processes

Article

A Lightweight Identification Method for Complex Power Industry Tasks Based on Knowledge Distillation and Network Pruning

Wendi Wang [1,*], Xiangling Zhou [2], Chengling Jiang [3], Hong Zhu [1], Hao Yu [1] and Shufan Wang [1]

[1] Nanjing Power Supply Branch, State Grid Jiangsu Electric Power Co., Ltd., Nanjing 210019, China; irisyhchoice@163.com (H.Y.); 13645151376@163.com (S.W.)
[2] State Grid Hubei Electric Power Co., Ltd., Wuhan 430077, China
[3] State Grid Jiangsu Electric Power Co., Ltd., Nanjing 210024, China; chengling_jiang@163.com
* Correspondence: wangwd2@js.sgcc.com.cn

Abstract: Lightweight service identification models are very important for resource-constrained distribution grid systems. To address the increasingly larger deep learning models, we provide a method for the lightweight identification of complex power services based on knowledge distillation and network pruning. Specifically, a pruning method based on Taylor expansion is first used to rank the importance of the parameters of the small-scale network and delete some of the parameters, compressing the model parameters and reducing the amount of operation and complexity. Then, knowledge distillation is used to migrate the knowledge from the large-scale network ResNet50 to the small-scale network so that the small-scale network can fit the soft-label information output from the large-scale neural network through the loss function to complete the knowledge migration of the large-scale neural network. Experimental results show that this method can compress the model size of the small network and improve the recognition accuracy. Compared with the original small network, the model accuracy is improved by 2.24 percentage points to 97.24%. The number of model parameters is compressed by 81.9% and the number of floating-point operations is compressed by 92.1%, making it more suitable for deployment in resource-constrained devices.

Keywords: power industry; service identification; knowledge distillation; network pruning

check for updates

Citation: Wang, W.; Zhou, X.; Jiang, C.; Zhu, H.; Yu, H.; Wang, S. A Lightweight Identification Method for Complex Power Industry Tasks Based on Knowledge Distillation and Network Pruning. *Processes* **2023**, *11*, 2780. https://doi.org/10.3390/pr11092780

Academic Editor: Xiong Luo

Received: 3 August 2023
Revised: 7 September 2023
Accepted: 8 September 2023
Published: 18 September 2023

1. Introduction

As new power systems continue to advance, more prominent "double new" features, load aggregators, integrated energy systems, virtual power plants, other new ecological emergences, and network boundaries will become more complex and control services will continue to extend to the end of the line. The demand for communication channels for new power systems is growing, and the requirements for communication network coverage, operation reliability, access flexibility, and the network performance index are becoming more stringent. In addition, the role of communication in the construction of new power systems is becoming more and more important. Terminal service identification refers to the technology of identifying the application type of data stream through the features in network data packets. In the rapid development of applications and the network size of new power systems, terminal service identification has become an important part of network management and security defense. In terms of network management, terminal service identification can provide enterprises with a detailed analysis of network usage, help them understand the use of the internal network, identify network bottlenecks, optimize network performance, and improve network usage and work efficiency. In terms of network security and defense, terminal service identification can help network administrators discover and respond to abnormal traffic and malicious attacks in the network in a timely manner so as to improve the level of network security and ensure network stability and data security.

At present, in the field of service traffic identification research, especially in the field based on machine learning, the main focus is on the accuracy of service traffic identification and the accuracy of identification for a specific service, ignoring the processing capability of the actual service and whether the identification technology is applicable to various types of equipment with limited resources. For example, deep packet inspection technology is a technology that has been developed more thoroughly and has better results in practical applications, utilizing the contents of packet messages for feature string matching. However, deep packet inspection techniques require a large amount of hardware and computational power and are less effective in recognizing unknown feature strings and encrypted data streams. Therefore, it is necessary to investigate more efficient, accurate, and practical service traffic identification techniques to meet practical requirements.

Machine learning is becoming more and more popular in the field of service traffic identification due to its continuous development. Compared with traditional methods, machine learning-based algorithms can avoid the content parsing of encrypted traffic, which is a difficult problem encountered by traditional traffic classification methods. In recent years, deep packet detection based on machine learning has become a hot issue in the field of measurement [1,2]. According to the characteristics of the input model, service identification methods based on machine learning can be divided into two classification methods based on the data flow features and deep learning.

A power marketing audit business identification model based on K-mean clustering and text categorization was proposed in [3]. The main idea of the qualitative analysis of the marketing work quality evaluation based on marketing results introduces, in detail, the process of constructing a qualitative analysis model of marketing quality and application examples. However, some power IoT access layer network traffic anomaly identification methods have the problem of low F1 value when facing unknown types of traffic. For this reason, Ref. [4] designed an extreme learning machine-based network traffic anomaly identification method to power the IoT access layer to network traffic. To identify transmission segments in the power system, the shortest distance and power sensitivity index were adopted together to form a comprehensive clustering index, and then the partitioning method was used to identify transmission segments in the power system [5]. In [6], a large data volume P2P video streaming service was divided into small data volume units and machine learning algorithms were used to identify each data unit. Finally, the final identification result was derived by statistical weighting. A novel hybrid machine learning approach for classifying network service traffic was also proposed in [7] which effectively recognized multi-application traffic through multi-criteria decision trees with attribute selection.

The advantage of the deep learning-based approach to service identification is that it does not have to manually extract features, eliminating the complex process of feature extraction and avoiding the loss of information entropy caused by human feature extraction. The disadvantage is that the performance of the system is largely limited by the size of the network; the large-scale network model occupies a large number of computer resources, and it is difficult to deploy in some of the first resource equipment. Deep learning-based approaches for service traffic identification typically extract the byte information of packets and convert it into a one-dimensional vector or a two-dimensional matrix to be used for model training and testing. With the outstanding feature of automatic feature extraction, the application of a deep learning to network service traffic classification has become a hot research topic. The authors of [8] designed a two-phase network traffic classification algorithm based on the network traffic features group. The first phase in the rapid clustering and empirical attribute selection was based on the idea of the coarse classification of network data traffic. The second phase involved the use of an integrated learning approach on the first phase of the classification of a subset of fine-grained identification. The experimental results showed that the two-phase recognition algorithm could effectively improve the recognition accuracy and recognition time used. A lightweight deep learning-based business recognition framework [9] can cope well with the problem of recognizing encrypted traffic and requires fewer storage resources. The authors of [10] designed a deep learning model to cope with encrypted mobile application traffic which required only the first few packet payloads to complete the recognition. In addition, a CNN-based encrypted traffic algorithm [11] converted the traffic data into a two-dimensional image format and employed an image classification algorithm to identify business traffic, providing better classification results with a 99.7% accuracy compared to the method using a one-dimensional vector data format.

As the performance of computer hardware improves, more and more research is focusing on deep neural networks, which have been found to extract richer and more abstract semantic information. However, as the size of the network increases, the training of neural networks becomes more difficult because the gradient vanishing problem may occur. To solve this problem, researchers have proposed techniques such as residual networks. In addition, there are a large number of redundant parameters in large-scale deep neural networks which require more storage and computational resources. With the popularity of embedded devices, how to deploy efficient and accurate service recognition systems on such devices with limited arithmetic and resources is also a pressing issue for researchers.

In order to address the problem of deep neural networks consuming large amounts of memory and energy in deployment and prediction, researchers have proposed a variety of parameter compression methods which are broadly categorized into the following categories, quantization, factorization, network pruning, and knowledge distillation.

Network pruning refers to reducing the computational effort of a network model by removing redundant parameters from the network, thus improving the inference speed and generalization ability of the network. In convolutional neural networks, network pruning is often used to reduce the storage space and computational resource consumption of the model. Combining quantization and pruning can further reduce the storage space and computational resource requirements. HashedNet [12] is a method that randomly groups network connections into hash buckets and shares the weights between connections in the same hash bucket. However, the sparsely connected approach does not necessarily improve the inference speed, so the method of cropping complete convolutional templates instead of cropping individual connections has become a hot research topic. This approach enables dense matrix multiplication without the use of sparse convolutional libraries. In addition, the algorithm combining quantization and pruning can further reduce the demand for storage space and network computational resources [13], in which the size of VGG-16 can be reduced by a factor of three from 49 MB to 552.11 MB without a loss of accuracy. It should be noted that for some application scenarios with high accuracy requirements, network pruning and quantization can lead to the problem of accuracy loss. However, several studies have used structured sparse training strategies [14,15] which effectively improve the operation speed of the network and reduce the dependence on hardware devices. These strategies can be pruned according to the sparsity while maintaining the structural integrity of the network and avoiding the loss of accuracy.

Since Hinton, the godfather of AI, proposed knowledge distillation [16], researchers have been working on it and have made significant progress. Knowledge distillation refers to the use of soft goals of the teacher network to guide the training of the student network in order to achieve model compression. In this process, the student network learns the knowledge of the teacher network, optimizes it with its own model, and updates the weights in the student network, which makes the outputs of the student model and the teacher's model closer to each other, thus improving the recognition accuracy of the student model. Through knowledge migration, the service traffic accuracy of the model can be dramatically compressed at the expense of a small amount of accuracy loss. Recent studies have shown that multiple teacher networks can effectively improve the performance of student models by distilling to form a multi-branch student network [17]. The method improves classification accuracy and model generalization, especially with limited training data. The experiments also show that the EKD-based compact network outperforms EKD on the test dataset in terms of an average accuracy of 89.66% compared to other methods based on knowledge distillation. In addition, employing a loop training strategy to train multiple networks simultaneously and weakening the relationship between instruction and learning can further optimize the teacher–student strategy [18]. To address the problem of low resolution, a teacher–student strategy using the same architecture was proposed in [19] which can be applied to train images with different resolutions. Also, some improvements have been made to the traditional distillation methods for the semantic segmentation task, such as using an association adaptation module to enable the student model to obtain and extract more information when learning about the teacher's knowledge [20]. It can improve the performance of a student network by 2.5% and can train a better compact model with only 8% float operations (FLOPS) of a model that achieves comparable performances.

Some lightweight modeling methods based on VGG-16 also have high accuracy and compression rates, as shown in recent studies [21–24]. In [21], a model pruning method incorporating a scaling factor and mutual information value order was proposed for the lightweight research of models in target detection scenarios. A polarized L1 regular term was used instead of the L1 regular term for the sparse training of the scaling factor, and the mutual information values of the channels and labels were computed on the experimental set by aggregating the channel features through global pairwise pooling. The pruning of VGG-16 reduces the number of parameters by 85.9% and computation by 58.2%, with a 0.45% loss of accuracy, without significantly degrading the network performance. Ref. [22] combined the original VGG-16 with the fully convolutional model, reduced the parameters of the model and the number of layers of the fully connected layer, and adjusted the super parameters of the model; the model still had a high accuracy of 95% for the remote sensing images with ultra-low pixels and fewer feature points. Ref. [23] designed a lightweight convolutional neural network based on the VGG-16 network structure, replacing traditional convolutional units with lightweight convolutional units, and making lightweight improvements on a large deep convolutional neural network. The results showed that the method outperformed other state-of-the-art lightweight convolutional neural networks and had an accuracy of 93.27%. The authors of [24] built a lightweight recognition model by structural optimization and the pruning operation of VGG16Ne and constructed a surface defect evidence set for training and prediction. Compared with the original VGG16 network, the improved network achieved better performances on the accuracy and compression rate.

However, the compression methods described above are only used in isolation, and they can only achieve a certain level of compression. That is to say, given a fixed student network, it is not possible to use an arbitrarily large network of teachers; in other words, a teacher can effectively transfer his/her knowledge to a certain size of students rather than a smaller one. Therefore, a multi-step knowledge distillation approach [21] was introduced to bridge the gap between students and teachers using an intermediate-sized network.

Therefore, in this paper, we propose a lightweight service identification model algorithm based on knowledge distillation and network pruning. First, the network model is trained with a large number of data samples. Before training the small network, the weights that have the least influence on the network classification are pruned through pruning judgment to achieve the effect of reducing the fitting as well as the model compression. After pruning the small network model through the loss function to fit the pre-trained large network output soft label information, a cross-entropy is made between the hard labels of the pruned small network and the large network, and the above two losses are weighted to form a loss function, as the loss function for the training of the pruned small network, to complete the training of the network. Through the migration of knowledge from the large network, the accuracy of the pruned small network is restored and improved. Simulation results show that this method can realize the further compression of the small network model and the migration of the knowledge of the large-scale network so that the performance of the pruned small network is close to that of the large network.

2. System Model

Previous researchers usually focused on the accuracy of the model and ignored the problem of model size in the study of deep learning-based service recognition algorithms. However, with the increasing complexity of network services, deeper and wider network structures have been introduced to improve the recognition accuracy of the models, which poses a great challenge for some resource-constrained devices to deploy service recognition systems. In practice, large neural networks may require millions or even billions of parameters for the best performance, and such models usually require huge computational resources and expensive hardware to run. Small neural networks, on the other hand, can run faster with limited resources and can accommodate many low-power devices such as cell phones and IoT devices. In order to train service traffic recognition models that have a high accuracy in resource-constrained scenarios, this paper employs two algorithms, knowledge distillation and network pruning, to lighten the deep learning models. By employing these algorithms, the model size can be significantly compressed to accommodate the deployment on resource-constrained devices without significant loss of model accuracy. The system block diagram is shown in Figure 1.

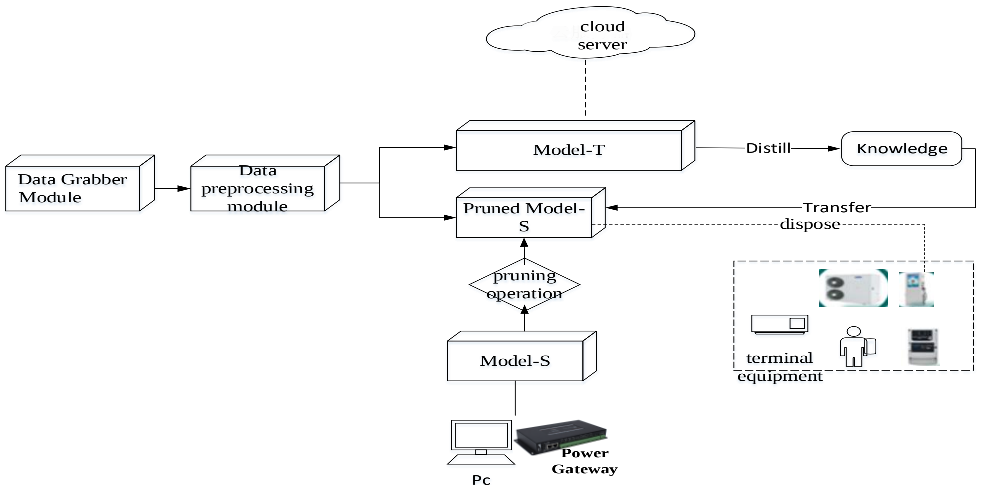

Figure 1. Block diagram of a complex power service lightweight identification system based on knowledge distillation and network pruning.

Figure 1 illustrates a service recognition system based on knowledge distillation and network pruning. In order to realize service recognition, raw data messages need to be processed before they are input into the system. For different sizes of the teacher network and student network, it is necessary to choose the appropriate data input format and process the raw data. Since the training time of the teacher network is long and requires a large amount of computational resources, pre-training is carried out in advance on a well-resourced cloud server, and the pre-trained teacher network is used to guide the student network to optimize the parameters to be faster and better. Before participating in knowledge distillation, the student network needs to go through a network pruning operation to remove the convolutional kernels that have less impact on the final classification results in order to reduce the parameter redundancy of the network and improve the inference speed of the model. At the same time, the loss of model accuracy needs to be reduced as little as possible to achieve high-precision service traffic recognition in resource-constrained scenarios. Ultimately, the student network with restored accuracy is deployed into the device to accomplish the corresponding service traffic recognition task.

To facilitate the reader's understanding, some of the modeling concepts are described below:

(1) Teacher model: referred to as Model-T, it is characterized by a relatively complex model with more parameters that can be obtained by integrating multiple distributed

models. For Model-T, there are no constraints on model architecture, latency, model size, etc. The only requirement is that for any input X, Model-T can output the result Y, where Y can be mapped by the Softmax layer to obtain the probability distribution size of the corresponding category.

(2) Student model: referred to as Model-S, it is characterized by a relatively simple model with fewer parameters. The requirements are the same as Model-T: for input X, the outputs are all Y. Y can be mapped by the Softmax layer to obtain the probability distribution of the corresponding category.

(3) Streamlined student model: abbreviated as Pruned Model-S, it is obtained by the network pruning of the student model. The model is lighter and has fewer parameters than Model-S. It should be noted that when pruning the parameters, it is necessary to ensure that the loss of accuracy is not too large.

(4) "Knowledge" transfer: Model-T has strong learning generalization ability and can transfer the learned knowledge to the relatively weak learning ability of the Pruned Model-S, which is used to enhance the generalization ability of the Pruned Model-S as well as to improve its recognition accuracy. Complex Model-T is not online, and only Pruned Model-S with low resource consumption and low latency is deployed when the model is deployed.

3. Algorithmic Process

Uses knowledge distillation methods to transfer knowledge from a large-scale teacher network to a small-scale student network, improving the student network's recognition capabilities without increasing the size of its own network. It helps to fulfill the conditions of resource constraints and enables the student network to be deployed in resource-constrained scenarios. The specific steps are shown in Figure 2.

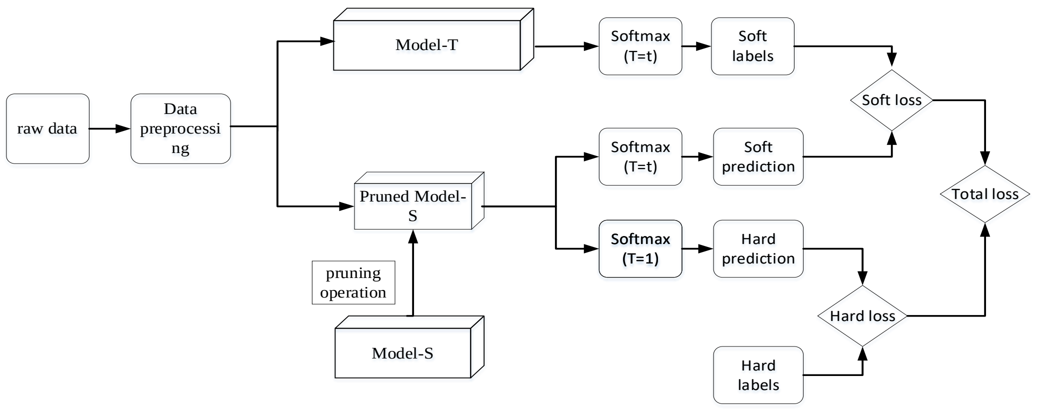

Figure 2. Flowchart of the two-stage algorithm.

The raw data are acquired by crawling through a specific environment built in the lab. In order to train a highly accurate and lightweight service recognition neural network model, the raw data messages need to be processed and preprocessed through a series of operations in order to be trained as inputs to the neural network model. Meanwhile, two important algorithms, namely, knowledge distillation and network pruning, need to be focused on while performing the training process. Before training, the teacher network needs to be pre-trained, because the teacher network usually has a larger model size and longer training time, so it is deployed on a well-resourced cloud server for pre-training, and the pre-trained teacher network is used to guide the student network to optimize the parameters faster and better. The student network needs to go through a pruning operation to remove redundant parameters and compress its own model size before knowledge distillation so that the final network model deployed into the device is more lightweight.

When performing training, it is necessary to pay attention to the loss function of the student network, part of which comes from the hard labeling between its own model predictions and the real service categories, and part of which comes from the soft labeling with the instructor's model prediction distribution. This makes the learning goal of the student network clearer and guides it to learn better about service recognition. The teacher network obtains the probability distribution of each input sample belonging to each category through forward propagation and uses it as the learning objective of the probability distribution of the student network, which is introduced into the training of the student network through the loss function. In this way, the student network is made to better learn the knowledge of the teacher network and improve the model accuracy.

3.1. Data Preprocessing

Data preprocessing is the process of processing raw data, which usually includes steps such as data cleaning, data integration, data conversion, and data statute. In this paper, the data preprocessing process is divided into two parts. The first half is the same as the third data processing method: first, the captured data packets need to be streamed, and then the byte information is extracted and converted.

Considering the temporal relationship between different packets in the same stream, the second half of the data preprocessing is improved in this paper. Here, the first 32 packets in a stream are selected and the first 32 bytes of each packet are taken and formed into a 32×32 2D image matrix. These processed data are closer to the form of an image which helps to better capture the temporal sequence between packets and thus improves the accuracy of service identification. The specific data processing flow is shown in Figure 3:

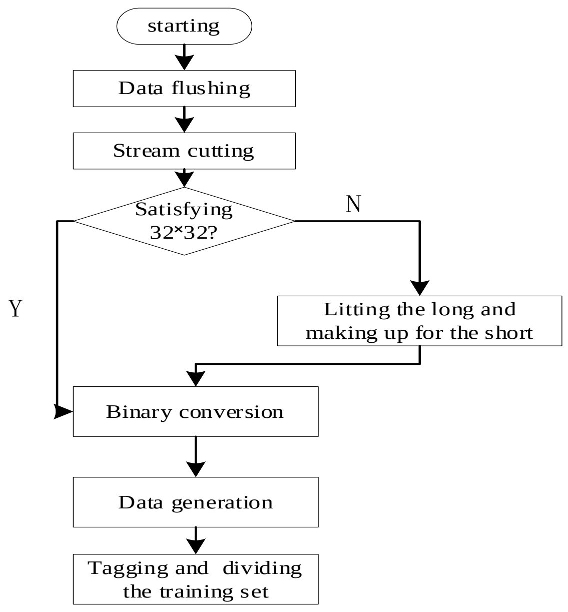

Figure 3. Data processing flow.

3.2. Construction of Teacher Networks and Student Networks

In the field of machine learning, in order to improve the performance and accuracy of models, some efficient model architectures or optimization techniques are usually used to enhance the performance of models. Among them, the design idea of a teacher–student network has been widely used in the design of deep neural networks. Teacher–student networks train and design models by allowing larger networks (teacher networks) to "teach" smaller networks (student networks). The teacher network usually has a larger network depth and number of parameters to learn richer feature information, while the student network gradually learns and extracts more critical feature information by comparing with the teacher network, thus achieving the effect of model streamlining and speeding up.

When choosing a teacher–student network, a series of factors need to be considered, such as model performance, complexity, and training speed. For some high-precision and high-complexity models, such as ResNet50, the use of teacher–student networks for streamlining and acceleration can significantly reduce the complexity and number of parameters of the model and improve the inference speed of the model while maintaining a certain level of performance. Therefore, multiple factors such as model performance, complexity, training speed, etc. need to be considered when choosing the teacher–student network in order to realize the streamlining and acceleration of the model while guaranteeing the performance of the model. Meanwhile, in the design of the teacher–student network, it is also necessary to consider the aspects of the selection of the teacher network and the student network, the design of the network structure, and the setting of the distillation loss in order to achieve the optimal model design.

(1) Selection of Teacher Models

Because pruning the student network has a certain degree of accuracy loss, this paper is based on the knowledge distillation algorithm to restore the accuracy of the pruned student network. As the teacher network is not involved in the deployment process of the model in the terminal equipment, there is no need to consider the size of the teacher model. Therefore, this paper analyzes and compares a variety of classical convolutional neural networks (CNN) using the same parameter settings and all training 20 epochs; the teacher network is then selected as the classical model in the CNN and makes a comparative comparison, as shown in Table 1:

Table 1. Comparison of classical CNN models.

Model	Accuracy Rate (%)	Quantity of Participants (Bytes)
VGG19	96.24	2.15×10^8
VGG16	95.15	1.50×10^7
AlexNet	92.37	2.40×10^8
ResNet18	95.46	1.12×10^7
ResNet50	97.36	2.35×10^7

When using a classical deep learning model, the data need to be processed to fit the input size of the model. For example, when using the ResNet50 model [25], the standard format of its data is 3-channel image data of 224 × 224 size, which can be adapted by changing the parameters of the input layer, and, at the same time, in the output layer, RsrNet50 defaults to 1000 categories, which needs to be modified to the number of categories of the data class in this paper. The format of the data can also be changed to adapt it to the structure of the input model. When processing the data, care needs to be taken to maintain the temporal order of the data between different packets in the same stream to improve the recognition accuracy of the service traffic. Generally speaking, a teacher network with higher accuracy can better guide students to learn, and the model of the teacher network is not involved in the final deployment process, so in this paper, ResNet50 is selected as the teacher network to guide the accuracy recovery of the pruned student network. The modified ResNet50 network structure is shown in Table 2.

(2) Selection of the student model

In order to make the deployment of the student network suitable for end devices, a relatively simple network structure needs to be designed. When designing a neural network, the number of layers of the network, the choice of convolutional layers, the pooling layer, the activation function, the loss function, and the optimizer need to be considered. In order to obtain the best network performance, several experiments are needed to continuously adjust and optimize the network. Based on the experimental results in this paper, a suitable network structure is designed as shown in Table 3.

Table 2. ResNet50 network structure.

Layer Name	Output Size	50-Layer
Conv1	16 × 16	7 × 7, 64, stride 2
Conv2	8 × 8	3 × 3 max pool, stride 2
		$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$
Conv3	4 × 4	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$
Conv4	2 × 2	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$
Conv5	1 × 1	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$
	1 × 1	Average pool, 11-d fc, softmax

Table 3. Network structure of the student model.

	Conv1	Conv2	Pool3	Conv4	Conv5	Conv6	Pool7	FC8	FC9
Convolution Kernel count	32	64	-	128	256	512	-	256	11
Convolution Kernel size	5	5	2	3	3	3	2	-	-
Pacemaker	1	1	2	1	1	1	2	-	-

The convolutional kernel size and number of kernels are key hyperparameters that need to be chosen based on the specific task and dataset. In general, the convolutional kernel size should become smaller and smaller in order to capture more detailed features of the data. Also, the number of kernels should be chosen based on the complexity of the dataset and the available computational resources. If the dataset is very complex and more features are needed to capture its structure, then more convolutional kernels can be selected. Also, the size and number of convolutional kernels can be adjusted by methods such as cross-validation to find the best combination of hyperparameters. It should be noted that too many convolutional kernels can lead to the problem of model overfitting, so the performance and generalization ability of the model need to be considered comprehensively when selecting hyperparameters. After experimental verification, for the data input of this paper in the format of a 32 × 32 two-dimensional matrix, selecting the size of the convolution kernel as 5 × 5 or 3 × 3 can obtain relatively good results.

The activation function serves to introduce a nonlinear element that improves the expressive power of the network and enhances the fitting ability of the network. In a neural network, each neuron receives a certain number of input signals and performs a weighted sum of these input signals to obtain a value. This value is then fed into an activation function which converts it into an output value for the neuron. By using activation functions, neural networks can represent more complex nonlinear models so that the network can better handle various types of input data such as images, text, audio, etc. Commonly used activation functions are sigmoid, tanh, ReLU, LeakyReLU, etc. Different activation functions have different characteristics, and choosing a suitable activation function can improve the performance of the network. After experimental verification, the network in this paper finally chose ReLU and Sigmod as the activation functions of the network.

3.3. Pruning Judgment

In order to further compress the model of the student network to solve the student network overfitting problem as well as to reduce the number of parameters of the student network, the network model size is reduced to facilitate the subsequent model deployment. The steps of pruning are shown in Figure 4:

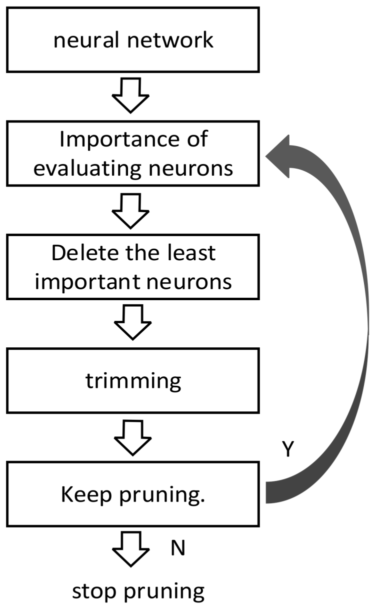
neural network
Importance of evaluating neurons
Delete the least important neurons
trimming
Keep pruning.
Y
N
stop pruning

Figure 4. Pruning process.

The pruning method chosen in this paper is convolutional kernel pruning whose advantage is that it can ensure that the number of parameters and computation of the neural network can be greatly reduced under the condition of high accuracy, so as to improve the running efficiency of the model and save storage space. Compared with neuron pruning and weight pruning, convolutional kernel pruning is more efficient because convolutional kernel pruning can cut out a whole convolutional kernel, while neuron pruning can only cut out a neuron, and weight pruning can only cut out a weight. In addition, convolutional kernel pruning results in easier hardware acceleration because the number of multiply–add operations can be reduced by decreasing the number of convolutional kernels, thus increasing the computational speed of the model. In addition, convolutional kernel pruning can also improve the generalization ability of the model, as pruning can induce the model to learn more general features and avoid overfitting. However, it should be noted that excessive pruning can lead to model underfitting, so the convolution kernel pruning needs to be adjusted according to the actual situation. In conclusion, convolutional kernel pruning is an effective model compression method that can significantly reduce the number of parameters and computation of the neural network, improve the operation efficiency of the model, and save storage space. At the same time, it can also improve the generalization ability of the model, but we need to pay attention to the appropriate control of the degree of pruning to avoid model underfitting.

Oracle pruning is a method of traversing all the convolutional kernels of a network model using the violent search method which uses the importance of the convolutional kernel as the criterion for pruning, judges the importance of the convolutional kernel by the degree of change in the loss function, and performs the deletion operation of unimportant convolutional kernels by exhaustive enumeration. Although the violent method can accurately remove redundant convolutional kernels, it has the disadvantages of a large number of computations and inefficiency. In this paper, a pruning method based on Taylor expansion is used to minimize the impact on cross-entropy and thus maximize the information retained by the neural network. This method is faster and more accurate than violent pruning. The evaluation function of this method can be derived by the following steps:

Assume that there is a mapping $D = \{X = \{x_0, x_1, \cdots, x_N\},\ Y = \{y_0, y_1, \cdots, y_N\}\}$, where x and y denote the target input and output of the model, respectively. CNN network models are denoted as $W = \{(w_1^1, b_1^1), (w_1^1, b_2^1), \cdots (w_L^{c_l}, b_L^{c_l})\}$, where (w^i, b^i) denotes a convolutional kernel, and the goal of pruning is to minimize the difference in the cost function before and after pruning. When pruning, to maintain the original accuracy of the network so that $C(D\Big|W') \approx C(D\Big|W)$, we need to solve a combinatorial optimization problem, as shown below, where $C(\cdot)$ denotes the cross-entropy cost function.

$$min|C\left(D\Big|W'\right) - C(D|W)\Big|s.t\ \Big|\Big|W'\Big||_0 \leq B \tag{1}$$

According to Taylor's formula, $f(x)$ can be expanded at $x = a$ as:

$$f(x) = \sum_{p=0}^{p} \frac{f^{(p)}(a)}{p!}(x-a)^p + R_p(x) \tag{2}$$

where $f^{(p)}(a)$ is the p-order derivative of $f(x)$ at $x = a$ and $R_p(x)$ is the residual term of the expansion.

Based on the Taylor expansion approach, we directly approximate the value of the change in the loss function caused by the removal of a parameter. Let h_i denote the output produced by parameter i. In the feature map, $h = \left\{z_0^{(1)}, z_0^{(2)}, \ldots, z_L^{(C\updownarrow)}\right\}$. For ease of representation, the parameters and the outputs corresponding to the parameters are considered to have the same effect on the value of the loss function: $C(D|h_i) = C(D|(w,b)_i)$. The parameters are assumed to be independent of each other:

$$|\Delta C(h_i)| = |C(D, h_i = 0) - C(D, h_i)| \tag{3}$$

where $C(C, h_i = 0)$ is the cost function after pruning and $C(D, h_i)$ denotes the cost function before pruning.

Similarly, assuming that the individual parameters of the convolution kernel are independent of each other, the first-order Taylor expansion of $C(C, h_i = 0)$ is:

$$C(D, h_i = 0) = C(D, h_i) - \frac{\delta C}{\delta h_i} h_i + R_1(h_i = 0) \tag{4}$$

Neglecting the residual term, bringing (4) into (3) yields:

$$\Theta_{TE}(h) = |C(D, h_i) - \frac{\delta C}{\delta h_i} h_i - C(D, h_i) = |\frac{\delta C}{\delta h_i} h_i| \tag{5}$$

The above equation represents the magnitude of the change in the loss function caused by the removal of a parameter. The computational ordering of this value enables the parameters corresponding to feature maps with relatively flat gradients to be removed. The pruning problem is then transformed into a search for the h_i that minimizes $|\Delta C(h_i)|$. The error between the results and the samples is represented by the cross-entropy loss function. ACH is the gradient of the error that is passed from the loss function to the layer during backpropagation and $\frac{\delta C}{\delta h_i}$ is the gradient of the error that is passed from the loss function to the h_i layer during backpropagation.

The specific formula for Θ_{TE} is:

$$\Theta_{TE}(z_l^{(k)}) = |\frac{1}{M} \sum_m \frac{\delta C}{\delta z_{l,m}^{(k)}} z_{l,m}^{(k)}| \tag{6}$$

where $z_l \in R^{H_l \times W_l \times C_l}$ is the feature map of the lth layer, H_l and W_l are the height and width, respectively, C_l is the number of channels, $z_l^{(k)}$ is the feature map outputted by the kth channel therein, and M is the number of $z_l^{(k)}$.

3.4. Improved Knowledge Distillation Methods

Neural network knowledge distillation is a technique for transferring knowledge from large neural networks to small neural networks. The technique aims to increase the speed and efficiency of model inference by reducing the number of parameters and computational burden of the model while maintaining the performance of the model. The purpose of knowledge distillation is to compress the knowledge from a large and accurate neural network into a smaller, less computationally resource-demanding neural network.

In practice, large neural networks may require millions or even billions of parameters for their best performance, and such models usually require huge computational resources and expensive hardware to run. Small neural networks, on the other hand, can run faster with limited resources and can be adapted to many low-power devices, such as cell phones and IoT devices. Therefore, converting large neural networks to small neural networks using knowledge distillation techniques can improve the utility of the model while maintaining its performance.

A typical solution to the classification problem is to use a Softmax function at the last layer of the network to obtain the probability distribution corresponding to each category. In the process of knowledge distillation, a model Model-T with strong generalization ability has been pre-trained, and when training Model-S on Model-T, the most direct way is to let Model-S learn the generalization ability of Model-T. One simple and effective way is to use the output of Model-T as a "soft label" to guide Model-S learning, which is called the "Soft-Target" method. Specifically, the Soft-Target method uses the category probability distribution of Model-T's output as the label for training Model-S, which can be expressed as shown in Equation (7):

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{7}$$

In traditional neural network training, a "Hard-Target" is used, i.e., all negative labels are processed uniformly, which results in the relative relationship between negative labels being partially ignored. In order to solve this problem, a temperature coefficient T can be used to generate a Soft-Target, where the larger T is, the "softer" the resulting probability distribution is, and the more information the negative labels carry. In the Softmax activation function, z_i is the ith component of Logits, and the category probabilities output from the Softmax activation function obtained by the temperature parameter T can be used as a Soft-Target. By increasing the value of T, the relative size of the probabilities of the negative labels is enlarged, which retains the information of the relative relationship before the negative labels.

The loss function of the knowledge distillation technique consists of two parts. The first part is the cross-entropy between the classification probabilities, where both Model-T and Model-S soften the probability distributions using the same temperature coefficient T. The second part is the cross-entropy loss between the probability distribution of Model-S and the true label of the service, where the temperature is set to 1. The two parts of the loss can be combined by weighted summation. Specifically, the form of the loss function shown in Equation (8):

$$\mathrm{L_{kd}} = \frac{1}{N}\sum_{i=1}^{N}\left[L_{CE}(y_i, \sigma(z^{S_i})) + T^2 \times L_{CE}(\sigma(z^{S_i}/T), \sigma(z^{T_i}/T))\right] \tag{8}$$

where N is the number of small batches of data at a time, L_{CE} stands for cross-entropy. $\sigma()$ stands for the Softmax function, T is the distillation temperature, which is used to soften the knowledge of the target, y_i is the true label of the ith sample, and z^{S_i} and z^{T_i} are the Logits that represent the output of the classification task Model-S and Model-T, respectively.

In traditional knowledge distillation methods, although the accuracy of Model-T is higher than that of Model-S, Model-T still suffers from prediction errors. When Model-T has a prediction error, Model-S learns this incorrect part of the knowledge, which affects the performance of Model-S. Therefore, this paper improves the traditional knowledge distillation approach by adding an indicator function to the original one which is used to constrain Model-T to pass only the correctly predicted distributions to the student network and ignore Model-T's wrong predictions. Specifically, the improved method in this paper only considers the cross-entropy between the correctly predicted probability distribution in Model-T and the probability distribution predicted by Model-S. The specific calculation formula is shown below:

$$L_{KD}^{*} = \frac{1}{N}\sum_{i=1}^{N}\left[L_{CE}(y_i, \sigma(z^{S_i})) + I(y_i, y_i^{\mathrm{T}}) \times T^2 \times L_{CE}(\sigma(z^{S_i}/T), \sigma(z^{T_i}/T))\right] \tag{9}$$

where $I(y_i, y_i^{\mathrm{T}})$ is an indicator function and y_i^{T} is the label predicted by Model-T. An indicator function is used to determine whether Model-T can correctly predict the classification of input samples. If Model-T can correctly classify the input samples, the Soft-Targets of its Soft-Targets and the Soft-Targets of the labels predicted by Model-S are used in the calculation of cross-entropy for the training of Model-S; if Model-T cannot correctly classify, only the classification of Model-S and the service are calculated in the cross-entropy between the real categories. The objective function of the method consists of two parts, i.e., the Soft-Target loss and cross-entropy loss. Among them, the indicator function is used to control whether the Soft-Target loss participates in the training or not, which can be regarded as a kind of weight control method.

4. Performance Analysis

4.1. Data Set Description

In this paper, we aim to deploy a lightweight service traffic identification system in a low-resource equipment environment, make identification tests for several common types of power services, build a pure network environment, and, when capturing, have the running equipment ensure that only the target service is running, so as to ensure the purity of the data as much as possible; the captured dataset is shown in Table 4 below.

Table 4. Data set description.

Application Type	Type Introduction	Service Category
Advanced measurement	Harvesting type	0
Electric vehicle charging	Harvesting type	1
Transformer condition sensing	Harvesting type	2
Distributed energy regulation	Control type	3
Precision negative control	Control type	4
Distribution network protection	Control type	5
Transmission line online monitoring and drone inspection	Mobile class	6

The above processing of the original dataset, as well as the above description, is briefly described here, and for each data stream, the first 32 packets are selected. Each packet is selected for its 32 bytes, which are stacked to form a 32 × 32 matrix form, as a training sample.

4.2. Evaluation Indicators

The following three aspects are used to evaluate the model in the experiments of this paper, namely, service recognition accuracy, model size, and single sample model inference time.

(1) The metric describing the service recognition accuracy is chosen as $Accuracy$, and the largest classification term in the classification probability predicted by the neural network is taken as the prediction result. The calculation formula is as follows:

$$Accuracy = \frac{TP + TN}{P + N} \tag{10}$$

where P and N denote the number of positive samples and the number of negative samples, respectively, and TP and TN denote the number of correct predictions of positive and negative samples, respectively.

(2) Parameters: the number of parameters is used to describe the number of parameters that need to be trained or optimized. These parameters include weights, biases, etc. In a convolutional neural network, the number of parameters mainly consists of the parameters of the convolutional layer and the fully connected layer, in which the parameters of the convolutional layer can be expressed by the following formula:

$$W_c = C_i \times C_0 \times k \times k + C_0 \tag{11}$$

where C_i and C_0 denote the number of channels of the input feature map and output feature map, respectively, and $k \times k$ denotes the size of the convolution sum. The additive C_0 denotes the bias term with a size equal to the number of output channels. For example, a 3 × 3 convolutional layer contains 10 parameters, which includes 9 parameters for the convolution operation and 1 parameter for the bias operation. The parameters of the fully connected layer in the neural network can be expressed by the following equation:

$$W_F = T_i \times T_o + T_o \tag{12}$$

where T_i denotes the one-dimensional feature vector of the input, T_0 denotes the one-dimensional feature vector of the output, and the additive T_0 also denotes the bias term. It can be understood that each output neuron is connected to all the input neurons, and each output neuron is added with a bias term to accelerate the fitting of the network. Therefore, the total number of parameters of the convolutional neural network in this paper is formulated as follows:

$$W_p = W_c + W_F \tag{13}$$

(3) FLOPs (floating point operations): Floating point operations are a metric used to measure the computational complexity of a model, i.e., an estimate of the number of floating point operations per second of the model. In deep learning, FLOPs are usually calculated to evaluate the computational cost, inference speed, and efficiency of a model. It is dependent on factors such as the number of parameters, layers, input sizes, and types of operations of the model; thus, it allows for more informed decision-making when training or deploying a model to achieve optimal computational efficiency. FLOPs are mainly generated in the convolutional layer and the fully-connected layer. Among them the convolutional layer FLOPs are calculated as follows:

$$F_c = 2 \times C_i \times k^2 \times C_0 \times H_0 \times W_0 \tag{14}$$

where $C_i \times H_i \times W_i$ denotes the input size and $C_0 \times H_0 \times W_0$ denotes the output size. The FLOPs of the fully connected layer in the neural network can be expressed as follows:

$$F_F = 2 \times C_i \times C_0 \tag{15}$$

Therefore, the total FLOPs formula for the convolutional neural network in this paper is as follows:

$$F_P = F_C + F_F \tag{16}$$

4.3. Analysis of Results

The dataset after the data preprocessing module is subjected to a shuffle operation, i.e., the order of the dataset is randomly disrupted as a way to avoid the phenomenon of the overfitting of the network model caused by the fixed order of the dataset. The training set is partitioned into 70% for model training and 30% for testing. The training accuracy as well as the loss of Model-T and Model-S are shown in Figures 5 and 6:

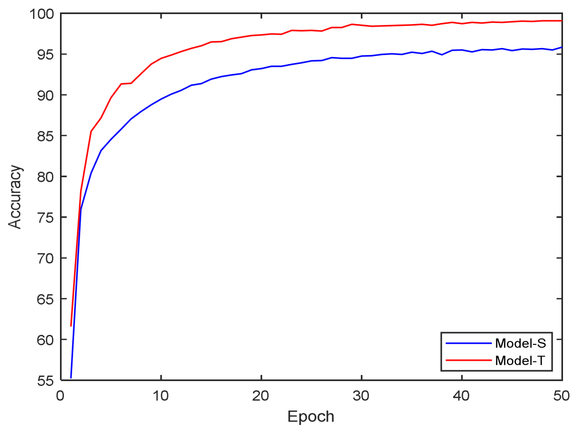

Figure 5. Comparison of the recognition accuracy between Model-S and Model-T.

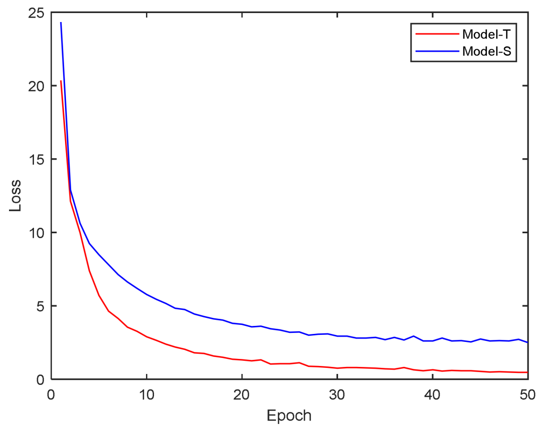

Figure 6. Comparison of Model-S and Model-T losses.

In the process of knowledge distillation, the student model tries to learn the decision basis and feature representation of the teacher model by observing the predictions of the teacher model. As the number of iterations increases, the student model gradually benefits from the knowledge of the teacher model. The parameters of the student model gradually converge to those that can more accurately mimic those of the teacher model and the student model becomes more attuned to the key features of the task and reduces overfitting, thus improving its predictive ability and leading to an increase in recognition accuracy. The simulation results show that the accuracy of both Model-S and Model-T gradually improves with the number of training iterations, i.e., the accuracy of the final converged value of Model-S is 95.84%, and the accuracy of the final converged value of Model-T is 99.07%. The loss of both networks decreases against the increase in the number of iterations, which indicates that the optimization of the model is improving. The reduced loss rate means that the simplified model is more successful in mimicking the predictive distribution of the complex model. By minimizing the loss function, the simplified model gradually learns from the complex model to perform better on similar tasks.

In order to further streamline the parameters of Model-S, it is pruned before the knowledge transfer between Model-S and Model-T, with the goal of reducing the excessive parameter redundancy in Model-S, accelerating the model derivation speed, and making it more suitable to be deployed in service scenarios with high requirements on accuracy and efficiency. Figure 7 shows the change in Model-S accuracy under different pruning ratios:

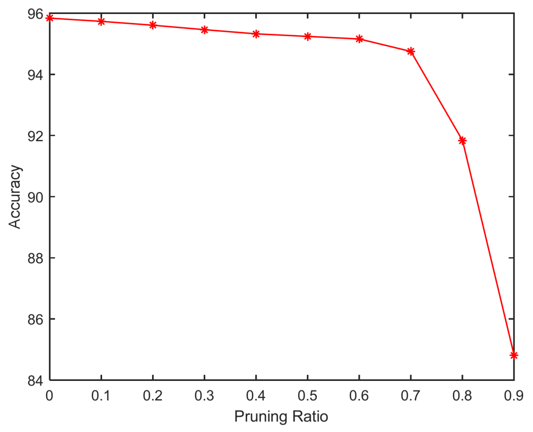

Figure 7. Effect of pruning ratio on Model-S recognition accuracy.

Although model pruning can reduce model parameters, the accuracy of the pruned model is often not as good as the original model. The simulation results show that when the pruning ratio is only 0.1, the Model-S recognition accuracy rate increases slightly, indicating that the appropriate amount of pruning can not only accelerate the inference speed of the model but also reduce the network redundancy and improve the network accuracy rate. When the pruning ratio decreases to 0.7, the network accuracy rate decreases less, indicating that the deleted convolutional kernel at this time has less impact on the network accuracy rate; with the pruning ratio increasing again, the network accuracy rate shows a cliff-like decline, indicating that the deleted convolutional kernel at this time has a greater impact on the network accuracy rate. In order to compress the network model as much as possible under the premise of guaranteeing accuracy, this paper chooses a pruning ratio of 0.7 as the student network after pruning to carry out the following accuracy recovery experiments. The accuracy of the model before pruning is 95.84%, and the accuracy of the model after convolution with a ratio of 0.7 is 94.45%. The loss in accuracy of the model before and after pruning is 1.39 percentage points. Table 5 lists the change in the number of convolution kernels before and after Model-S pruning:

Table 5. Changes in the number of convolution kernels before and after Model-S pruning.

Layer Index	Number before Pruning	Number after Pruning
Layer1	32	24
Layer2	64	37
Layer4	128	63
Layer5	256	72
Layer6	512	102

Before engaging in knowledge distillation, the student network needs to go through a network pruning operation to remove the convolutional kernels that have less impact on the final classification result in order to reduce the parameter redundancy of the network and improve the inference speed of the model. From the results, it can be seen that deeper convolutional layers have a higher pruning ratio, this is because the features learned by deeper convolutional layers are more abstract and complex, and therefore have more redundant parameters with a higher pruning ratio.

Figure 8 shows the accuracy comparison of Model-S before and after Model-T guidance after a pruning operation with a pruning ratio of 0.7:

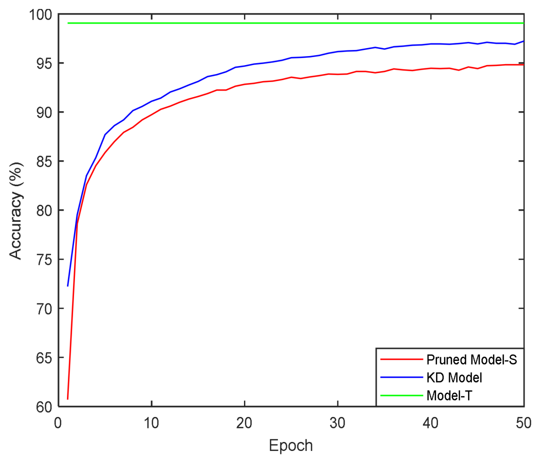

Figure 8. Comparison of the accuracy model before and after distillation.

Model-T is the pre-training model, so the model accuracy when Model-T is stabilized is selected for this comparison, which is 99.07%; the temperature coefficient in the knowledge distillation is set to 2, and the cross-loss function between Pruned Model-S and Mode-T accounts for 0.3% of the whole loss function. Model-S before and after distillation both improve the accuracy with the increase of training cycles and it can be seen that the accuracy of the model has stabilized after 30 rounds of epochs. After the 50th round of epochs, the accuracy of Pruned Model-S before distillation improves to 94.45%, and the accuracy of Pruned Model-S after distillation improves to 97.24%, with the latter improved by 2.79 percentage points over the former.

In this paper, all parameters are quantized to four bits, and since knowledge distillation does not change the model structure of the network, the number of parameters of the network does not change, so in the case of fixing the quantization unit, only the number of parameters of the model before and after the pruning needs to be considered. Table 6 shows the number of parameters of Model-S before and after pruning and the FLOPS transformation.

Table 6. The number of parameters and FLOPS transformation of Model-S before and after pruning.

Model	Quantity of Participants	FLOPs
Before pruning	9.993×10^6	3.528×10^9
After pruning	1.804×10^6	2.778×10^8

The results in the above table show that the number of senators and FLOPs of Model-S before and after pruning is drastically reduced, and the number of model senators before pruning is 9.993×10^6, i.e., about 9.99 million. After pruning, the number of modeled senators decreases to 1.804×10^6, i.e., about 1.8 million, and the decrease in the number of modeled senators is 81.9%. This shows that the pruning strategy effectively removes the redundant parameters from the model. The FLOPs of the model before pruning is 3.528×10^9, i.e., about 3.5 billion, and after pruning, the FLOPs of the model decreased to 2.778×10^8, i.e., about 280 million. The decrease in the FLOPs of the model is 92.1% which shows that the pruning strategy is able to reduce the computational complexity of the model. Overall, these data show that the pruning strategy can effectively compress the model, reduce the number of model parameters and computational complexity, and thus improve the efficiency of model training and inference.

In order to compare the recognition efficiency of the proposed algorithm with popular algorithms such as CNN, DPI, and the traditional two-stage service recognition, different numbers of pre-processed completed data were used for testing. All the samples of the traditional two-stage model have to go through the DPI detection module and the deep learning module, while the two-stage model of this paper only inputs the results that DPI fails to recognize into the deep learning model for recognition. The results are shown in Figure 9.

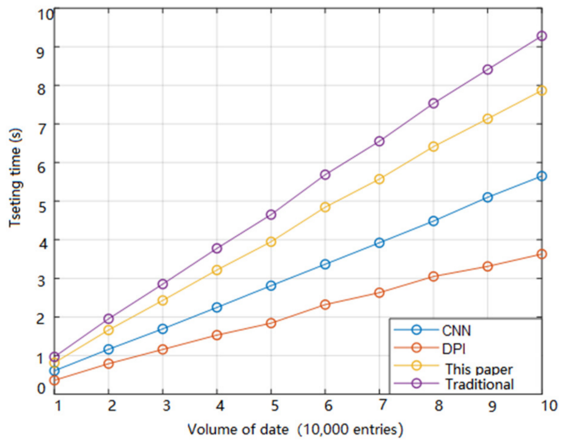

Figure 9. Comparison of the recognition efficiency of each model under different data volumes.

In this experiment, the data preprocessing time is ignored and only the algorithm prediction time is considered. The simulation results show that with the increase in data volume, the prediction time of all four models shows an upward trend and an approximately linear relationship. The two-stage model has the longest prediction time, the CNN model has the second longest prediction time, and the DPI model has the shortest prediction time. This indicates that the DPI model is more efficient, but the data that cannot be recognized by the DPI model need to be further recognized by the CNN model, which will increase the recognition time. The two-stage approach is more time-consuming compared to a single model, but with a larger amount of data, using the two-stage approach can more effectively utilize the advantages of the DPI model and identify the data that cannot be identified by the DPI through the CNN model, improving the overall recognition accuracy. Compared with the traditional two-stage algorithm, the efficiency of the proposed algorithm in this paper is improved by nearly 18 percentage points. This contributes to the idea that the proposed student network in this paper can obtain an accurate trained model from the teacher network.

5. Concluding Remarks

This paper introduces a complex power service lightweight identification method based on knowledge distillation and network pruning. It mainly includes the following aspects: First, in the system model, the overall structure and workflow of the service lightweight identification algorithm are introduced in detail, including two stages of model pruning and knowledge distillation. Secondly, knowledge distillation and network pruning are used to make the model lightweight. Specifically, the data are first preprocessed, then the teacher network and student network are constructed, and the knowledge provided by the teacher network is used to guide the student network for training, while the student network is pruned to remove useless parameters to reduce the model size and improve the inference speed. In the knowledge distillation method, the algorithm adopts an improved knowledge distillation method that takes into account the classification effect and the lightweight effect. Finally, the algorithms are fully evaluated and analyzed. The experimental results show that the proposed algorithm can significantly reduce the model size and improve the model inference speed and accuracy rate compared with the original model. In future work, we will focus on studying new identification methods of new services of new power systems based on distributed learning, such as federated learning, to further improve the number of identified service types and efficiency.

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

Funding: This research was supported by the Science and Technology Project of State Grid Corporation of China (5108-202218280A-2-415-XG).

Data Availability Statement: No additional data are available.

Conflicts of Interest: No potential competing interests were reported by the authors.

References

1. Elnawawy, M.; Sagahyroon, A.; Shanableh, T. FPGA-Based Network Traffic Classification Using Machine Learning. *IEEE Access* **2020**, *8*, 175637–175650. [CrossRef]
2. Konopa, M.; Fesl, J.; Janecek, J. Promising new techniques for computer network traffic classification: A survey. In Proceedings of the 2020 10th International Conference on Advanced Computer Information Technologies, Deggendorf, Germany, 13–15 May 2020; pp. 418–421.
3. Wang, Z.; Zhao, G.; Su, Y.; Wu, P.; Jiang, D.; Bu, X. New business identification model of power marketing inspection based on K-means clustering and text classification. In Proceedings of the 2021 6th International Symposium on Computer and Information Processing Technology (ISCIPT), Changsha, China, 11–13 June 2021; pp. 424–427.
4. Deng, Q. Transmission Section Identification in a Power System Considering Impacts of a Natural Gas Network. In Proceedings of the 2019 9th International Conference on Power and Energy Systems (ICPES), Perth, WA, Australia, 10–12 December 2019; pp. 1–5.
5. Zhong, B. Research on the identification of network traffic anomalies in the access layer of power IoT based on extreme learning machine. In Proceedings of the 2022 International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC), Kunming, China, 19–21 August 2022; pp. 374–378.
6. Ooka, R.; Miyoshi, T.; Yamazaki, T. Unit traffic classification and analysis on p2p video delivery using machine learning. *IEICE Commun. Express* **2019**, *8*, 640–645. [CrossRef]
7. Al-Obeidat, F.; El-Alfy, E.S. Hybrid multi criteria fuzzy classification of network traffic patterns, anomalies, and protocols. *Pers. Ubiquitous Comput.* **2019**, *23*, 777–791. [CrossRef]
8. Wei, W.; Sheng, Y.; Wang, J. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* **2018**, *6*, 1792–1806.
9. Zeng, Y.; Gu, H.; Wei, W. Deep-Full-Range: A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework. *IEEE Access* **2019**, *7*, 45182–45190. [CrossRef]
10. Rezaei, S.; Kroencke, B.; Liu, X. LargeScale Mobile App Identification Using Deep Learning. *IEEE Access* **2020**, *8*, 348–362. [CrossRef]
11. Shapira, T.; Shavitt, Y. FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019.
12. Chen, W.; Wilson, J.T.; Tyree, S. Compressing Neural Networks with the Hashing Trick. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
13. Song, H.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
14. Denil, M.; Shakibi, B.; Dinh, L. Predicting parameters in deep learning. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 40–54.
15. Han, S.; Pool, J.; Tran, J. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 7–31.
16. Hinton, G.; Vinyal, S.; Odean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
17. Asif, U.; Tang, J.B.; Harrer, S. Ensemble knowledge distillation for learning improved and efficient networks. In Proceedings of the European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 29 August–8 September 2020; IOS Press: Amsterdam, The Netherlands, 2020; pp. 953–960.
18. Chung, I.; Park, S.; Kim, J. Feature-map-level online adversarial knowledge distillation. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020; pp. 2006–2015.
19. Karlekar, J.; Feng, J.S.; Wong, Z.S. Deep face recognition model compression via knowledge transfer and distillation. *arXiv* **2019**, arXiv:1906.00619.
20. He, T.; Shen, C.H.; Tian, Z. Knowledge adaptation for efficient semantic segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 578–587.

21. Mirzadeh, S.I.; Farajtabar, M.; Li, A. Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 5191–5198.
22. Ye, M. A Lightweight Model of VGG-16 for Remote Sensing Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 6916–6922. [CrossRef]
23. Yuan, Z.; Lu, C. Research on Image Classification of Lightweight Convolutional Neural Network. In Proceedings of the 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Nanchang, China, 27 March 2021; pp. 498–501.
24. Fang, Y.L.; Chen, X.C.; Du, S.C. Surface defect detection method based on lightweight deep learning VGG16 network model. *Mech. Des. Res.* **2023**, *39*, 143–147.
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA; 2016; pp. 770–778.