ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

Joint Distillation Training Scheme Based on Lightweight VGG

Shuiping Ni, Yizhe Zhang, Mingfu Zhu, Xinliang Ma, Wendi Wang

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China.

Abstract

Model compression through lightweight network design often incurs a decline in accuracy, and the performance gap between teacher and student networks in conventional knowledge distillation can hinder distillation efficacy. To address these issues, we propose a novel joint distillation training scheme anchored on a lightweight variant of the VGG network. The method incorporates multiple branch structures into the neural network, establishing a self-distillation framework for online distillation. Subsequently, it uses a pre-trained original network to perform offline distillation on this self-distillation framework, thus realizing a joint distillation training process. Simultaneously, group convolutions are adopted to lighten VGG11(BN) and VGG16(BN) networks, after which they undergo joint distillation training. Experimental results confirm that this training strategy effectively boosts distillation performance while maintaining the efficacy of the lightweighted networks.

Keywords

Neural Network; Image Classification; Knowledge Distillation; Self-distillation.

1. INTRODUCTION

As convolutional neural networks have undergone extensive research and advancement, deep neural network algorithms have gained broad adoption, demonstrating impressive performance in various domains including image classification, object detection, and semantic segmentation. However, achieving optimal network performance necessitates substantial storage and computational resources, which poses significant constraints when deploying these networks on resource-constrained devices and in real-time applications. In recent times, several model compression techniques have emerged to tackle these challenges by minimizing computational demands and storage requirements to align with hardware limitations. Notable methods include pruning [1], quantization [2], lightweight network architectures [3], and knowledge distillation [4]. Pruning involves eliminating connections with weights below a certain threshold, whereas quantization converts the floating-point computations of a neural network into fixed-point operations. Lightweight network design aims to streamline networks primarily in terms of size and speed while attempting to maintain high levels of accuracy. Knowledge distillation, on the other hand, transfers the learned knowledge from a powerful, complex teacher model into a simpler student model. In the realm of lightweight network design, models like MobileNet [5] and EfficientNet [6] offer promising solutions. While such lightweighting strategies can indeed be implemented within large-scale neural networks such as VGG and ResNet, it is an inherent challenge that some degree of accuracy deterioration typically accompanies the process of network lightweighting. Despite this trade-off, researchers continue to explore ways to minimize the loss of accuracy during network compression without compromising its overall functionality.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

As a straightforward yet potent model compression technique, knowledge distillation transfers knowledge from a pre-trained, large-scale teacher model to a smaller student model, thereby significantly enhancing network classification accuracy, and serving as an effective means to improve overall network performance. Since its introduction, knowledge distillation approaches have found widespread application across diverse fields. In practical applications, the student model learns from the output values of the teacher model, thereby reducing its reliance on the label values of datasets. There are primarily two modes of knowledge distillation: offline distillation and online distillation. In offline distillation, a pre-trained model with fixed parameters acts as the teacher model to guide the distillation process for the student model. For instance, the Evolutionary Embedding Learning Framework proposed by Wu et al. [7] efficiently transfers a considerable amount of knowledge from the teacher network to the student network. However, offline distillation does not allow for real-time adjustments in the knowledge refinement process according to the student model's current learning status, and the student model heavily depends on the teacher model. Online distillation, on the other hand, addresses the limitations of offline distillation by simultaneously updating the parameters of both the teacher and student models during training. This approach, as exemplified by the Mutual Contrastive Learning Online Knowledge Distillation proposed by Yang et al. [8], enables each network to learn additional comparative knowledge from the other, thus facilitating the acquisition of better feature representations. Nonetheless, online distillation increases computational resource consumption due to the concurrent training of both the teacher and student models. Additionally, self-distillation can be regarded as a particular case of knowledge distillation. Here, the teacher and student models stem from the same network. For example, Zhang et al. [9] introduced a setup where a deeper sub-network classifier was used as the teacher model to distill knowledge from the shallow layers of the original network, thereby boosting the performance of convolutional neural networks. However, self-distillation may lead to issues such as circular dependencies or overfitting to the model's own characteristics, especially when no extra data or noise is incorporated into the process.

Given the observed decrease in accuracy post-network lightweighting and the limitations of existing knowledge distillation methods, a joint distillation training scheme is proposed. This strategy integrates both offline and online distillation techniques. Firstly, the scheme capitalizes on the hierarchical modular structure of the neural network, introducing attention modules and shallow layers sequentially to build multiple branch classifiers within each module, thus constructing a self-distillation framework. Subsequently, the entire self-distillation framework is guided by a pre-trained original network on the same task, which has already undergone conventional training, thus realizing joint distillation training and enhancing the classification accuracy of the original network within the framework. Moreover, the neural network undergoes a lightweighting process to reduce the number of parameters, thus achieving model compression. The compressed network is then subjected to the joint distillation training scheme to further refine its performance, ultimately boosting its classification accuracy.

2. RELATED WORK

2.1. Knowledge Distillation

Knowledge distillation stands out as a prevalent technique in model compression, concurrently serving as a potent training strategy to enhance network classification accuracy. The essence of this method lies in harnessing a teacher network to mentor the training of a student network, enabling the latter to internalize the cumulative expertise the former acquires when addressing identical tasks, thereby bolstering the student network's performance. This groundbreaking principle was initially introduced by Hinton et al. [10], employing a large-scale teacher model to supervise the education of a smaller student counterpart. Since then, the field

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

of knowledge distillation has flourished with ceaseless innovation. Zagoruyko et al. [11] contributed to this progression by defining two categories of spatial attention maps and devising several novel approaches to shift attention from a bulky teacher model to a more compact student model, thereby amplifying the student model's proficiency. In parallel, Zhang et al. [12] conceived a deep mutual learning paradigm that actively fosters collaborative learning and reciprocal instruction among multiple models throughout the training regimen. On another front, Yang et al. [13] proposed a versatile feature-based knowledge extraction method applicable across a wide array of tasks. Lastly, Zhao et al. [14] made strides by decoupling the distillation loss into target class loss and non-target class loss components, thereby transcending the limitations imposed by classical loss functions.

Self-distillation is an improved variant of knowledge distillation that encourages a network to assume both teacher and student roles simultaneously. Zhang et al. [9] introduce a deep sub-network classifier as the teacher model to distill knowledge from the shallow layers of the original network. Enhanced self-distillation schemes, such as those in [15][16], integrate attention mechanisms into sub-network classifiers to further optimize the self-distillation framework. Ji et al. [17] propose a feature self-distillation method that utilizes soft labels and feature maps, guiding the distillation of peer-level feature maps by integrating and refining features from different depth layers. Given that both the teacher and student models in self-distillation are classifiers within the same neural network, this mitigates the impact of the performance gap between teacher and student models on distillation effectiveness seen in traditional knowledge distillation. The joint distillation training scheme combines aspects of both offline and online distillation methodologies and absorbs the benefits of self-distillation. It constructs additional student models to supplement and enhance the overall model performance. By doing so, it offers a comprehensive training approach that capitalizes on the strengths of different distillation techniques to maximize the potential of the distilled models.

2.2. Model Compression

The objective of model compression is to develop a network that is computationally swift and possesses a reduced number of parameters, with the aim to transform a large-scale model into a compact and leaner one. After compression, the network boasts a smaller architectural layout and fewer parameters, thereby efficiently reducing computational and storage demands, making it more suitable for deployment within resource-constrained hardware setups. A common approach to achieving model compression is through lightweight network design. For instance, MobileNetV1[18] utilizes depthwise separable convolutions to replace conventional convolutions, its network configuration consists of a series of such layers. By adjusting width multiplier and resolution multiplier factors, it allows for a flexible control over the model's size and computational complexity. MobileNetV2[5] introduces an inverted residual structure that mitigates computational costs without compromising information retention. ShuffleNetV1[19], in contrast, makes use of group convolutions combined with channel shuffling operations to enhance information exchange among distinct groups. ShuffleNetV2[20] explores network design from the perspective of memory access cost and GPU parallelism, scrutinizing methods to minimize runtime, thus achieving a more direct improvement in model efficiency. It presents four practical guiding principles for efficient network design based on both theoretical calculations and empirical conclusions. In addition to lightweight design strategies, there are other model compression techniques such as pruning and quantization. For example, Fang et al. [21] have made advancements in automating structured pruning, successfully implementing a structurally universal pruning technique. In this paper, our proposed scheme adapts the concept of grouped convolutions to the VGG network framework, effectively lightening and optimizing its computational profile.

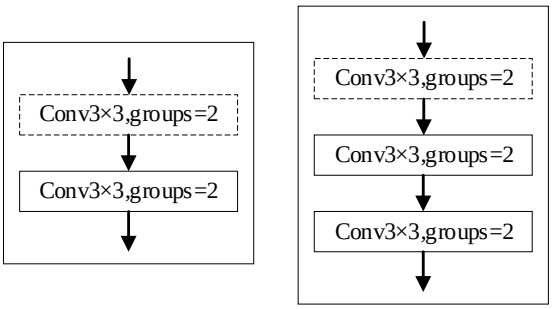
ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

3. SCHEME DESIGN

In this paper, the overall framework is composed of a lightweight network scheme and a joint distillation framework. Primarily, the lightweight network scheme attains network compression by applying the concept of grouped convolutions to the VGG network. Subsequently, the joint distillation scheme is executed to notably uplift the performance of the ensuing lightweight network.

3.1. Lightweight Network Scheme

The proposed scheme opts for lightweight processing on VGG11(BN) and VGG16(BN) neural networks. Each of these architectures is built around five convolutional stages, each comprising several convolutional layers. As illustrated in Figure 1, the two networks undergo targeted lightweight transformations based on this configuration.



（1）VGG11(BN) lightweight （2）VGG16(BN) lightweight

Figure 1. VGG network lightweight

Regarding VGG11(BN), its first two convolutional stages include just one convolutional layer each, while the subsequent three stages incorporate two convolutional layers per stage. In the last four stages of VGG11(BN), the regular convolutional layers are substituted with grouped convolutions featuring a kernel size of 3 and a group count of 2. Precisely, in stages hosting a single convolutional layer, only that layer is replaced; conversely, in stages with two convolutional layers, both layers are converted to the grouped convolutional format.

For VGG16(BN), the initial two convolutional stages harbor two convolutional layers, and the trailing three stages encompass three convolutional layers each. Analogously, within the final four convolutional stages of VGG16(BN), standard convolutional layers are traded for grouped convolutions having a kernel dimension of 3 and a grouping factor of 2. Here, in stages possessing two convolutional layers, both layers are exchanged, whereas in stages with three convolutional layers, all three layers are refashioned into grouped convolutional layers.

3.2. Joint Distillation Framework

The joint distillation training framework constructed upon the VGG11(BN) network is outlined in Figure 2. To initiate the process, the neural network is partitioned into four distinct regional modules based on its depth and inherent architecture. Next, attention modules and shallow modules are appended successively to the first three of these divided modules, thus forming three branch classifiers that constitute the self-distillation framework. Among them, branch classifier i (i=1,2,3) consists of Attention Module i, Shallow Module i, FC layer i, and Softmax i. To effectively align and guide the learning of these branch classifiers despite the differing dimensions of feature maps produced at different depths, auxiliary alignment layers are added. Unlike resorting to convolutional layers, fully connected (FC) layers are employed here specifically for this alignment purpose. Ultimately, a VGG11(BN) network, which has already undergone standard training on the same task, is leveraged to supervise and guide the

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

training of the entire self-distillation framework. By carrying out this procedure, joint distillation training is effectively implemented for the VGG11(BN) network. This process not only optimizes the network structure but also transfers and consolidates knowledge within the network itself, enhancing its overall performance and efficiency through a combination of self-distillation and the use of a pretrained VGG11(BN) model as a teacher network.

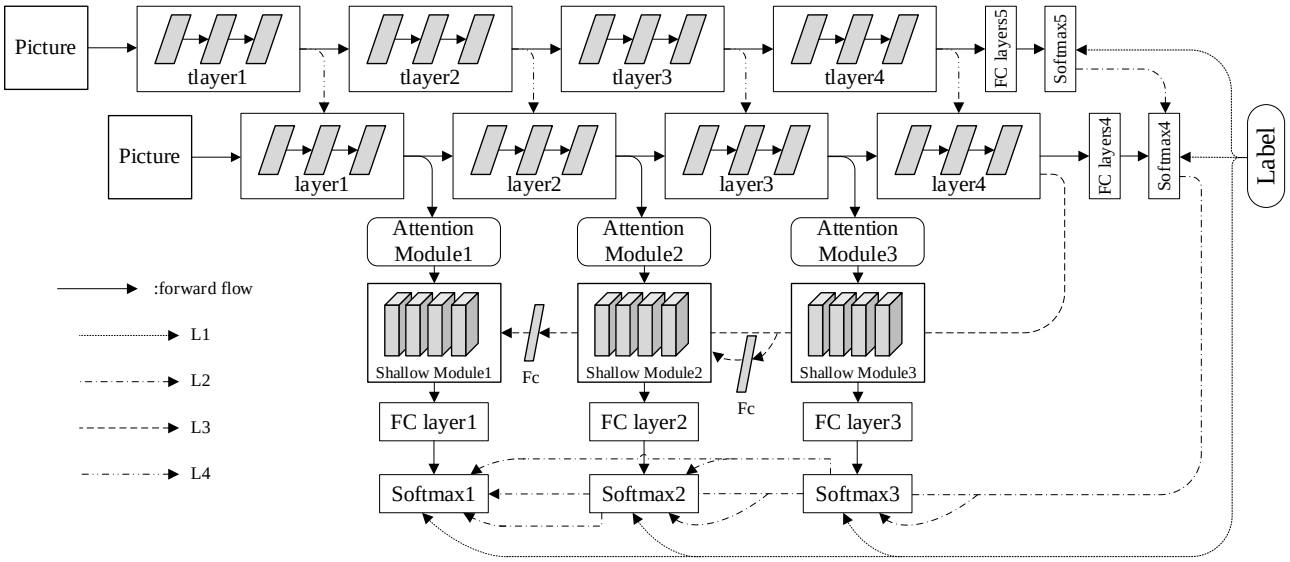


Figure 2. Joint distillation training framework

3.3. Branch Classifier

The three branch classifiers are each composed of attention modules and shallow modules, with the specific structures of these modules detailed in Figure 3.

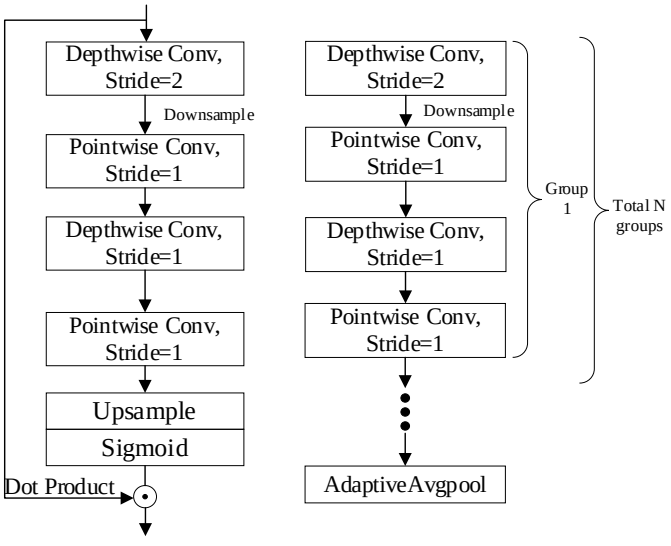


Figure 3. Branch classifier

In the attention module, the input feature map undergoes a sequence of depthwise convolution and pointwise convolution. Subsequently, an upsampling operation is performed followed by a sigmoid activation function. Then, a dot product operation is conducted between the processed feature map and the original input feature map, yielding the output of the attention module. Notably, after the depthwise convolution, a downsampling operation is

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

executed before entering the pointwise convolution stage. Here, the stride of the first depthwise convolution is set to 2, while the stride for all other convolutional operations is set to 1.

The shallow module consists of N sets of convolutional layers and adaptive pooling layers. Each set contains two depthwise convolutions and two pointwise convolutions. Similar to the attention module, the first depthwise convolution in each set has a stride of 2, with the stride for all other convolutional operations being 1. After the convolutional operations, an adaptive pooling layer is applied to produce the output of the shallow module. Within the framework, there are three branch classifiers, each featuring one attention module and one shallow module. In branch classifier i, the number of convolutional layer groups in its shallow module is $N = 4 - i$, where $i = 1, 2, 3$.

3.4. Distillation Loss

The joint distillation training framework comprises three branch classifiers and one deep classifier, all under the supervision of a teacher model. Given R classes and M samples $X = \{x_i\}_{i=1}^{M}$, the corresponding ground truth label set is represented as $\mathrm{Y} = \{y_i\}_{i=1}^{R}$, where $y_i \in \{1, 2, ..., R\}$. Assume in the joint distillation training scheme, the total number of classifiers in the neural network is C, denoted as $O = \{\theta_i\}_{i=1}^{C}$. A temperature-scaled Softmax function is attached to the end of each classifier.

$$q_i^C = \frac{\exp\left(z_i^C / \mathrm{T}\right)}{\sum_{j}^{R} \exp\left(z_j^C / \mathrm{T}\right)} \tag{1}$$

Here, z_i^C refers to the output of the classifier θ_C after the fully connected layer, and q_i^C signifies the classification probability of the i-th class given by classifier θ_C. T represents the distillation temperature, which is usually set to 1. When $T = 1$, equation (1) corresponds to a standard Softmax function. As the value of T increases, the produced labels become softer, meaning the probabilities assigned to non-maximum classes are less suppressed, providing richer gradient information for training.

First, the teacher network undergoes conventional training. During this process, the teacher network is solely supervised by the ground-truth labels. The training loss for the teacher network is computed using cross-entropy loss between the network's outputs and the true labels. The loss function for the teacher network's training is expressed as follows:

$$L_t = Cr\left(q^5, y\right) \tag{2}$$

Here, Cr represents the cross-entropy loss function, q^5 is the output of the teacher network's Softmax function with $T = 1$, representing the estimated probabilities for each class, y denotes the actual or ground-truth label values.

After the teacher network has finished its initial training, the self-distillation framework proceeds to undergo distillation training. Within the joint distillation training scheme, each classifier within the self-distillation framework encounters multi-source supervision during its training phase. Therefore, the design of the loss function for such classifiers comprises four distinct parts, aimed at harmonizing the influence of these various supervisory inputs. To effectively balance these sources of supervision, hyperparameters α, loss coefficients λ, feature loss coefficient β, and ensemble loss coefficient γ are introduced. These parameters serve to calibrate the weighting of the four-part loss function, ensuring that the contributions from different sources of supervision are appropriately balanced and integrated.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

L_1: The first component of the loss is the cross-entropy loss between the real labels and each classifier's output. This is calculated by using the dataset's true labels and the Softmax output of every classifier. This loss serves to impart the underlying knowledge encapsulated in the real labels into all the classifiers, thereby aligning their predictions with the actual class distributions in the dataset.

$$L_1 = \sum_{i=1}^{3} Cr\left(q^i, y\right) \cdot (1-\lambda) + (1-\alpha) \cdot Cr\left(q^4, y\right) \tag{3}$$

Here, q^i represents the output of the Softmax function with temperature $T=1$ for each classifier θ_i within the self-distillation framework, q^4 denotes the output of the Softmax function with temperature $T=1$ for the deep classifier, and Cr signifies the cross-entropy loss function.

L_2: The second loss term is derived from computing the Kullback-Leibler (KL) divergence loss between the Softmax outputs of the teacher model and the deep classifier of the student network, as well as between the teacher model and each of the branch classifiers, and further between the deep classifier and each of the branch classifiers in the self-distillation framework. By incorporating this KL divergence loss, the teacher network imparts its knowledge onto the self-distillation framework. This way, not only does the teacher network affect the overall self-distillation process, but the deep classifier within the self-distillation framework also influences the behavior of the branch classifiers, facilitating a flow of knowledge from the teacher to the deep classifier and subsequently to the branch classifiers. This mutual interaction refines the learning process and improves the performance of the entire network ensemble.

$$L_2 = \alpha \cdot KL\left(q^4, q^5\right) \cdot T^2 + \sum_{i=1}^{3} KL\left(q^i, q^4\right) \cdot \lambda + \sum_{i=2}^{3} KL\left(q^1, q^i\right) \cdot \gamma + KL\left(q^2, q^3\right) \cdot \gamma \tag{4}$$

Here, q^i represents the output of the Softmax function with temperature $T=3$ for each classifier θ_i within the self-distillation framework, q^4 denotes the output of the Softmax function with temperature $T=3$ for the deep classifier within the self-distillation framework, q^5 indicates the output of the Softmax function with temperature $T=3$ for the teacher network, KL signifies the KL divergence.

L_3: The third loss component is obtained by calculating the L2 loss between the feature maps of the deep classifier and each of the branch classifiers within the self-distillation framework. By employing the L2 loss calculation method, hidden layer knowledge from the feature layers is infused into the branch classifiers. This process guides the features of branch classifiers to adapt to those of the deep classifier, promoting consistency and knowledge transfer across the different levels of abstraction within the network.

$$L_3 = \beta \cdot \| F_i - F_C \|_2^2 \tag{5}$$

Here, F_i represents the features from classifier θ_i (where $i=1,2$, $i \neq 3$), F_C denotes the features from the deep classifier θ_C.

L_4: The fourth loss component is derived by calculating the mean squared error (MSE) loss between the four regions partitioned from the teacher network and the corresponding four regions partitioned from the target network within the self-distillation framework. This MSE loss enables the spatial attention features from each region of the teacher network to be effectively transferred and integrated into the self-distillation framework, thereby guiding the target network to learn and mimic the finer-grained spatial characteristics captured by the teacher network.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

$$L_4 = \beta \cdot \sum_{i=1}^{4} \text{MSELoss}\left(\frac{mean(X_i^2)}{\| X_i \|_2}, \frac{mean(TX_i^2)}{\| TX_i \|_2}\right) \quad (6)$$

Here, X_i denotes the output feature map of the student network layeri module (where $i = 1,2,3,4$); TX_i represents the output feature map of the teacher network tlayer i module (where $i = 1,2,3,4$); $MSELoss$ signifies the mean square loss function, and $mean$ refers to the operation of taking the average.

The total loss function includes the four components listed above, which can be written as:

$$Loss = L_1 + L_2 + L_3 + L_4 \quad (7)$$

4. EXPERIMENTS

4.1. Experiments Datasets

Training was carried out using the CIFAR10, CIFAR100, and Tiny-imagenet datasets respectively. The CIFAR10 dataset consists of images of size 32×32, divided into 10 classes, with 6,000 images per class. Among these, 5,000 images serve as the training set, while 1,000 images form the test set. The CIFAR100 dataset, on the other hand, encompasses 100 classes of 32×32 color images, with 600 images per class. The training set includes a total of 50,000 images, with 500 images from each class, and the testing set consists of 10,000 images, with 100 images distributed evenly among the 100 classes. Lastly, the Tiny-imagenet dataset features 200 classes, with 500 training images, 50 validation images, and 50 test images allocated to each class.

4.2. Experiments Settings

In the lightweight experiments, the networks used were VGG11(BN) and VGG16(BN). For the joint distillation experiments, the networks employed included ResNet18, VGG11(BN), VGG16(BN), MobileNetV2, and SqueezeNet. In both types of experiments, when training on CIFAR10 and CIFAR100 datasets, models were trained for 200 epochs each. With Tiny-imagenet dataset, training was conducted for 100 epochs.

All experiments employed the SGD optimizer to train the neural networks with a weight decay value of 5e-4 and a momentum of 0.9. An initial learning rate of 0.1 was set, which was decreased by a factor of 10 at the 66th, 133rd, and 190th epochs. The batch size was set to 128 across all experiments. These experiments were carried out on GPU devices using the PyTorch environment version 1.9.1. Moreover, recommended hyperparameters for the experiments included: $\alpha = 0.9$ for the hyperparameter, $\lambda = 0.3$ for the loss coefficient, $\beta = 0.03$ for the feature loss coefficient, and $\gamma = 0.01$ for the ensemble loss coefficient.

4.3. Lightweight Experimental Results and Analysis

In the lightweight experiments, VGG11(BN) and VGG16(BN) networks were subjected to network slimming, substituting specific convolutional layers with grouped convolutions featuring a kernel size of 3 and a group count of 2. Upon modification, the networks were retrained, and their combined evaluation metrics, including parameter counts, computational demands, and post-lightweighting accuracy, are presented in Table 1.

Table 1 illustrates that the parameter count and computational complexity of both VGG11(BN) and VGG16(BN) have been efficiently diminished across various datasets. VGG11(BN) witnessed a 17.5% decrease in its parameter quantity, and VGG16(BN) registered a reduction of 22.7%. Moreover, Table 1 also discloses that there was a corresponding decline in the accuracy of the networks following the lightweighting process. The VGG11(BN) network's accuracy rates on the CIFAR10, CIFAR100, and Tiny-imagenet datasets respectively decreased from 92.3%, 70.9%, and 54.4% to 91.7%, 68.8%, and 54.1%, with declines of 0.6%, 2.1%, and

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

0.3% respectively. Similarly, for the VGG16(BN) network, the accuracy on the CIFAR10, CIFAR100, and Tiny-imagenet datasets fell from 94.1%, 73.3%, and 57.0% down to 92.9%, 70.6%, and 54.3%, experiencing drops of 1.2%, 2.7%, and 2.7% correspondingly.

Table 1. VGG network parameters

Models	Params	FLOPs		Baseline accuracy		
		CIFAR100	Tiny-imagenet	CIFAR-10	CIFAR-100	Tiny-imagenet
VGG11(BN)	28.52M	229.34M	632.78M	92.3%	70.9%	54.4%
Group-VGG11	23.54M	125.16M	330.42M	91.7%	68.8%	54.1%
VGG16(BN)	34.02M	418.60M	1276.50M	94.1%	73.3%	57.0%
Group-VGG16	26.31M	238.93M	728.78M	92.9%	70.6%	54.3%

Table 2. Group-VGG11 experiment results of accuracy (%) on datasets

Classifier	CIFAR10		CIFAR100		Tiny-imagenet	
	SKD	Our approach	SKD	Our approach	SKD	Our approach
branch classifier1	92.1	92.4	73.0	72.4	60.8	61.4
branch classifier2	92.2	92.4	72.7	73.0	60.6	60.9
branch classifier3	91.9	92.4	71.9	72.0	57.5	58.4
deep classifier	91.8	92.2	71.1	72.3	56.5	58.4

Table 3. Group-VGG16 experiment results of accuracy (%) on datasets

Classifier	CIFAR10		CIFAR100		Tiny-imagenet	
	SKD	Our approach	SKD	Our approach	SKD	Our approach
branch classifier1	93.4	93.2	75.0	74.4	62.5	62.2
branch classifier2	93.1	93.4	74.2	74.7	63.1	62.4
branch classifier3	93.1	93.6	73.8	74.8	60.7	61.1
deep classifier	93.0	93.4	73.2	74.3	58.3	60.1

After the network slimming, self-distillation and joint distillation training were conducted on the three datasets, and the results are displayed in Tables 2 and 3. In these tables, 'SKD' directly represents cases where the self-distillation framework was implemented on the respective datasets, while 'our approach' denotes the usage of our joint distillation training method on each dataset.

The experimental findings show that the accuracy rates of both VGG11(BN) and VGG16(BN) networks improved, with average increases of 2.77% and 3.33%, respectively. It can be deduced that even after network slimming, employing the joint distillation scheme during training can indeed boost network precision, maintain or surpass the original accuracy rate of the network prior to slimming. This confirms that the joint distillation method can effectively compensate for any potential accuracy drop caused by network slimming, and potentially enhance the overall performance of the lightweight networks.

4.4. Joint Distillation Experimental Results and Analysis

In the joint distillation experiment, five image classification networks were selected and trained on three different datasets to evaluate the effectiveness of the joint distillation scheme by comparing its accuracy with the original network accuracy and the self-distillation accuracy reported in [16]. First, the baseline accuracy of the five networks was established through standard training. Subsequently, the joint distillation training framework was constructed using each of the five networks, and they were separately trained on the three datasets. The experimental outcomes for these configurations are respectively presented in Tables 4, 5, and 6. In these tables, 'SKD' signifies the accuracy attained via the self-distillation framework, while 'our approach' refers to the accuracy resulting from the employment of the joint distillation training method. Additionally, 'baseline' represents the original accuracy of the networks before any distillation. Branch Classifier 1, Branch Classifier 2, and Branch Classifier 3 refer to the

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

training accuracy of the three separate branches, while the Deep Classifier signifies the accuracy of the network when trained using the joint distillation approach.

Table 4. Experiment results of accuracy (%) on CIFAR10

Models	Baseline	Method	branch classifier1	branch classifier2	branch classifier3	deep classifier
ResNet18	95.0	SKD	94.4	95.1	95.6	95.7
		Our approach	93.8	95.2	95.7	95.9
VGG11(BN)	92.3	SKD	92.4	92.7	92.9	92.9
		Our approach	91.9	92.8	93.0	93.1
VGG16(BN)	94.1	SKD	93.8	94.2	94.1	94.2
		Our approach	93.7	94.3	94.3	94.3
MobileNetV2	90.8	SKD	89.2	90.9	90.8	90.9
		Our approach	90.4	92.0	91.9	91.9
SqueezeNet	93.0	SKD	93.1	93.4	93.4	93.3
		Our approach	93.2	93.4	93.6	93.9

Table 5. Experiment results of accuracy (%) on CIFAR100

Models	Baseline	Method	branch classifier1	branch classifier2	branch classifier3	deep classifier
ResNet18	77.8	SKD	75.8	77.8	78.5	79.0
		Our approach	75.5	77.7	78.8	80.4
VGG11(BN)	70.9	SKD	72.3	72.7	72.5	72.2
		Our approach	71.8	72.4	73.0	73.3
VGG16(BN)	73.3	SKD	74.7	75.2	75.4	75.3
		Our approach	74.3	75.5	75.6	75.8
MobileNetV2	68.9	SKD	66.9	69.4	70.0	69.9
		Our approach	68.5	71.0	71.8	71.6
SqueezeNet	71.0	SKD	74.0	74.3	74.9	71.8
		Our approach	73.9	74.6	74.9	73.1

Table 6. Experiment results of accuracy (%) on Tiny-imagenet

Models	Baseline	Method	branch classifier1	branch classifier2	branch classifier3	deep classifier
ResNet18	63.0	SKD	61.7	63.9	63.9	65.7
		Our approach	61.7	63.9	63.5	66.9
VGG11(BN)	54.4	SKD	59.6	60.7	58.7	57.5
		Our approach	59.0	59.5	58.8	59.8
VGG16(BN)	57.0	SKD	61.8	63.6	62.1	61.5
		Our approach	61.5	63.2	62.2	62.4
MobileNetV2	56.1	SKD	52.4	55.8	57.7	57.2
		Our approach	53.7	57.5	59.9	59.3
SqueezeNet	56.4	SKD	60.8	61.5	61.9	57.6
		Our approach	61.2	61.7	62.5	57.7

Based on the data presented in Tables 4, 5 and 6, it can be seen that across the three different datasets, the accuracy of the Deep Classifier consistently shows improvement, and the joint distillation training scheme outperforms self-distillation training in terms of accuracy gains. Moreover, most of the Branch Classifiers also demonstrate increased accuracy. Compared to the original networks, the accuracy has been raised by an average of 0.78%, 2.46%, and 3.84% on the respective datasets. When contrasted with the self-distillation framework, the average increase in accuracy is 0.42%, 1.2%, and 1.32% on the different datasets. These results indicate that the joint distillation training approach significantly enhances network precision and achieves higher classification accuracy than both the original networks and those trained via

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

self-distillation alone. This underscores the effectiveness of the joint distillation method in improving model performance.

5. DISCUSSION

5.1. Loss Function Effect

The joint distillation training scheme's loss function consists of four components: L_1, L_2, L_3, L_4. To investigate the specific roles played by each part, experiments were conducted using the VGG16(BN) network on two different datasets by employing varying combinations of these loss functions. The experimental outcomes are presented in Tables 7 and 8.

Table 7. VGG16(BN) experiment results of accuracy (%) on CIFAR100

classifier	L1+L2+L3+L4	L2+L3+L4	L1+L3+L4	L1+L2+L4	L1+L2+L3
branch classifier1	74.3	73.0	74.4	73.2	74.4
branch classifier2	75.5	74.3	75.4	74.5	74.8
branch classifier3	75.6	75.4	75.9	75.3	75.5
deep classifier	75.8	75.3	75.4	75.2	75.5

Table 8. VGG16(BN) experiment results of accuracy (%) on Tiny-imagenet

classifier	L1+L2+L3+L4	L2+L3+L4	L1+L3+L4	L1+L2+L4	L1+L2+L3
branch classifier1	61.5	58.1	60.9	58.7	60.9
branch classifier2	63.2	59.7	63.2	61.2	62.7
branch classifier3	62.2	55.2	61.4	62.1	62.4
deep classifier	62.4	61.0	59.6	62.3	62.3

From Tables 7 and 8, it can be observed that removing any of the individual loss components leads to a decrease in the classifier's performance. Specifically, the removal of L_1 causes the most significant impact on the performance of the branch classifiers, whereas eliminating L_2 affects the deep classifier's performance the most. The removal of L_3 and L_4 also has a noticeable effect on both the branch classifiers and the deep classifier.

Consequently, the presence of all four components (L_1, L_2, L_3, and L_4) in the loss function contributes to the high performance of the overall joint distillation training scheme. This implies that each component plays a crucial role in guiding the training process and optimizing the network's ability to accurately classify the input data. Therefore, maintaining all four parts of the loss function ensures that the joint distillation framework attains optimal performance across different datasets and architectures such as VGG16(BN).

5.2. Attention Analysis

In the joint distillation training scheme, attention modules are introduced to capture specific features of the image, thereby enhancing the performance of the branch classifiers. In the context of joint distillation training based on the VGG11(BN) network, the attention feature maps were further visualized, as illustrated in Figure 4. Attention1, Attention2, and Attention3 denote the output feature maps of the input image after passing through the attention module in the three respective branch classifiers.

From Figure 4, it is evident that after going through the attention module, the classifiers become more focused on detailed features of the image, such as paying closer attention to animal eyes, ears, heads, etc. As we progress from Branch 1 to Branch 3, there is a clear increase in the variety of features the classifiers attend to, indicating that deeper classifiers are capable of extracting richer and more diverse feature information from the image.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

Figure 4. Visualization of feature maps from the Attention Module

5.3. Scheme Comparison

In order to validate the effectiveness of the joint distillation training scheme, comparative experiments were conducted against other similar yet distinct distillation methods. Three networks were chosen for experimentation using the CIFAR100 dataset. The results, as shown in Table 9, demonstrate that the joint distillation training scheme yields an average accuracy improvement of 2.5% across the three networks compared to the other distillation strategies, which exhibit slightly lower accuracy improvements.

Table 9. Comparison with other methods on CIFAR100

Teacher	Student	Baseline	DML[12]	MGD[13]	DKD[14]	SCAN[15]	FRSKD [17]	Ours
ResNet18	ResNet18	77.8	76.3	78.5	77.7	79.4	74.5	80.4
VGG11(BN)	VGG11(BN)	70.9	70.3	71.3	—	72.8	69.0	73.3
VGG16(BN)	VGG16(BN)	73.3	74.0	74.0	67.3	74.9	75.6	75.8

From these results, it can be inferred that when the teacher and student networks employ the same type of neural network architecture, the joint distillation approach proves more effective. Combining offline distillation with online distillation in the joint distillation framework effectively boosts network performance, leading to better classification accuracy. Thus, integrating multiple distillation techniques in the joint distillation scheme offers a superior strategy for enhancing network precision and overall efficiency.

6. CONCLUSION

A novel training scheme has been proposed that integrates lightweight network design with joint distillation. In the lightweight network aspect, group convolutions were introduced,

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

leading to a reduction of 17.5% in the parameter count for VGG11(BN) and 22.7% for VGG16(BN). Subsequent joint distillation training resulted in improved accuracy for the slimmed-down networks, sometimes surpassing the accuracy before the lightening process.

Employing joint distillation across several networks demonstrated an average improvement of 2.36% in accuracy over the original networks across various datasets, and an average boost of 0.98% compared to self-distillation. Furthermore, the integration of multiple losses in the joint distillation framework contributed to its high performance. The inclusion of a branching structure allowed the network to focus more closely on the image's salient features.

In conclusion, the proposed joint distillation training scheme effectively enhances the distillation process and raises the classification accuracy of the networks. Notably, it maintains or improves the performance of the networks even after lightweight modifications, making it a promising approach for achieving both compactness and improved accuracy in neural network models.

ACKNOWLEDGMENTS

This paper is supported by the Key Research and Development Program in Henan Province (231111210500) and the National Key R&D Program of China: Key technical equipment for disaster monitoring, warning, and information acquisition based on communication big data (2023YFC3010700).

REFERENCES

[1] Zeng W, Xiong Y, Urtasun R. Network automatic pruning: start nap and take a nap[J]. arXiv preprint arXiv, 2021, 2101: 06608.

[2] Yao Z, Dong Z, Zheng Z, et al. Hawq-v3: dyadic neural network quantization[C]//International Conference on Machine Learning. PMLR, 2021: 11875-11886.

[3] Qin Z, Li Z, Zhang Z, et al. ThunderNet: towards real-time generic object detection on mobile devices[C]//Proc of the 17th IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE Press, 2019: 6718-6727.

[4] Gou J, Yu B, Maybank S J, et al. Knowledge distillation: A survey[J]. International Journal of Computer Vision, 2021, 129: 1789-1819.

[5] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: inverted residuals and linear bottlenecks[C]//Proc of the 31th IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE Press, 2018: 4510- 4520.

[6] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105- 6114.

[7] Wu X, He R, Hu Y, et al. Learning an evolutionary embedding via massive knowledge distillation[J]. International Journal of Computer Vision, 2020, 128: 2089-2106.

[8] Yang C, An Z, Zhou H, et al. Online knowledge distillation via mutual contrastive learning for visual recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

[9] Zhang L, Song J, Gao A, et al. Be your own teacher: Improve the performance of convolutional neural networks via self distillation[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 3713-3722.

[10] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202407_10(7).0002

[11] Zagoruyko S, Komodakis N. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer[C]//ICLR. 2017.

[12] Zhang Y, Xiang T, Hospedales T M, et al. Deep mutual learning[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4320-4328.

[13] Yang Z, Li Z, Shao M, et al. Masked generative distillation[C]//European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022: 53-69.

[14] Zhao B, Cui Q, Song R, et al. Decoupled knowledge distillation[C]//Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. 2022: 11953-11962.

[15] Zhang L, Tan Z, Song J, et al. Scan: A scalable neural networks framework towards compact and efficient models[J]. Advances in Neural Information Processing Systems, 2019, 32.

[16] Zhang L, Bao C, Ma K. Self-distillation: Towards efficient and compact neural networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 44(8): 4388-4403.

[17] Ji M, Shin S, Hwang S, et al. Refine myself by teaching myself: Feature refinement via self-knowledge distillation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 10664-10673.

[18] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv: 1704.04861. 2017.

[19] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[20] Ma N, Zhang X, Zheng H T, et al. ShuffleNet V2: practical guidelines for efficient CNN architecture design[C]//Proc of the ECCV. Cham: Springer, 2018: 122-138.

[21] Fang, Gongfan, et al. "Depgraph: Towards any structural pruning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.