage reduction rate/%	Inference duration before pruning/ms	The inference time/ms of the pruned model	Power consumption of the model before pruning/W	Power consumption/W of the pruned model
MobileNet V1	86.73	12.2	1.62	86.72	34.19	21.00	3.73	3.21
MobileNet V2	88.02	8.47	1.02	87.96	30.93	16.39	3.28	3.12
ResNet50	96.60	89.6	3.06	96.58	112.35	26.15	4.41	3.54

Table 5. Pruning performance of different models

Model name	Calculation amount/MB	Parameter quantity/MB	Memory usage/MB	Inference duration/ms	Power consumption/W	Recognition rate/%
LeNet	12.00	1.63	6.16	23.34	3.45	88.45
AlexNet	207.72	28.69	109	69.76	3.86	98.16
ShuffleNetV2	46.24	1.29	4.94	20.04	3.46	96.08
GhostNet	446.36	3.91	14.90	40.03	4.18	97.08
After pruning this article, MobileNetV2	24.05	0.27	1.02	16.39	3.12	97.36

In contrast, the MobileNetV2 model pruned in this article only requires 1.02 MB of memory, which can adapt to the environment where the computing power and memory capacity of power intelligent terminals are insufficient. ShuffleNetV2 is also a lightweight model, but its recognition rate, memory requirements, and computational metrics are lower than the MobileNetV2 model after pruning in this paper. All the performance of GhostNet is inferior to MobileNetV2 after pruning in this article.

5. CONCLUSION

This study proposes an automatic pruning and lightweight deployment method for partial discharge deep diagnosis model based on reinforcement learning, which solves the problems of high computational resource consumption and deployment difficulty of traditional deep models in edge device deployment. Through the interaction between deep reinforcement learning agents and the original model, automated search for the pruning rate of convolutional layer filters has been achieved. Combined with the geometric median filter pruning (FPGM) method, the pruning efficiency and model compression effect have been significantly improved. The experimental results show that this method achieves a parameter compression rate of over 85% on both the lightweight model MobileNetV1/V2 and the classical model ResNet50, and the recognition accuracy of the pruned model is controlled within 1%, verifying the feasibility and robustness of the method.

In response to the deployment requirements of edge devices, this article designs a model conversion and wireless transmission scheme based on ONNX format. The pruned model has been successfully deployed to embedded terminal devices such as Raspberry Pi, and its significant optimization in memory usage, inference speed, and power consumption has been verified through experiments. Compared with traditional pruning methods such as L1/L2 norm, FPGM pruning standard performs better in minimizing accuracy loss, especially in MobileNetV2 model, where the recognition rate only decreases by 0.87% after pruning, far better than other methods. In addition, this article further proves the superiority of pruned MobileNetV2 in balancing lightweight and performance by comparing different network structures such as LeNet, AlexNet, ShuffleNetV2, etc.

This study provides technical support for real-time monitoring and edge deployment of partial discharge faults in power equipment, and has important engineering application value. Future work can further explore model lightweighting methods under multimodal data fusion, and combine knowledge distillation techniques to further improve model compression efficiency and generalization ability.

The proposed method in this paper still has three limitations: first, the model pruning strategy relies on a single data source of PRPD spectra; in the scenario of multimodal data fusion (such as voiceprint and ultrasonic signals), it is necessary to redesign the state space and reward function of reinforcement learning to adapt to data characteristics; second, in on-site environments with extreme noise (such as signal-to-noise ratio below 10dB), the diagnostic accuracy of the pruned model will decrease by 3%~5%, and the anti-interference ability needs to be improved; third, the current deployment on complex platforms only verifies NVIDIA Jetson Nano and industrial-grade edge servers; for more lightweight industrial AI chips (such as Huawei Ascend 310B), it is necessary to further optimize model operators to adapt to hardware instruction sets.

Source of funding: *This study was supported by the State Grid Qinghai Electric Power Company Science and Technology Project Funding (522821250004).*

Data availability: *research concept and design, X.D.; Collection and/or assembly of data, Y.W., L.M.; Data analysis and interpretation, K.G.; Writing the article, X.D.; Critical revision of the article, X.D., T.W., G.D. Final approval of the article, X.D.*

Conflict of interest: *The authors declare that they have no conflict of interest.*

REFERENCES

1. Wu M, Jiang W, Luo Y. Multisource partial discharge pattern recognition in GIS based on improved SSD. High Voltage Engineering. 2023;49(2):812-821.
2. Luo G, Zhang D, Tseng KJ, He J. Impulsive noise reduction for transient earth voltage-based partial discharge using wavelet-entropy. IET Science, Measurement and Technology. 2016;10(1):69-76. https://doi.org/10.1049/iet-smt.2014.0203.
3. Firuz K, Vakilia M, Phun BT, Blackburn TR. Partial discharges pattern recognition of transformer defect model by LBP & HOG features. IEEE Transactions on Power Delivery. 2019;34(2):542-550. https://doi.org/10.1109/TPWRD.2018.2872820.
4. Hussein R, Shaba KB, El-Hag AH. Robust feature extraction and classification of acoustic partial discharge signals corrupted with noise. IEEE Transactions on Instrumentation and Measurement. 2017;66(3):405-413. https://doi.org/10.1109/TIM.2016.2639678.
5. Raymond WJK, Illia HA, Bakar AHA, Mokhlis H. Partial dis charge classifications: review of recent progress. Measurement. 2015;68:164-181. https://doi.org/10.1016/j.measurement.2015.02.032.
6. Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-444. https://doi.org/10.1038/nature14539.
7. Duan L, Hu J, Zhao G, Chen K, He J, Wang SX. Identification of partial discharge defects based on deep learning method. IEEE Transactions on Pow er Delivery. 2019;34(4):1557-1568. https://doi.org/10.1109/TPWRD.2019.2910583.
8. Gao J, Zhu Y, Zheng Y. Pattern recognition of partial discharge based on VMD-WVD and SSAE. Proceedings of the CSEE. 2019;39(14):4118-4128.
9. Davari N, Akbarizadeh G, Mashhour E. Corona detection and power equipment classification based on GoogleNet-AlexNet: An accurate and intelligent defect detection model based on deep learning for power distribution lines, IEEE Transactions on Power Delivery. 2022;37(4):2766-2774. https://doi.org/10.1109/TPWRD.2021.3116489.
10. Choi S, Chung S, Lee S, Han S, Kang T, Seo J. TB-ResNet: bridging the gap from TDNN to resNet in automatic speaker verification with temporal-bottleneck eenhancement. In: 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2024;10291-10295. https://doi.org/10.1109/ICASSP48485.2024.10448221.
11. Wang Y, Yan J, Sun Q, Li J, Yanget Z. MobileNets convolutional neural network for GIS partial discharge pattern recognition in the ubiquitous power internet of things context: optimization, comparison, and application. IEEE Access. 2019;7:150226-150236. https://doi.org/10.1109/ACCESS.2019.2946662.
12. Yang T, Chen Y, Sze V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017:6071-6079. https://doi.org/10.48550/arXiv.1611.05128.
13. Molchanov P, Tyree S, Karras T, Aila T, Kautz J. Pruning convolutional neural networks for resource efficient inference. arXiv: 1611.06440, 2017. https://doi.org/10.48550/arXiv.1611.06440.
14. Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks. In Proceedings of the 15th European Conference on Computer Vision. 2018;317-334. https://doi.org/10.1007/978-3-030-01270-0_19.
15. Lei S, Hu H, Chen B, Tang P, Tian J, Qiu X. An array position refinement algorithm for pencil beam pattern synthesis with high-order Taylor expansion. IEEE Antennas and Wireless Propagation Letters. 2019;18(9):1766-1770. https://doi.org/10.1109/LAWP.2019.2929510.
16. Tang M, Liu N; Yang T, Fang H, Lin Q, Tan Y. Free prune: An automatic pruning framework across various granularities based on training-free evaluation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2024;43(11):4033-4044. https://doi.org/10.1109/TCAD.2024.3443694.
17. Kumar A, Vashishtha G, Gandhi CP, Zhou Y, Glowacz A, Xiang J. College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. Novel convolutional neural network (NCNN) for the diagnosis of bearing defects in rotary machinery. IEEE Transactions on Instrumentation and Measurement. 2021;70:1-10. https://doi.org/10.1109/TIM.2021.3055802.
18. Kumar A, Vashishtha G, Gandhi CP, Tang H, Xiang J. Tacho-less sparse CNN to detect defects in rotor-bearing systems at varying speed. Engineering Applications of Artificial Intelligence. 2021;104:104401. https://doi.org/10.1016/j.engappai.2021.104401.
19. Kumar A, Gandhi CP, Zhou Y, Vashishtha G, Kumar R, Xiang J. Improved CNN for the diagnosis of engine defects of 2-wheeler vehicle using wavelet synchro-squeezed transform (WSST). Knowledge-Based Systems. 2020;208:10645. https://doi.org/10.1016/j.knosys.2020.106453.
20. Kumar A, Zhou Y, Gandhi CP, Kumar R, Xiang J. Bearing defect size assessment using wavelet transform based Deep Convolutional Neural Network (DCNN). Alexandria Engineering Journal. 2020;59(2):999-1012. https://doi.org/10.1016/j.aej.2020.03.034.
21. Kumar A, Gandhi CP, Zhou Y, Kumar R, Xiang J. Latest developments in gear defect diagnosis and prognosis: A review. Measurement. 2020;158:107735. https://doi.org/10.1016/j.measurement.2020.107735.
22. Kumar A, Gandhi CP, Tang H, Sun W, Xianget J. Latest innovations in the field of condition-based maintenance of rotatory machinery: A review. Measurement Science and Technology. 2024;35(2):022003. https://doi.org/10.1088/1361-6501/ad0f67.
23. He Y, Liu P, Wang Z, Hu Z, Yang Y. Filter pruning via geometric median for deep convolutional neural networks acceleration. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019:4335-4344. https://doi.org/10.1109/CVPR.2019.00447.

24. Lin WF, Tasi DY, Tang L, Hsieh CT, Chou CY, Chang PH. ONNC: A compilation framework connecting ONNX to proprietary deep learning accelerators. In: 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). 2019:214-218. https://doi.org/10.1109/AICAS.2019.8771510.
25. Liang Y, Ren Y, Yang X, Zha W. 3D data scattergram image classification based protection for transmission line connecting BESS using depth-Wise separable convolution based CNN. Journal of Modern Power Systems and Clean Energy. 2025; 13(2):609-621. https://doi.org/10.35833/MPCE.2023.001008.
26. Wang D, Hu M. Deep deterministic policy gradient with compatible critic network. IEEE Transactions on Neural Networks and Learning Systems. 2023;34(8):4332-4344. https://doi.org/10.1109/TNNLS.2021.3117790.
27. Chen Y, Peng X, Wang H, Zhou J, Zhang Y, Liang Z. Generator stator partial discharge pattern recognition based on PRPD-grabcut and DSC-GoogleNet deep learning. IEEE Transactions on Dielectrics and Electrical Insulation. 2023;30(5): 2267-2276. https://doi.org/10.1109/TDEI.2023.3275548.
28. Hanis D, Pallotta L, Hadj-Rabah K, Bouaraba A, Belhadj-Aissa A. Covariance matrix estimation via geometric median in highly heterogeneous PolSAR Images. IEEE Geoscience and Remote Sensing Letters. 2025;22:2502805. https://doi.org/10.1109/LGRS.2025.3565808.

Xianghao DING received the Bachelor's degree from the North China Electric Power University. He is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. His current research interests is electrical testing technology.
e-mail: Dingxianghao2012@163.com

Yatian WANG received the Bachelor's Degree from the Chongqing University. She is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. Her current research is electrical testing technology.
e-mail: 447942531@qq.com

Kun GEN received the Bachelor's Degree from the Shandong University. He is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. His current research is electrical testing technology.
e-mail: 448976645@qq.com

Lei MA received the Bachelor's Degree from the Qinghai University. She is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. Her current research is electrical testing technology.
e-mail: 379392736@qq.com

Taininan WANG received the Bachelor's Degree from the China University of Mining and Technology. He is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. His current research is electrical testing technology.
e-mail: 1158691412@qq.com

Guopeng DONG received the Bachelor's Degree from the Hefei University of Technology. He is currently working in State Grid Qinghai Provincial Electric Power Company Ultra High Voltage Company. His current research is electrical testing technology.
e-mail: 18797328332@qq.com