EasyChair Preprint

№ 7221

Rapid and High-Purity Seed Grading Based on Pruned Deep Convolutional Neural Network

Huanyu Li, Cuicao Zhang, Chunlei Li, Zhoufeng Liu, Yan Dong and Shuili Tang

December 16, 2021

Rapid and high-purity seed grading based on pruned deep convolutional neural network

Huanyu Li[1(✉)], Cuicao Zhang[1], Chunlei Li[1(✉)], Zhoufeng Liu[1], Yan Dong[1], and Shuili Tang[2]

[1] Zhongyuan University of Technology, ZhengZhou, China, 450007
[2] HI-TECH HEAVY INDUSTRY CO., LTD, Zhengzhou, China
lihuanyu,lichuanlei1979@zut.edu.cn

Abstract. The crop seed grading method based on deep learning has achieved ideal recognition results. However, an effective deep neural network model for seed grading usually needs a relatively high computational complexity, memory space, or inference time, which critically hampers the utilization of complex CNNs on devices with limited computational resources. For this reason, a method of combining layer pruning and filter pruning is proposed to realize fast and high-purity seed grading. First, we propose an effective approach based on feature representation to eliminate redundant convolutional layers, which greatly reduces the model's consumption of device storage resources. Then, the filter-level pruning based on the Taylor expansion criterion is introduced to further eliminate the redundant information existing in the convolutional layer. Finally, an effective and practical knowledge distillation technology (MEAL V2) is adopted to transfer knowledge of well-performing models, to compensate for the information loss caused by the pruning of the network. Experiments on red kidney bean datasets demonstrate that the method is effective and feasible. We proposed the Vgg_Beannet, which can achieve 4× inference acceleration while the accuracy is only reduced by 0.13% when the filter is pruned by 90%. Moreover, we also compared some handcrafted lightweight architectures such as MobileNetv2, MixNet, etc. The results show that the pruned network outperforms the above network in inference time (2.07ms vs. 7.83ms, 22.23 ms) and accuracy (96.33% vs. 95.94%, 94.89%).

Keywords: Seed grading · Deep learning · Neural network pruning · Knowledge distillation.

1 Introduction

In the processing industry, the purity of seeds is an important evaluation criterion of quality rating. The seed grading can improve the seed quality, save the sowing quantity and cereals, advantageous to achieving sowing mechanization and precision sowing, and it can bring significant social benefits. Traditional

seed defect detection methods generally rely on manual detection, which is inefficient and subjective. Therefore, an objective and automated seed grading method is required.

To solve this problem, researchers have applied machine vision technology to detect seed quality [1–3]. Features, such as histogram of oriented gradient (HOG), color , texture, Gabor etc, can be extracted from images of seeds, and then, the various effective classifiers are employed to identify the defects of the seed, such as support vector machine (SVM), decision tree (DT) etc. However, because of the diversity and fine-grained recognition of defective seeds, these methods based on manual feature extraction are difficult to distinguish fine-grained difference, resulting in low classification accuracy and lack of self-adaptivity.

Recently, some researchers also adopted deep learning technology in crop identification tasks and achieved good performance [4–6]. The deep network model represented by a convolutional neural network (CNN) significantly improves the accuracy of traditional detection and recognition problem by automatically learning a hierarchical feature representation from raw data. Heo et al. [4] used CNN to filter weed seeds from high-quality seeds, Uzal et al. [5] adopted CNN to estimate the number of soybean seeds. However, the accuracy of the above crop classification methods based on deep learning depends on the model depth. However, with the rise of network depth and width, the time complexity and spatial complexity of the depth model will increase, which will suffer from slow inference speed, especially the seed sorting system with high throughput. Moreover, the massive researches indicate that the existing DNN models have numerous parameter redundancy, which consumes massive computing and storage resources.

Due to the limited computing resource platform such as FPGA, GPU, MCU, etc. Deep model compression provides an effective solution for reducing the model size and lowering the computation overheads, such as network structure search (NAS) [7], weight quantization [8], knowledge distillation [9, 19], and network pruning [11, 12]. NAS requires massive computing resources and brings a set of new hyperparameter problems. Quantization reduces the bit-width of parameters, thus decreases memory footprint, but requires specialized hardware instructions to achieve latency reduction [13]. Network pruning has the advantages of simple operation, efficient implementation, can reduce network complexity and solve over-fitting problems, and has shown broad prospects in various emerging applications.

Neural network pruning can realize the pruning of weights, filters, and convolutional layers. The fine-grained pruning at the weight level is flexible, but it needs specialized software or hardware to achieve the practical effect. The coarse-grained pruning based on the filter not only owns high flexibility but also does not need the corresponding cooperation of software and hardware, however, it has certain limitations in reducing latency. The pruning model obtained by layer pruning owns less runtime memory usage and inference time because fewer

layers mean fewer data moving in memory, thereby improving computational efficiency.

In this study, a mixed pruning strategy, which takes both layer pruning and channel pruning into consideration, is proposed to achieve model compression and improve the inference speed of the algorithm. First, we designed a set of linear classifiers to explore the roles and dynamics of intermediate layers and combined with feature visualization analysis to remove the redundant convolution layer. Then, we adopted the criterion based on Taylor expansion to approximate the change in the loss function if removing the least important parameters and then directly prunes those corresponding to the almost flat gradient of the loss function. Finally, a multi-teacher integrated knowledge distillation technology [10] is introduced to transfer knowledge to the pruning network to compensate for the accuracy loss caused by pruning. Overall, our contributions are three-fold as follows:

1) We proposed a mixed pruning strategy based on feature representation and Taylor expansion to achieve fast and high-purity seed grading.

2) A simple and effective multi-teacher integrated knowledge distillation method (i.e., meal V2) is introduced to transfer the knowledge of CNN with high accuracy to the pruned network to recover its predictive accuracy.

3) Experiments are conducted on our constructed red kidney bean datasets, and the results show this method greatly improves the inference speed, memory consumption, and computation cost with almost no loss of accuracy.

2 Proposed Method

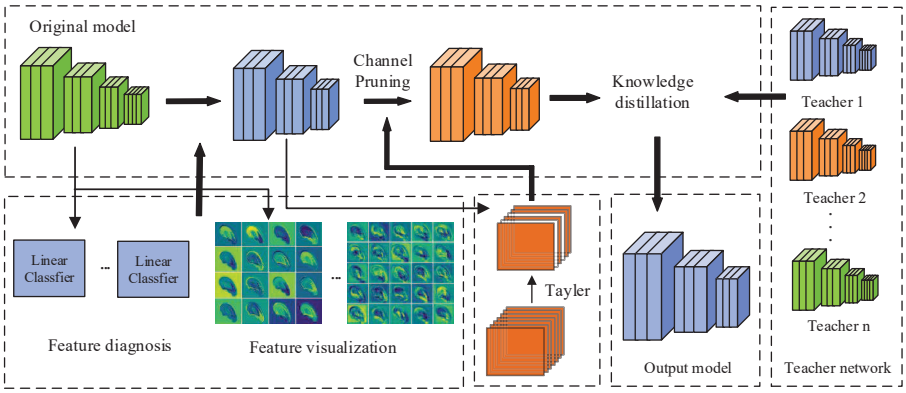

Fig. 1. The structure flow chart of the proposed method.

Automatic seed grading based on deep learning provides an effective solution for improving seed quality. However, high computational complexity, memory space and inference time of the deep learning model limit its deployment on resource-constrained edge devices. We proposed a mixed pruning strategy based on feature representation and Taylor expansion to achieve fast and high-purity

seed grading, as shown in Fig. 1. In the stage of layer pruning, we designed a set of linear classifiers for feature diagnosis and combined feature visualization technology to remove redundant convolutional layers. In the filter pruning stage, a highly efficient method based on the Taylor expansion criterion is employed, which adopts the first derivative as the criterion to measure the importance of filters and eliminate redundant filters. Finally, multi-teacher integrated knowledge distillation technology is utilized to transfer knowledge to the pruning network to compensate for the accuracy loss caused by pruning. And the proposed method is specifically described as follows.

2.1 Network Structure and Feature Diagnosis

A.Network structure. Vggnet [14] is a very deep convolutional network and has good generalization ability to a wide range of complex pattern recognition tasks. Moreover, compared with some complex networks, it has the advantages of simple structure, fast inference speed, and easy deployment, so it is still heavily used for real-world applications in both academia and industry. In this paper, an improved VggBN-16 [15] network is selected as the feature extraction network, its structure is shown in Fig. 2.

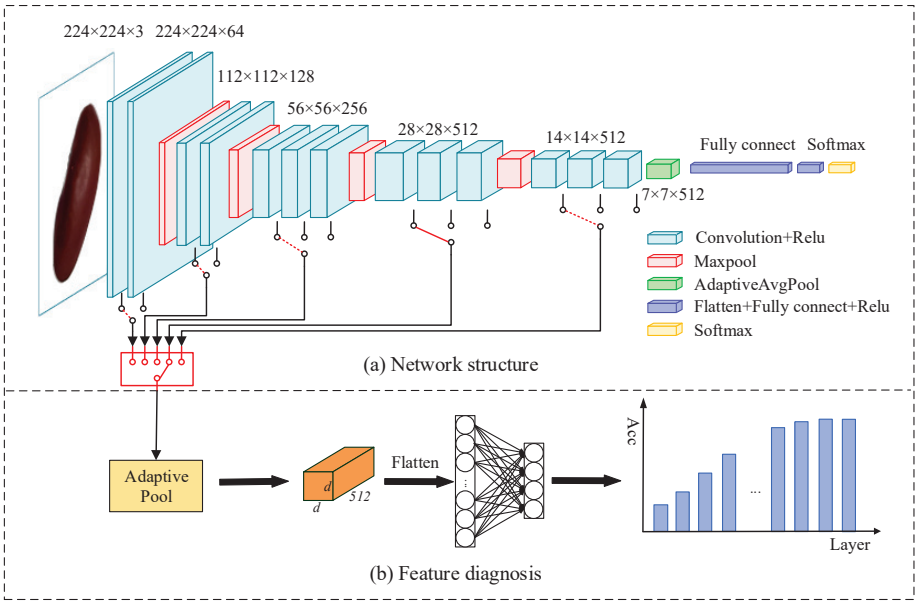

Fig. 2. The structure flow chart of the proposed method.

B.Feature diagnosis. CNN (including VggBN-16) containing both feature extractor and classifier, learns reasonable feature representations specific to the current task with corresponding training data. While the features from the last convolutional layer of a CNN tend to provide the best discriminative power,

features from intermediate layers also contain important information related to the tasks and can be utilized to analyze the behaviors of the corresponding CNN. At the same time, with the increase of network depth, the computational cost of this method will increase dramatically and the test accuracy will be affected. Therefore, we trained a set of linear classifiers on features extracted at different layers within the network and visualized the outputs to explore the roles and dynamics of intermediate layers, then determined the optimal number of network layers.

As shown in Fig. 2, since each layer has a different output feature shape, we adopted adaptive average pool to simplify the method and unified the embedding length so that each layer can produce roughly the same output size, the pooling is done as follows:

$$d_i = round\left(\sqrt{\frac{N}{n_i}}\right) \tag{1}$$

$$E_i = AdaptiveAvgPool\left(M_i, d_i\right) \tag{2}$$

where N is the embedding length, i $(1 \leq i \leq L)$ is the convolution layer index, n_i is the number of filters in the i-th convolutional layer, M_i is layer i's $(1 \leq i \leq L)$ output map. AdaptiveAvgpool reduces M_i to embedding $E_i \in R^{n \times d \times d}$. Then, the output E_i of each layer is flattened to z_i as the input of the fully connected network. Finally, we train a set of linear classifiers $F_i(z_i)$ to predict the correct class y using only the specific intermediate result:

$$z_i = flatten\left(E_i\right) \tag{3}$$

$$y_i = F_i\left(z_i\right) = softmax\left(w_i z_i + b_i\right) \tag{4}$$

where w_i and b_i respectively represent the weight and bias of the i-th linear classifier. During training, we will freeze the parameters of the original network model, finetune the auxiliary classifier through backpropagation, and finally determine the feature extraction ability of the convolutional layer.

Combining the feature visualization technology, we found that the shallow network has good feature extraction capabilities. Through the experimental results, the layer with less contribution in the deep model is eliminated, and the compression of the convolutional layer of the model is realized, which speeds up the reasoning of the model.

2.2 Taylor Expansion Criterion-Based Channel Pruning

Through feature visualization, we also found that some feature maps are similar to each other, which proves that the neural network structure still has some redundant information, so we can prune the convolution network model to remove redundant information, thus enhancing the inference speed. And the Tayler expansion based network pruning methods [15] have been widely used to condense the structure of the CNN and then make a balance between the generalization and compact network because it does not lack basic theoretical guidance or bring a new set of hyperparameter problems. It regards channel pruning as an

optimization problem, i.e., minimizing the difference between the cost functions before and after pruning.

Raw data were processed to generate training samples

$$\mathcal{W} = \left\{ \left(w_1^1, b_1^1\right), \left(w_1^2, b_1^2\right), \cdots, \left(w_L^{C_1}, b_L^{C_l}\right) \right\}$$

where $w_i^l (i = 1, 2, \cdots, L)$ is the weight parameter, C_l is the number of channels. $\mathcal{L}(\mathcal{D} \mid \mathcal{W})$ represents the cost function, which is the optimization objective of this study. In the process of channel pruning, a subset $\mathcal{W}'$ is refined from original parameters $\mathcal{W}$ by using the following combinatorial optimization:

$$\min_{\mathcal{W}'} \left| \mathcal{L}\left(\mathcal{D} \mid \mathcal{W}'\right) - \mathcal{L}(\mathcal{D} \mid \mathcal{W}) \right| \quad s.t. \left\| \mathcal{W}' \right\|_0 \leq B \tag{5}$$

the norm l_0 in $\left\| \mathcal{W}' \right\|_0$ limits the number of nonzero parameters B. If $\mathcal{L}\left(\mathcal{D} \mid \mathcal{W}'\right) \approx \mathcal{L}(\mathcal{D} \mid \mathcal{W})$, it is easy to reach the global minimum of Eq.(5). After pruning a specific parameter, the change in the loss function is approximated by

$$\left| \Delta \mathcal{L}\left(h_i\right) \right| = \left| \mathcal{L}\left(\mathcal{D}, h_i = 0\right) - \mathcal{L}\left(\mathcal{D}, h_i\right) \right| \tag{6}$$

where $\mathcal{L}\left(\mathcal{D}, h_i = 0\right)$ is the cost after pruning, $\mathcal{L}\left(\mathcal{D}, h_i\right)$ is the cost without pruning, h_i is the eigenvalue of parameter i output. Using a first-order Taylor polynomial $\mathcal{L}\left(\mathcal{D}, h_i = 0\right)$ is approximated by [15]

$$\mathcal{L}\left(\mathcal{D}, h_i = 0\right) = \mathcal{L}\left(\mathcal{D}, h_i\right) - \frac{\delta \mathcal{L}}{\delta h_i} h_i + R_1\left(h_i = 0\right) \tag{7}$$

Where $R_1\left(h_i = 0\right)$ is expressed as:

$$R_1\left(h_i = 0\right) = \frac{\delta^2 \mathcal{L}}{\delta\left(h_i^2 = \xi\right)} \frac{h_i^2}{2} \tag{8}$$

where ξ is a value in the range of 0 and h_i. If the influence caused by removing the high-order term can be ignored, substituting Eq.(7) into Eq.(6).

$$\theta_{\mathrm{TE}}\left(h_i\right) = \left| \Delta \mathcal{L}\left(h_i\right) \right| = \left| \mathcal{L}\left(\mathcal{D}, h_i\right) - \frac{\delta \mathcal{L}}{\delta h_i} h_i - \mathcal{L}\left(\mathcal{D}, h_i\right) \right| = \left| \frac{\delta \mathcal{L}}{\delta h_i} h_{\mathrm{i}} \right| \tag{9}$$

Based on this definition, the parameters having an almost flat gradient of the cost should be pruned, and then θ_{TE} is computed for a feature map by [15].

2.3 Knowledge Distillation of Multi-model Ensemble

After the mixed pruning of channel and layer, we have obtained networks with a more compact architecture. However, in the process of network pruning, some useful information may be lost, which leads to the performance degradation of the model. To compensate for the performance loss, we introduced a simple and effective knowledge distillation technology (Meal v2) to transfer knowledge from

the original model and some CNNs with high accuracy to the pruned model for boosting its performance.

The method adopted the similarity loss and discriminator only on the final outputs and used the average of SoftMax probabilities from all teacher ensembles as the stronger supervision for distillation [10]. The realization process is shown in Fig. 3, which mainly consists of three parts: teacher ensemble, KL divergence loss, and the discriminator.

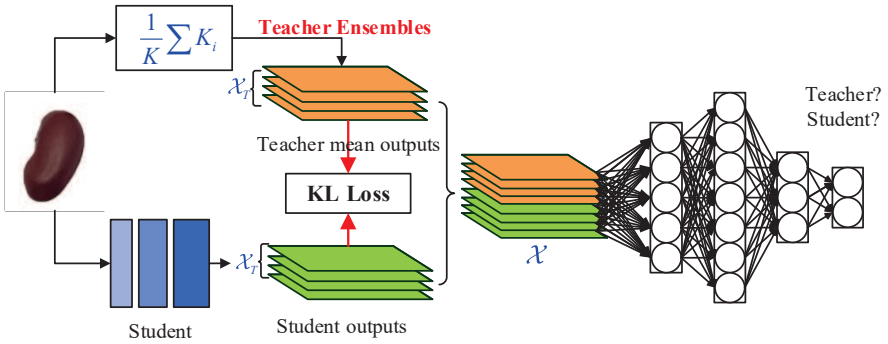

Fig. 3. The realization process of knowledge distillation based on Meal v2 [10].

A. Teachers ensemble. We chosen some models that perform well in the seed dataset as the teacher network and adopted the average of softmax probabilities of these pre-trained teachers as an ensemble. Assuming $\mathcal{T}_\theta$ as a teacher network,the output ensemble probability $\hat{\mathrm{p}}_{\mathrm{e}}^{\mathcal{T}_\theta}$ can be described as:

$$\hat{\mathrm{p}}_{\mathrm{e}}^{\mathcal{T}_\theta}(X) = \frac{1}{k}\sum_{\mathrm{t}=1}^{k} \mathbf{p}_{\mathrm{t}}^{\mathcal{T}_\theta}(X) \tag{10}$$

where $\mathbf{p}_t^{\mathcal{T}_\theta}$ represents the t-th teacher's softmax prediction. X is the input image and k is the number of total teachers [10].

B. KL divergence. It is used to measure the similarity of two probability distributions. We trained the student network $\mathcal{S}_\theta$ by minimizing between its output $\hat{p}_e^{\mathcal{S}_\theta}(x_i)$ and the ensembled soft labels $\hat{p}_e^{\mathcal{T}_\theta}(x_i)$ generated by the teacher ensemble. In practice, we can simply minimize the equivalent cross-entropy loss as follows [10]:

$$\mathcal{L}_{\mathrm{CE}}(\mathcal{S}_\theta) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_{\mathrm{t}}^{\mathcal{T}_\theta}(x_i) \log \mathbf{p}^{\mathcal{S}_\theta}(x_i) \tag{11}$$

where n is the number of samples.

D. Discriminator. The discriminator is a binary classifier, which is used to determine whether the input features come from the teacher set or the student network. It consists of a sigmoid function following the binary cross-entropy loss [10]. The loss can be formulated as:

$$\mathcal{L}_{\mathcal{D}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{y}_i \cdot \log \hat{\mathbf{p}}_i^{\mathcal{D}} + (1-\mathbf{y}_i)\cdot \log\left(1-\hat{\mathbf{p}}_i^{\mathcal{D}}\right)\right] \tag{12}$$

$$\mathbf{p}^{\mathcal{D}}(x;\theta) = \sigma\left(f_\theta\left(\{x_{\mathcal{T}}, x_{\mathcal{S}}\}\right)\right) \tag{13}$$

where $y_i \in \{0,1\}$ is the binary label for the input features x_i, and $\hat{\mathbf{p}}_i^{\mathcal{D}}$ is the corresponding probability vector. $\mathbf{p}^{\mathcal{D}}(x;\theta)$ is a *sigmoid* function is used to model the individual teacher or student probability. f_θ is a three-FC-layer subnetwork and θ is its parameter, $\sigma(*)$ is the logistic function. The final loss function is:

$$\mathcal{L}_{LOSS} = \mathcal{L}_{\mathcal{D}} + \mathcal{L}_{CE} \tag{14}$$

Finally, the loss is minimized by backpropagation.

3 Experiments

In this section, we demonstrate our experiments as follows. Part 1 introduces the dataset used in the experiments. Part 2 indicates training details and the performance metrics we used such as Acc, Flops, F1_scores, interference time, and parameters. At last, Part 3 presents the experimental results and discussions.

3.1 Red Kidney Bean Dataset

In the training stage, as the deep convolutional network described, a large amount of data is required. However, there is currently no suitable seed database, so the seed images used in our method were acquired by the sorting machine in the actual seed harvest process using a highspeed camera in a real environment. According to the requirements for the quality grading of red kidney beans by enterprises, the sample images of red kidney beans were divided into four categories: plump beans (1661), peeled beans(509), dried beans(1173), and broken beans(488), which were randomly assigned to the training set, verification set and test set at a ratio of 3:1:1, typical images are shown in Fig. 4.

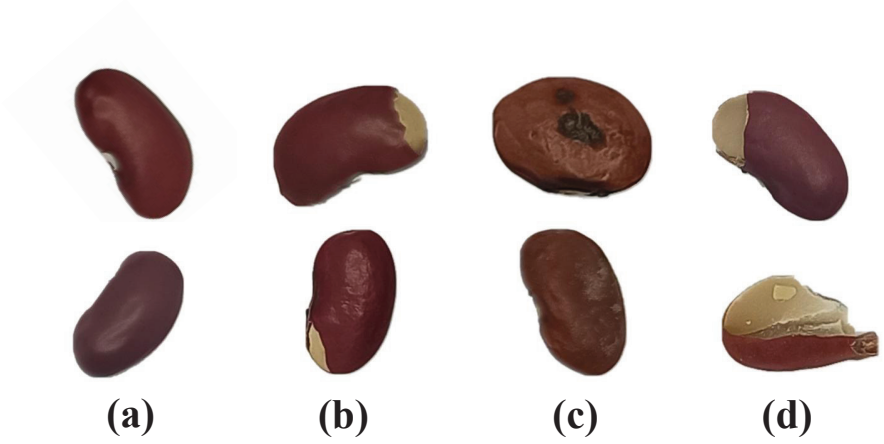

Fig. 4. Red kidney bean dataset. (a) plump beans (b) peeled beans (c) dried beans (d) broken beans.

3.2 Training Details and Evaluation Metric

All experiments were performed on a 64-bit Linux-based operation system, Ubuntu 18.04. The software is mainly based on the deep learning architecture of Pytorch and python development environment Spyder. The hardware is based on an NVIDIA 1080 TI GPU, with CUDA10.1 accelerating calculation.

The size of the input image is 224×224, and use small batch stochastic gradient descent (SGD) to train the network, the initial learning rate is 0.001, the Batch_size is 16, the epoch is 100. In the number of steps at 1/2 and 3/4, the learning rate is adjusted to 1/10 of the original, the momentum parameter is set to 0.9, and the weight decay parameter is set to 0.0001. Besides, every iteration, each of the input batch-size images through some transformations, such as Flip horizontally or vertically etc.

The number of parameters and required Float Points Operations (denoted as FLOPs) are employed to evaluate the model size and computational requirement, which are widely used protocols. To evaluate the seed grading task performance, we also provide the accuracy, F1_score models, and inference time on Quadro m5000 GPU for an image.

Table 1. Performance of some popular CNN in red kidney bean test set.

Model	Params(M)	FLOPs	Time(ms)	Acc(%)	F1_score(%)
Alexnet [16]	57.02	711.46M	2.31	88.60	88.22
Resnet50 [17]	23.52	4.12G	10.44	95.54	95.54
DenseNet121 [18]	6.96	2.88G	22.71	95.67	95.67
Googlenet [19]	5.60	1.51G	10.89	96.85	96.87
VggBN-16	14.82	15.41G	9.53	96.59	96.58

3.3 Experimental Results and Analysis

Selection of pruning model. In this study, we first compared the performance of some popular CNNs in the red kidney bean test set. It mainly includes VggBN-16, Alexnet [16], ResNet50 [17], DenseNet121 [18], and GoogleNet [19]. Experimental results of Table 1 show that VggBN-16 and GoogleNet can achieve higher accuracy (96.59%, 96.85%) and f1-score (96.69% and 96.87%). In addition, although VggBN-16 has more parameters and computation, it has a faster inference speed than other networks (excluding Alexnet). The main reasons for this problem are as follows: 1) The complicated multi-branch designs make the model difficult to implement and customize, slow down the inference and reduce the memory utilization. 2) Some components (e.g., depthwise Conv in Xception and MobileNets and channel shuffle in ShuffleNets) increase the memory access cost and lack supports of various devices. Therefore, we will further compress the VggBN-16 to make it easy to be deployed on edge devices to achieve fast and high-purity seed grading.

Feature map visualization and feature diagnosis. CNN is an end-to-end architecture. The recognition result can be automatically obtained by feeding only the pictures to be recognized to the network. The intermediate process is usually a black box and not interpretable. We used visualization technology to extract the output feature maps of each layer in the network. To facilitate observation, we selected seeds with obvious damage. We showed the output feature map of the active layer from layer 3 to layer 13 in Fig. 5. It could be seen that the layer retained the original image color, shape, and texture feature information. In addition, we can observe that the features extracted by the network become more abstract as the depth of the layer increases.

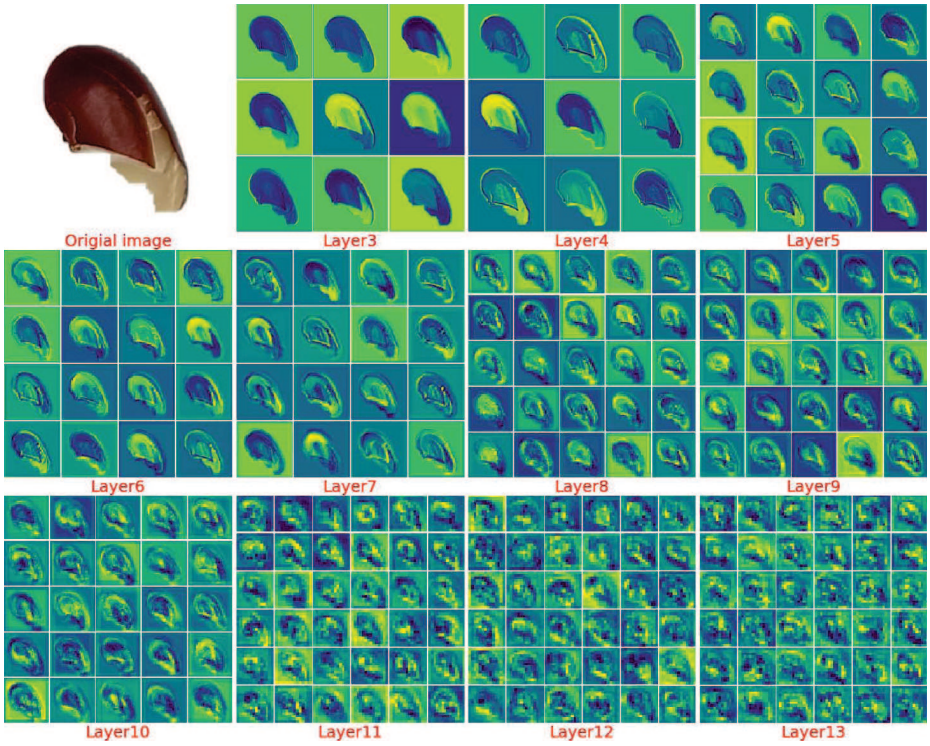

Fig. 5. The visualization results of the feature map in a pseudocolor image.

Meantime, we trained a set of linear classifiers on features extracted at different layers within the network to explore the roles and dynamics of intermediate layers, then determine the optimal number of network layers. As shown in Fig. 6, The recognition accuracy of CNN is gradually improved with the increase of network depth, and after the tenth layer, its performance is not significantly improved. Combined with the above analysis results of feature visualization, we think that the shape, texture, and color feature information extracted from the first ten layers of the network can achieve better results, and if the network is too deep, the computational cost of the network will increase dramatically and the test accuracy will be affected. Therefore, we try to change the VggBN-16 net-

work structure to further optimize the performance of seed sorting. Ultimately, we only keep the convolution structure of the first ten layers of VggBN-16 and name it Vgg_BeanNet. The F1_score after finetuning is 96.47%, as shown in Table 2.

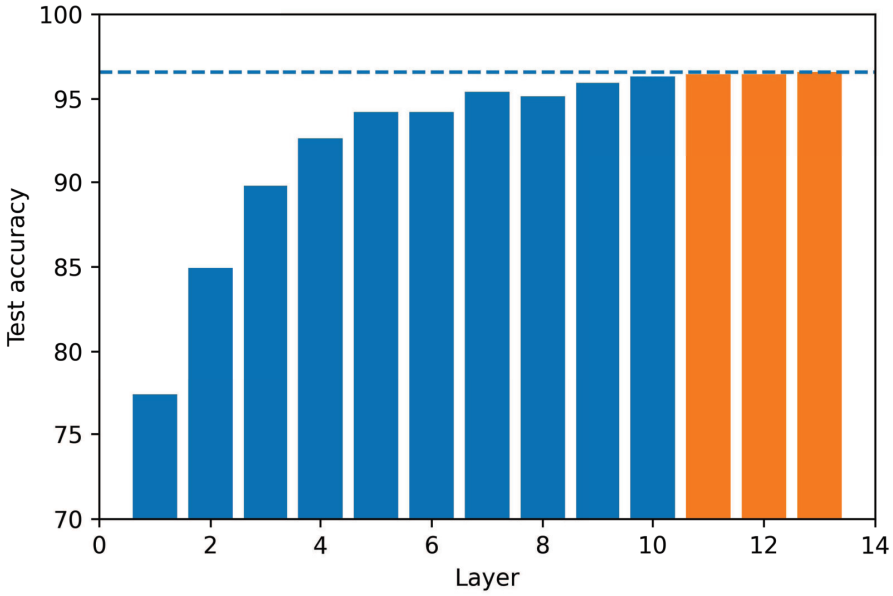

Fig. 6. Visualization of feature diagnosis.

Table 2. Performance comparison between VggBN-16, Vgg_BeanNet and network after pruning

Model(Filter pruning ratio)	Params(M)	FLOPs	Time(ms)	Acc(%)	F1_score(%)
VggBN-16(Baseline)	14.82	15.39G	9.53	96.59	96.58
Vgg_BeanNet	7.74	14.02G	8.43	96.47	96.47(↓0.11)
Vgg_BeanNet(66.67%)	0.81	1.98G	2.25	96.99	97.00(↑0.42)
Vgg_BeanNet(90.48%)	0.07	210.92M	2.07	95.93	95.97(↓0.61)

Channel pruning and knowledge distillation. From the visualization results of the output feature map of each convolutional layer of the VggBN-16 model, it can be found that there are many similar feature maps, which indicates that there is still redundant information of the parameters stored in the high-dimensional tensors, so we use the channel pruning method based on Taylor to further compress the Vgg_BeanNet. The relationship between pruning rate and model precision is shown in Fig. 7. The results show that the model still has good performance when the pruning rate is less than 70%. Sometimes the performance of the pruning model (66.67% filter pruning) is even higher than that of the original model, which may be due to overfitting caused by too many parameters, and network pruning is essentially a problem of searching the optimal network structure. When the pruning rate is more than 70%, the accuracy

of the Vgg_BeanNet is significantly reduced, but the compression effect of the model is better. Specifically, we also report the test results of filter pruning rate of 66.67% and 90.48% in Table 2.

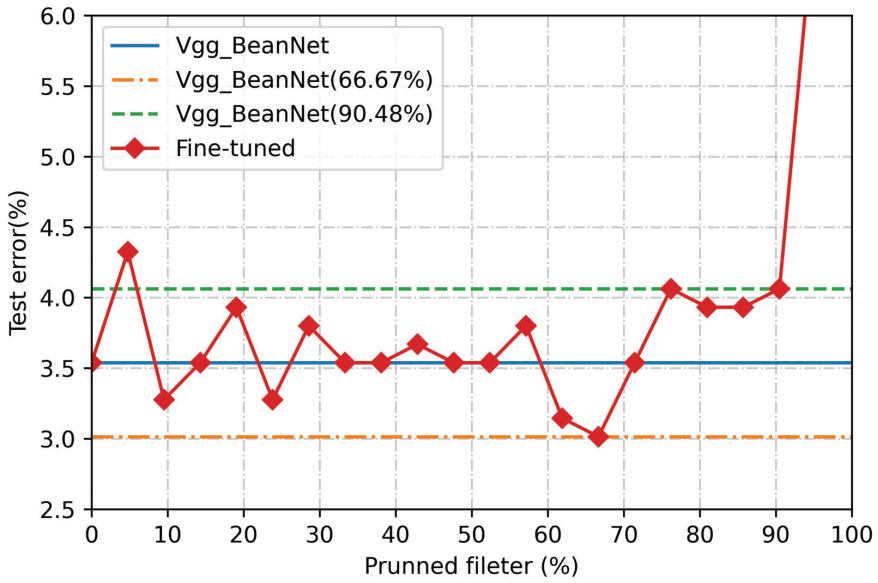

Fig. 7. The effect of pruning varying percentages of channels.

Meantime, we adopt knowledge distillation technology to improve the recognition accuracy of the Vgg_BeanNet(90.48%). From the above experimental results, we can see that GoogleNet and Vgg_Beannet (66.67%) have achieved good recognition accuracy of 96.85%, and 96.99% respectively. Therefore, we construct a multi-teacher model to guide students to learn the feature information of the teacher model and narrow the gap between them. Table 3 compares the performance improvement of the Mealv2 method and KD [9] on the Vgg_BeanNet (90.48%) and Alexnet networks. From the experimental results, we can find that both methods are significantly improved and Mealv2 obtains better results.

Table 3. Accuracy comparison before and after knowledge distillation experiment

Model(Filter pruning ratio)	Time(ms)	Scrach	Fitune	KD [9]	Mealv2
Vgg_BeanNet(90.48%)	2.07	95.67	95.97	96.07	96.33
Alexnet	2.31	88.60	—	93.05	94.62

Performance comparison of lightweight network In addition, we also compared with typical handcrafted lightweight networks (Mobilenetv2 [20], ShfleNetv2 [21], GhostNet [22], and Mixnet [23]). As shown in Table 4, in the case of similar computational complexity, the pruned network also gets the best classification accuracy (96.33%) and inference speed (2.07ms), this also shows that the proposed method does not rely too much on expert experience and is highly efficient under actual deployment conditions.

Table 4. Performance comparison before and after knowledge distillation experiment

Model(Filter pruning ratio)	Params(M)	FLOPs	Time(ms)	Acc(%)	F1_score(%)
Mobilenetv2 [20]	2.23	318.96M	7.83	95.94	95.93
ShuffleNetv2 [21]	1.26	149.58M	10.01	94.10	94.07
GhostNet [22]	3.91	149.4M	15.56	94.36	94.28
MixNet [23]	3.48	357.63M	22.23	94.89	94.88
Vgg_BeanNet(90.48%)	0.07	210.92M	2.07	96.33	96.33

4 Conclusions

To solve the problem of a large number of parameters and low efficiency of a deep learning model for seed sorting, this paper propose a mixed pruning strategy based on feature representation and Taylor expansion to achieve fast and high-purity seed grading. Moreover, we introduce a simple and effective knowledge distillation technology to transfer the knowledge of CNNs with high accuracy to the pruned network to recover its predictive accuracy. The experimental results verify the reliability of the scheme on red kidney bean data sets, which not only ensures the accuracy of classification but also reduces the volume of the network model. Meantime, experiments on Quadro m5000 GPU verify that the compressed model has better performance and inference speed on mobile devices than some cleverly designed lightweight CNN networks such as mobilenetv2 and shufflenetv2, etc.

5 Acknowledge

This work was supported by NSFC (U1804157, No. 61772576, No. 62072489), Henan science and technology innovation team (CXTD2017091), IRTSTHN (21IRTSTHN013), ZhongYuan Science and Technology Innovation Leading Talent Program (214200510013). Program for Interdisciplinary Direction Team in Zhongyuan University of Technology.

References

1. Sofu M M, Er O, Kayacan M C, et al. Design of an automatic apple sorting system using machine vision[J]. Computers and Electronics in Agriculture (127), 395–405 (2016).
2. Y. Altuntaş, A. F. Kocamaz, R. Cengiz and M. Esmeray, "Classification of haploid and diploid maize seeds by using image processing techniques and support vector machines". 2018 26th Signal Processing and Communications Applications Conference (SIU), 1–4, (2018) doi: 10.1109/SIU.2018.8404800.
3. Choudhary R, Paliwal J, Jayas D S. Classification of cereal grains using wavelet, morphological, colour, and textural features of non-touching kernel images[J]. Biosystems engineering, **99**(3), 330–337 (2008).
4. Jin H Y , Jin K S , Dayeon K , et al. Super-High-Purity Seed Sorter Using Low-Latency Image-Recognition Based on Deep Learning[J]. IEEE Robotics and Automation Letters (3), 3035–3042 (2018).

5. L. C. Uzal, G. L. Grinblat, R. Namías et al., "Seed-per-pod estimation for plant breeding using deep learning," Computers and Electronics in Agriculture (14), 196–204 (2018).
6. Li C, Li H, Liu Z, Li B, Huang Y. SeedSortNet: a rapid and highly efficient lightweight CNN based on visual attention for seed sorting. PeerJ Computer Science 7:e639 (2021) https://doi.org/10.7717/peerj-cs.639.
7. Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search[J]. arXiv preprint arXiv:1806.09055, (2018).
8. K. Wang, Z. Liu, Y. Lin, J. Lin and S. Han, "HAQ: Hardware-Aware Automated Quantization With Mixed Precision," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8604–8612 (2019).
9. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, (2015).
10. Shen Z, Savvides M. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks[J]. arXiv preprint arXiv:2009.08453, (2020).
11. Wang Z, Liu X, Huang L, et al. Model Pruning Based on Quantified Similarity of Feature Maps[J]. arXiv preprint arXiv:2105.06052, (2021).
12. Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient inference[J]. arXiv preprint arXiv:1611.06440, (2016).
13. Elkerdawy S, Elhoushi M, Singh A, et al. To filter prune, or to layer prune, that is the question[C]. Proceedings of the Asian Conference on Computer Vision. (2020).
14. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, (2014).
15. Zhang T, Ye S, Zhang K, et al. A systematic dnn weight pruning framework using alternating direction method of multipliers[C]. Proceedings of the European Conference on Computer Vision (ECCV).184–199 (2018).
16. Smirnov E A, Timoshenko D M, Andrianov S N. Comparison of regularization methods for imagenet classification with deep convolutional neural networks[J]. Aasri Procedia, (6), 89–94 (2014).
17. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778 (2016).
18. Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 4700–4708 (2017).
19. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 1–9 (2019).
20. Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 4510–4520 (2018).
21. Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]. Proceedings of the European conference on computer vision (ECCV). 116–131 (2018).
22. Han K, Wang Y, Tian Q, et al. Ghostnet: More features from cheap operations[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.1580–1589 (2020).
23. Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. CoRR, abs/1907.09595, (2019).