2023 20th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) | 979-8-3503-1898-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCWAMTIP60502.2023.10387069

IMPLEMENTATION OF MODEL PRUNING METHODS FOR RESNET

WANG XUECHUN[1], ZHANG YAO[1]

[1]School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
E-MAIL: Cynthia_Hedwig@163.com, uestc_smeezy@163.com

Abstract:

The field of intelligent manufacturing is developing rapidly, with the number of network edge devices and the data generated growing rapidly. Technology of intelligent edge monitoring for surface defects has emerged, but the resources of edge devices are limited, making it difficult to deploy deep neural network models. This article proposes to use model pruning method to compress the network structure in response to this issue and implements the proposed method on ResNet network. The model reduces the number of parameters to 1/5 of the original while ensuring detection accuracy, and improves inference speed by 50%, ensuring the real-time detection of edge convolutional neural networks and improving the efficiency of varistor defect classification.

Keywords:

Deep learning; Model pruning; ResNet; Defect Detection; Edge Intelligence

1. Introduction

Recently, intelligent manufacturing has developed rapidly, with a rapid increase in network edge devices and data generation. The traditional centralized computing model not only fails to fully utilize computing resources, but also cannot fulfill the requirements of computationally intensive real-time tasks [1]. In actual industry scenarios, most business is concentrated at the edge of data source devices. The edge intelligence technology, which combines edge computing and artificial intelligence, can well meet the demand, and the edge intelligence technology came into being. Among them, the detection of surface defects in varistors also requires the use of deep learning. However, edge device resources are limited, and convolutional neural networks have many parameters that require a large amount of computing resources for inference, making deployment on edge devices with limited computing power difficult.

Deep learning technology is widely studied and applied in industry such as robotics [2], fault diagnosis, and object detection [3]. With the ability of deep learning to extract complex features from data, researchers are attempting to apply deep learning technology to the field of product surface defect detection to improve detection efficiency and thus improve product quality. For instance, Fang et al. [4] proposed a feature extraction method for deep learning combining Bayesian probability analysis for crack defect detection, achieving robust crack detection in noisy backgrounds. Cheng et al. [5] developed an automatic detection method, which is based on deep learning, to solve the problem of traditional manual interpretation of sewage pipeline detection video images.

Deep learning models need to be more streamlined and efficient to be applied to resource constrained platforms, such as embedded and mobile devices. A direct approach is designing and adopting more streamlined neural networks. SqueezeNet [6], proposed by researchers from Berkeley and Stanford University, is one of the earliest lightweight neural network models. Another method is to use deep model compression technology to reduce parameters and computational complexity of models and reduce the difficulty of deploying models on edge hardware. At present, several common model compression methods include [7]: network model pruning, low rank decomposition, knowledge extraction, parameter quantization, etc. However, low rank decomposition [8] is difficult to achieve network compression on small convolutional kernels, and knowledge extraction [9] requires a lot of experiments, with limited application scenarios.

2. Preliminary study on classical convolutional neural networks and compression methods

To complete model pruning, it is necessary to first select a suitable neural network and a convenient pruning method.

Authorized licensed use limited to: CHANGWON NATIONAL UNIVERSITY. Downloaded on June 08,2024 at 00:23:08 UTC from IEEE Xplore. Restrictions apply.

2.1. Common types of convolutional neural networks

Performance of common classification networks on ImageNet dataset is compared, which is shown in Fig.1.

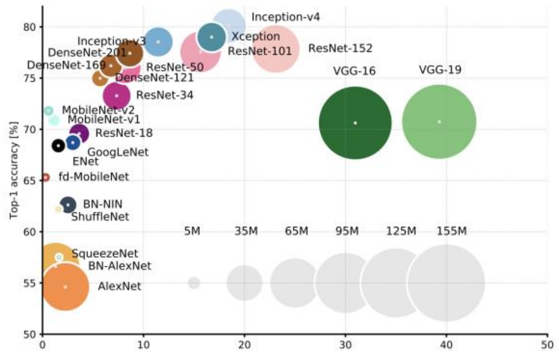

Fig.1 Comparison of common classification networks on ImageNet

The closer the dot position is to the upper left corner, the higher the recognition accuracy and the smaller the computational complexity of the corresponding model. The smaller the dot, the less the parameters of the corresponding model. According to the graph analysis, ResNet and DenseNet perform better in comparison of various indicators. ResNet, as the foundation of DenseNet and other networks, has a relatively mature and simple structure with good defect detection results. Therefore, ResNet was chosen as the research object.

2.2. ResNet:

It proposed a Residual Networks [10] structure to solve the problems of gradient explosion, gradient dissipation, and network degradation caused by the increasing depth of the model network structure. The residual network structure and common residual module forms are shown as:

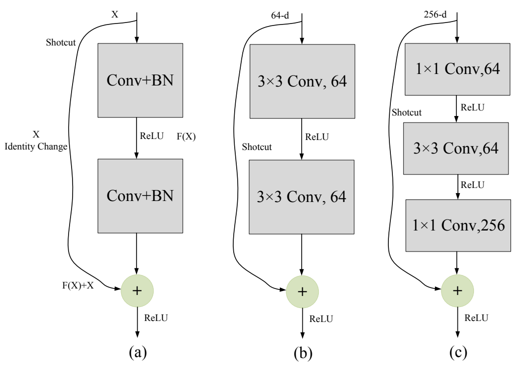

Fig.2 Residual network structure and commonly used residual structure modules

The residual network structure introduces a short circuit connection (shotcut) between the input and output of the main branch composed of a stack of convolutional operations, represented by an arc in the figure. The output is:

$$output = F(x) + x \tag{1}$$

where $F(x)$ represents output of the main branch, x is the output of shotcut. Then the output is passed through the activation function layer ReLU to obtain the final output of the residual network structure.

Fig.2 (b) is used for the construction of ResNet-18 and ResNet-34 networks, and Fig.2 (c) is used for the construction of ResNet-50/101/152 network models. Table 1 shows the accuracy and computational complexity of the ResNet series network on the ImageNet dataset.

Table 1 ResNet Networks Comparison

Model	Top-1 error rate	Top-5 error rate	Calculation quantity/GFLOPs
ResNet-34	21.34	5.56	3.60
ResNet-50	20.68	5.21	3.80
ResNet-101	19.78	4.53	7.60
ResNet-152	19.35	4.47	11.30

This article synthesizes indicators such as model accuracy, parameter quantity, and computational complexity. Considering the small dataset and limited number of classification analogies, ResNet34 is selected for training and comparison in the ResNet series of networks.

2.3. Model pruning

Model pruning is a widely used deep neural network compression method that removes some network connections that contribute too little to the inference results, thereby reducing parameters and computation while maintaining the same expression ability. From the perspective of model structure, model pruning method is mainly divided into Non-Structured Pruning and Structured Pruning [11].

Among them, unstructured pruning cannot achieve acceleration on general-purpose processors, while channel pruning, as a fine-grained form of structural pruning, combines the advantages of both the ability to use general-purpose processors for computation and the high flexibility of fine-grained pruning methods, making it widely used in model pruning.

To prune the model, the first step is to evaluate the importance of weights and neurons in the network. Liu et al. [12] proposed a model pruning method, Network Slimming, as shown in Figure 3, in the widely used BN layer γ The parameter serves as the Channel Scaling Factors for each layer, which are sparsized by the L1 norm. Then, the

Authorized licensed use limited to: CHANGWON NATIONAL UNIVERSITY. Downloaded on June 08,2024 at 00:23:08 UTC from IEEE Xplore. Restrictions apply.

channels corresponding to the set threshold (orange part in Figure 3) are pruned to obtain the pruned compact network. This method is widely used because it directly utilizes the network structure itself, which is simple and effective to implement.

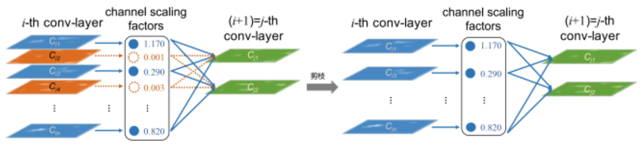

Fig.3 Network Slimming Method

Network Slimming first introduce a scaling factor γ to each channel to sparse the model. Then it is multiplied by channel output and perform sparse regularization:

$$L=\sum_{(x,y)} l(f(x,W),y)+\lambda\sum_{\gamma\in\Gamma} g(\gamma) \tag{2}$$

where (x,y) is model input, W is the weight parameter of the network model, $\sum l(f(x,W),y)$ is the loss function during normal training of network model, $\lambda\sum g(\gamma)$ is the sparsity induced penalty term, and the sparsity penalty function $g(s)$ is selected as $g(s)=\|s\|$, that is, using the $L1$ norm to achieve sparsity.

Batch Normalization (BN) layer has been widely applied in convolutional neural networks. Network simplification in BN selects γ as scaling factor for sparsity penalty:

$$Z=\frac{Z_{\text{in}}-\mu_B}{\sqrt{\delta_B^2+\varepsilon}} \tag{3}$$

$$Z_{out}=\gamma Z+\beta \tag{4}$$

where Z_{in} represents the input, Z_{out} represents the output of BN, B represents the small batch statistical data used, μ_B is the average deviation values, δ_B is standard deviation values on the small batch data, γ is a scaling coefficient used to expand and reduce the data, β is the offset coefficient, γ and β are parameters that need to be learned during training. Network Slimming is the reuse of the scaling coefficient γ in BN layer, which is used as a sparsity penalty representing the importance of the corresponding channel.

After several rounds of sparse training with joint weight parameters and scaling factor γ, the scaling factor γ is forced to be sparse. Then, the absolute values of the scaling factors are sorted to obtain a pruning threshold based on the set pruning rate. The channels corresponding to factors lower than the set threshold is pruned. Accuracy of the model after pruning may decrease due to changes in the model structure before and after pruning. So it is necessary to fine-tune the network and retrain several rounds to maintain model accuracy. After that, the model can quickly recover to its previous detection accuracy, and in some cases, it may even have higher accuracy than the original model. The entire pruning method is shown as:

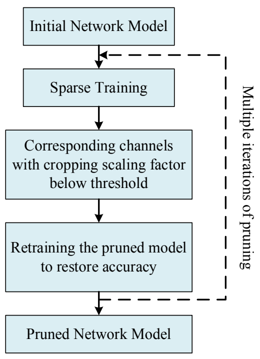
Initial Network Model
Sparse Training
Corresponding channels with cropping scaling factor below threshold
Retraining the pruned model to restore accuracy
Pruned Network Model
Multiple iterations of pruning

Fig.4 Pruning algorithm process

3. Experimentation

For classification tasks, if the images in the dataset are divided and summarized by predicted and actual categories, T (True) indicates correct prediction, and F (False) indicates incorrect prediction, P (Positive) indicates that the prediction is true, N (negative) indicates that the prediction is false, and the samples in the dataset can be classified into the following four categories. FP represents the predicted type is true, but the actual type is false. FN represents the predicted type is false, but the actual one is true. TP represents they are both true. And TN represents they are both false,

Based on the above basic indicators, some commonly used indicators are applied to evaluate model, including accuracy (Acc), precision (Pre), recall (Rec), F1 score (F1), etc.

$$accuracy=\frac{TP+TN}{TP+FP+TN+FN} \tag{5}$$

$$precision=\frac{TP}{TP+FP} \tag{6}$$

$$recall=\frac{TP}{TP+FN} \tag{7}$$

$$F1Score=2\times\frac{precision\times recall}{precision+recall} \tag{8}$$

Authorized licensed use limited to: CHANGWON NATIONAL UNIVERSITY. Downloaded on June 08,2024 at 00:23:08 UTC from IEEE Xplore. Restrictions apply.

3.1. Test before pruning

Accuracy of ResNet-34 on test set after training can reach to 96.29%. The specific results of accuracy in different classifications on the varistor defect test set are shown in Table 2.

By calculating the indicators mentioned, the experimental results of ResNet-34 are shown in Table 3.

According to the classification test indicators, ResNet-34 achieved an F1 score of 100% in certain classification tests, indicating the absence of false positives or missed detections. The worst detection results are surface stains and defects. This may be because the characteristics of surface stains and surface damage defects are similar, and the number of samples for surface damage defect types is too small, making it difficult for the model to effectively distinguish between the two.

Table 2 Accuracy of different classifications

Varistor Defect	Accuracy
Surface Damage	100%
Surface Stains	97.0%
Side Depression	97.9%
Surface adhesion	91.1%
No defects	100%
Test Set Accuracy	96.29%

Table 3 ResNet-34 classification test indicators

Defect Type	Accuracy	Recall	Precision	F1-Score
Surface Damage	100.0%	100.0%	95.2%	0.976
Surface Stains	97.0%	97.0%	99.0%	0.980
Side Depression	97.9%	97.9%	81.0%	0.887
Surface adhesion	91.1%	91.1%	99.1%	0.950
No defects	100.0%	100.0%	100.0%	1.000

For multi classification tasks, macro average, micro average, and weighted average can be used to evaluate the comprehensive prediction performance by combining multiple classification scenarios. Weighted averaging considers the issue of imbalanced sample sizes in different categories. Therefore, weighted average values are used to evaluate the model's overall predictive performance, which are shown in Table 4.

Table 4 Comprehensive classification indicators

Index	Accuracy
Weighted-average Recall	96.7%
Weighted-average Precision	96.5%
Weighted-average F1-Score	0.965

To measure the size and inference speed of the model, data samples were selected randomly from the test set. The model was tested for different indexes. Results are shown in Table 5. It requires 14.71G of computing power, which is so huge.

Table 5 Model inference parameters

Index		Value
Accuracy		96.29%
Parameter Quantity		21.29M
Computation (FLOPs)		14.71G
Inference Memory Usage		207.2MB
Average inference time	1	9.6
for different BatchSizes	4	29.2
(ms)	8	53.7

3.2. Test results after pruning

Model ResNet-34 was pruned at different pruning rates (PR). The accuracy (Acc), parameter quantity (PQ), computational complexity (CC), inference memory usage (IMU) and inference speed of the model test set are shown in Table 6.

Table 6 Results of different pruning rates

PR	Acc	PQ	CC	IMU	Average inference time for different BatchSizes (ms)		
					1	4	8
0.0	96.3%	21.3M	14.71G	207.2M	9.6	29	54
0.3	50.5%	14.7M	10.43G	168.3M	6.9	22	45
0.5	22.8%	4.77M	1753M	116.2M	7.4	18	28
0.7	11.8%	0.68M	690M	37.5M	3.2	6.4	10
0.9	30.7%	0.45M	374M	32.6M	2.6	6.1	11

According to Table 6, when the pruning rate is 0.3, PQ and CC after pruning are reduced to 7/10 of those before pruning. When the pruning rate is 0.5, they are about 1/5 and 1/4 of those before pruning, respectively. When the pruning rate is 0.7, PQ and CC are about 1/30 and 1/24 of those before pruning, respectively. It can be seen that after pruning, PQ and CC of the model are significantly reduced compared to before pruning, and the higher the pruning rate, the more significant the decrease.

From the perspective of average inference time, when BatchSize is 1 during inference, the trend of inference time decreasing with the increase of pruning rate is not significant. When BatchSize is 4, 8, and 16, the continuous increase in pruning rate further reduces the inference time required for the model. When BatchSize is 4 and pruning rate is 0.3, the average inference time after pruning decreases to 2/3 of that before pruning. The pruning rate is increased to 0.5, and the average inference time is reduced to half of before pruning. When pruning rate is 0.7, average inference time decreases to 1/5 of before pruning. The inference speed of the model has been improved after pruning, and the time required for model inference has significantly decreased. As the pruning rate increases, the inference time required by the model further decreases when BatchSize is greater than 1, but the decreasing trend is not as obvious as the parameter and

Authorized licensed use limited to: CHANGWON NATIONAL UNIVERSITY. Downloaded on June 08,2024 at 00:23:08 UTC from IEEE Xplore. Restrictions apply.

computational complexity.

After pruning, the model's accuracy in the test set is significantly reduced, and fine-tuning of the pruned model is necessary to restore accuracy.

3.3. Test results after fine tuning

After retraining and fine-tuning pruned model, the detection accuracy is shown in Table 7.

Table 7 ResNet-34 classification test indicators after pruning

Pruning rate	Accuracy before pruning (%)	Accuracy after pruning (%)	Accuracy after fine tuning (%)
0.3	96.3	50.5	96.1
0.5	96.3	22.8	95.6
0.7	96.3	11.8	85.4
0.9	96.3	30.7	82.7

According to Table 7, accuracy of the model decreases after pruning, but after retraining and fine-tuning, the accuracy significantly improves. At pruning rates of 0.3 and 0.5, after fine-tuning the accuracy, the accuracy of the test set can basically recover to the level before trimming. When the pruning rate reaches 0.7 or higher, after retraining and fine-tuning, accuracy of the model test set will still rebound, but cannot be restored to the accuracy before pruning, and the accuracy is less than 90%. When the pruning rate is 0.9, the accuracy is even lower than 85%. The reason for this situation may be due to excessive pruning force damaging the structure of the model, resulting in the inability of the trimmed model to extract effective feature information.

Therefore, by balancing multiple model performance evaluation indicators such as detection accuracy, number of model parameters, and computational complexity, it was ultimately found that when ResNet-34 was pruned with a channel pruning rate of 0.5, it has the best performance. After pruning, the number of parameters and computational complexity are about 1/5 and 1/4 of those before pruning, respectively, and the average inference time decreases to 1/2 of that before pruning.

4. Conclusions

This article proposes the use of model pruning methods to compress network structures. After comparing accuracy of ResNet before pruning, after pruning, and after fine-tuning parameters, it was found that the model can reduce the number of parameters to 1/5 of the original one while ensuring detection accuracy and improve inference speed by 50%. This ensures real-time detection of edge convolutional neural networks and improves the efficiency of varistor defect classification. The experiment further verified the rationality of this method.

The plan also has shortcomings. For example, further research can be conducted on the effectiveness of model pruning methods applied to DenseNet, as well as the effectiveness of applying parameter quantification to compress models.

References

[1] Chen Y. Integrated and intelligent manufacturing: perspectives and enablers[J]. Engineering, 2017, 3(5): 588-595.

[2] Chao Y, Chen X, Xiao N. Deep learning-based grasp-detection method for a five-fingered industrial robot hand[J]. IET Computer Vision, 2019, 13(1): 61-70.

[3] Liu N, Xu Y, Tian Y, et al. Background classification method based on deep learning for intelligent automotive radar target detection[J]. Future generation computer systems, 2019, 94: 524-535.

[4] Fang F, Li L, Gu Y, et al. A novel hybrid approach for crack detection[J]. Pattern Recognition, 2020, 107: 107474.

[5] Cheng J C P, Wang M. Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques[J]. Automation in Construction, 2018, 95: 155-171.

[6] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

[7] Cheng Y, Wang D, Zhou P, et al. A survey of model compression and acceleration for deep neural networks[J]. arXiv preprint arXiv:1710.09282, 2017.

[8] Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions[J]. arXiv preprint arXiv:1405.3866, 2014.

[9] Buciluǎ C, Caruana R, Niculescu-Mizil A. Model compression[C]. 12th ACM SIGKDD international conference on Knowledge discovery and data mining. 2006: 535-541.

[10] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[11] Anwar S, Hwang K, Sung W. Structured pruning of deep convolutional neural networks[J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2017, 13(3): 1-18.

[12] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]. IEEE international conference on computer vision. 2017: 2736-2744.

Authorized licensed use limited to: CHANGWON NATIONAL UNIVERSITY. Downloaded on June 08,2024 at 00:23:08 UTC from IEEE Xplore. Restrictions apply.