SCKD: STAGE-CORRELATION KNOWLEDGE DISTILLATION

Qi Wang[1], *Lu Liu*[1], *Wenxin Yu*[1,✉], *Shiyu Chen*[1,*], *Jun Gong*[2], *Peng chen*[3]

[1]Southwest University of Science and Technology,
[2]Beijing Institute of Technology, [3]Chengdu Hongchengyun Technology Co., Ltd
yuwenxin@swust.edu.cn

ABSTRACT

In this paper, we present Stage-Correlation Knowledge Distillation (**SCKD**), a novel and efficient knowledge distillation method, conditioned on the correlation of stage in structure, different from logtis and feature maps methods and achieve good performance. The whole structure divides into three parts. The first part is the cross-entropy Loss (CE), the second part is original knowledge distillation (KD), and last but not least is our proposed **SCKD**. By our proposed method, the traditional knowledge distillation can be significantly enhanced, and our method is not limited by the size of feature maps and insufficient samples in small datasets. Our method is validated on CIFAR100 and CIFAR10 datasets. The experimental results demonstrate the effectiveness and superiority of our method. Meanwhile, our method's results surpass most current methods based on logits or feature maps.

Index Terms— Knowledge distillation, Stage Correlation, MLP

1. INTRODUCTION

Deep convolutional neural networks (CNNs) have achieved significant success in computer vision domain. However, the CNNs are often accompanied by substantial computational effort and memory consumption, making their application on resource-limited devices be a challenging task. For this purpose, various approaches have been proposed over the past few years, including designing new architectures [1, 2, 3], network pruning [4, 5, 6], quantization [7] and knowledge distillation [8, 9, 10]. In these approaches, knowledge distillation aims to compress a powerful yet cumbersome teacher model into a lightweight student model without much sacrifice of performance.

In 2015, Geoffrey Hinton et al. [11] first proposed the concept of "Knowledge Distillation" and added "Temperature (T)" into the softmax function. Subsequently, some classical distillation methods have been proposed, such as [12, 13, 14]. These methods mainly study the feature maps of the intermediate layer between the teacher model and the student model to improve the accuracy of the student model. Later, to further improve the performance of the student model, some methods combine knowledge distillation with contrastive learning, such as CRD [15]. Although these methods have achieved excellent results, there is still a gap between the teacher and the student in the knowledge distillation domain.

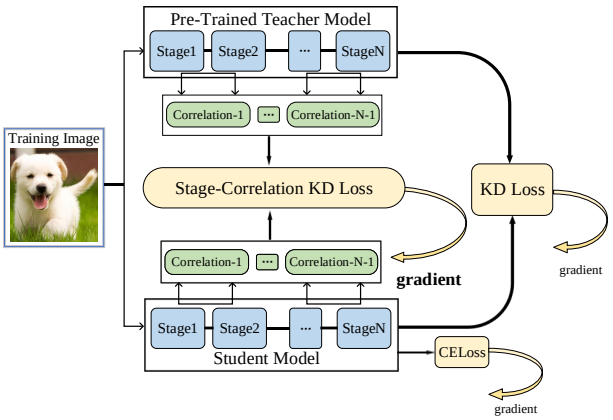

Fig. 1. Overall architecture diagram of **SCKD**. The detailed of our architecture and design of loss function are described in Section 2.

To achieve better performance and narrow the gap in model ability between teachers and students. We propose a novel knowledge distillation method, stage-correlation knowledge distillation, called **SCKD**. Unlike the commonly used logits and intermediate feature maps methods, or combine knowledge distillation with contrastive learning methods. Our **SCKD** applies the correlation between stages in the network and discovers a new "knowledge" for distillation, applicable to almost all network architectures and can combine with many other distillation methods.

Overall, our main contributions are as follows:

- A novel and efficient knowledge distillation method: Stage-Correlation Knowledge Distillation (**SCKD**), is proposed by us, different from logtis and feature maps methods, does not introduce overhead, and achieves good performance.

*This research is supported by Sichuan Science and Technology Program (No.2020YFS0307), Mianyang Science and Technology Program (2020YFZJ016), Sichuan Provincial M. C. Integration Office Program, and IEDA laboratory of SWUST.

- The experimental results on CIFAR100 [16] and CIFAR10 [16] datasets demonstrate the effectiveness and superiority of our method.
- We pioneer a knowledge distillation method different from the conventional methods, which provides a new idea for the study of knowledge distillation in the future.

The rest of this paper is arranged as follows. The details of our structure and design of the loss function are discussed in Section 2. Experimental results are presented in Section 3. Section 4 summarizes our work.

2. METHOD

The overall architecture of our method is shown in Figure 1. We divide the network architecture into three parts. The first part is to compare the final outputs of the student model with the ground truth, is for the classification task. And then, like other knowledge distillation methods, we use the original knowledge distillation method (KD) [11] to train the student model and transfer the "knowledge" learned by the teacher model to improve the performance of the student model. The last and most important part, we use **SCKD** (Ours) to improve the performance of the student model further and narrow the gap between the teacher and student models.

2.1. Network Architecture

As Geoffrey Hinton et al. [11] say, we tend to distill the parameters learned by the teacher model, but the difference in structure between the teacher model and the student model prevents us from using it. In order to solve this issue, we use resnet32x4 [17] and resnet8x4 [17] as the infrastructure and refer to outputs of the residual block as the stage, or general blocks in other models, just as ShuffleNet [2], vgg [18], which is represented in Figure 1 as Stage1, ..., StageN. Meanwhile, we argue that the correlation between adjacent stages can be disguised to represent the parameters learned by the model, which is denoted in Figure 1 as Correlation-1, ..., Correlation-N-1. Since, we make the following definitions: collection of stages *S* and collection of correlations *C*.

$$S = \{s_1, s_2, s_3, ..., s_i, s_n\} \tag{1}$$

$$C = \{c_1, c_2, c_3, ..., c_j, c_{n-1}\} \tag{2}$$

where *n* is the number of the stages, s_i is outputs of the i-th stage, c_j is the j-th correlation. The equation to obtain the collection of correlations *C* is as follows:

$$s_{j-1} = adaptive\_avg(s_{j-1}) \tag{3}$$

$$c_j = softmax(avg(s_{j-1} * s_j^T)) \tag{4}$$

The $adaptive\_avg$ in Equation 3 is convolution kernel adaptive function to getting the average value in spatial dimension of feature maps, from existing methods of PyTorch [19]. After this function, the size of feature maps in s_{j-1} is adjusted to be consistent with s_j. The s_j^T denoted the transpose of s_j, and the avg in Equation 4 denoted the average pooling operation. Due to the correlation representing the relationship between two neighboring stages, so, the number of correlations is n-1.

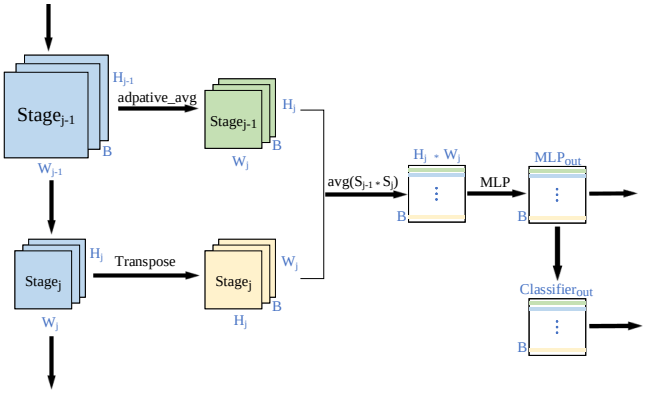

Fig. 2. The detailed processing flow to obtain the correlation c_j. The dimension of c_j is $\{B, H_{s_j} * W_{s_j}\}$, where B is denoted batch size and H_{s_j}, W_{s_j} is denoted the size of H and W in s_j. The MLP_{out} is denoted the output size of MLP, and the $Classifier_{out}$ is denoted the final output size of the teacher model.

After Equation 4, an additional Multi-Layer Perceptron (MLP) is developed to help the correlation of student match with the teacher, which renders our technique applicable to various teacher and student architectures. In addition, we argue that a teacher model's powerful class prediction ability is credited to not only those expressive features but, just as importantly, a discriminative classifier. Based on this argument, we put the output embedding of MLP into the classifier of the teacher model. Figure 2 shows the detailed processing flow. So, we make the following definitions:

$$MLP_{out} = L2_{Norm}(Linear(ReLu(Linear(c_j)))) \tag{5}$$

$$Classifier_{out} = Classifier(MLP_{out}) \tag{6}$$

where *Linear* is a linear layer, $ReLu$ is the relu activation function, $L2_{Norm}$ is the L2 Normalization, and the $Classifier$ is the linear classifier of the teacher model. After training, the teacher and the student get the exact size output embedding from MLP_{out} and $Classifier_{out}$, respectively. Furthermore, the processing flow is consistent in the teacher and student models.

Through this processing series, we can project the correlation of the student and teacher models into the same embedded feature space. Due to the teacher model being pre-trained and its correlations between adjacent stages being similar, it has a concentrated distribution for different samples in the

embedded feature space. In contrast, the untrained student model has a divergent distribution. Therefore, we believe that the student model student can learn the correlation of the teacher model's adjacent stages by the teacher's distribution in embedded feature space.

2.2. Loss Function

By the method proposed in section 2.1, correlations from adjacent stages of the student and teacher models are projected into the same embedded feature space. Therefore, we need to select an appropriate loss function to calculate the similarity of embedded features between the teacher and student models. Due to the embedding obtained by untrained students being wildly divergent, if we use the MAE (L1-Loss) or MSE (L2-Loss) directly, the performance of training will be poor. So we choose the Huber loss as our loss function. In terms of gradient and model convergence, the MSE is better than the MAE, but the MAE is significantly better than the MSE in handling outliers. The equation of Huber loss is shown below:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & |y - f(x)| > \delta \end{cases} \quad (7)$$

Where δ defaults to 1. In addition, final tensor dimension of our **SCKD** is $\{C, B, M\}$ and $\{C, B, L\}$ after Equation 5, 6 , C is the number of the correlations, B is batch size, M is size of outputs from MLP_{out} and L is size of outputs from $Classifier_{out}$. To capture more information, we compute the loss from the embedding dimension (M, L dimension) and batch size dimension (B dimension) to increase the diversity of the loss function. The equation is shown below:

$$\sigma(x) = \frac{1}{b}\sum_{i}^{b} x_i \quad \rho(x) = \frac{1}{m}\sum_{i}^{m} x_i \quad (8)$$

$$L_{bm}(c_t, c_s) = L_\delta(\sigma(c_t), \sigma(c_s)) + L_\delta(\rho(c_t), \rho(c_s)) \quad (9)$$

where b is number of batch size, and m is the length of output embedding from MLP_{out} or $Classifier_{out}$. The $\sigma(x)$ $\rho(x)$ is getting the mean value in the batch size dimension and embedding dimension respectively. And c_t, c_s is denoted correlations of teacher and student, the subscript t, s is denoted teacher and student. Meanwhile, our **SCKD** loss is designed as follows:

$$L_{sckd}(t, s) = L_{bm}(M(t), M(s)) + L_{bm}(C(t), C(s)) \quad (10)$$

where $M(t)$, $M(s)$, $C(t)$, $C(s)$ is denoted MLP_{out} and $Classifier_{out}$, from teacher and student respectively, is equivalent to c_t, c_s in Equation 9. The equation of final loss function is as below:

$$L_{final} = \alpha * L_{ce}(y, s) + \beta * L_{kd}(t, s) + \gamma * L_{sckd}(t, s) \quad (11)$$

The y, s of L_{ce} represent the ground truth and the final outputs of the student, the t, s of L_{kd} represent the logits from the teacher and student, and the t, s of L_{sckd} represent the collection of outputs from each stage in the teacher model and the student model respectively, excluding the outputs of the final linear layer. In addition, L_{ce} is the cross-entropy loss for classification task, L_{kd} is the classic knowledge distillation loss function.

3. EXPERIMENTAL RESULTS

We perform experiments on CIFAR100 [16] and CIFAR10 [16] datasets and compare with other methods. We use the pre-trained resnet56 [20], resnet110 [20], resnet32x4 [17], wrn-40-2 [21], vgg13 [18] as our teacher models, the pre-trained teacher models are publicly available from CRD [15]. To ensure the objectivity and fairness of the experimental results, the hyperparameters of all our experiments are same with CRD [15], excluding the weight α, β, γ in Equation 11. Due to the width of the experimental data table, we can only display all experimental results on the penultimate page in a uniform manner.

3.1. Same Architecture

Experiments with the same architecture on the CIFAR100 and CIFAR10 datasets, we use resnet20 [20], resnet32 [20], resnet8x4 [17], wrn-16-2 [21], vgg8 [18] as our student model. In the specific training, we set the hyperparameter $\alpha = 1, \beta = 5, \gamma = 25$. The results are shown in Table 1 and Table 2, respectively.

3.2. Diffrent Architecture

Since our method is based on model architecture, we need to reduce the percentages of hyperparameter γ in Equation 11 when training different architectures, which can alleviate the problem of insufficient learning and unstable convergence due to excessive gap between different architectures. So, we set the hyperparameter $\alpha = 1, \beta = 5, \gamma = 20$. The results are shown in Table 3.

4. CONCLUSION

In this paper, we propose a novel knowledge distillation method named Stage-Correlation Knowledge Distillation (**SCKD**), which considers not only logits information but also the correlation between stages when transferring knowledge. To better capture correlation, we train the MLP and use the classifier of the teacher. Extensive experiments on multiple models show that the proposed approach can significantly improve the performance of the student. Finally, we will continue to perfect our idea and hope to nourish our idea to other domains.

Table 1. Top-1 test accuracy (%) of on the **CIFAR100** for same architecture. In this experiment, we choose the best-performing CRD [15] among other methods to compare with **SCKD (Ours)**, and our method outperforms all other methods. The symbol (↑) denotes that the result has been improved, the symbol (↑) denotes that the result has been significantly improved, and the symbol (-) denotes that the result is similar. All results are the average over 3 trials.

Distillation	Teacher Acc	Resnet56 72.34	Resnet110 74.31	Resnet32x4 79.42	WRN40-2 75.61	WRN40-2 75.61	VGG13 74.64
Manner	Student Acc	Resnet20 69.06	Resnet32 71.14	Resnet8x4 72.50	WRN16-2 73.26	WRN40-1 71.98	VGG8 70.36
Logits	KD [11]	70.66	73.08	73.33	74.92	71.98	70.36
Single Layer	FitNet [8]	69.21	71.06	73.50	73.58	72.24	71.02
	PKT [22]	70.34	72.61	73.64	74.54	73.54	72.88
	RKD [23]	69.61	71.82	71.90	73.35	72.22	71.48
	CRD [15]	71.16	73.48	75.51	75.48	74.14	73.94
Mutiple Layer	AT [10]	70.55	72.31	73.44	74.08	72.77	71.43
	VID [12]	70.38	72.61	73.09	74.11	73.30	71.23
	FSP [9]	69.95	71.89	72.62	72.91	n/a	70.23
Mutiple Stages	**SCKD**	**71.38** (↑)	**73.56** (↑)	**75.81** (↑)	**75.64** (↑)	**74.15** (-)	**74.17** (↑)

Table 2. Top-1 test accuracy (%) of on the **CIFAR10** for same architecture. In this experiment, we choose the best-performing KD [11] among other methods to compare with **SCKD (Ours)**. In addition, we found from the experiments that the performance of CRD [15] drops sharply in small datasets with insufficient sample diversity. At the same time, our method still performs well, demonstrating the effectiveness and robustness of our method. All results are the average over 3 trials.

Distillation	Teacher Acc	Resnet32x4 95.52	WRN40-2 94.73	WRN40-2 94.73	VGG13 94.05	ResNet34 95.47
Manner	Student Acc	ResNet8x4 92.61	WRN16-2 93.83	WRN40-1 93.66	VGG8 91.95	ResNet18 95.13
Logits	**KD** [11]	93.80	94.66	94.07	92.95	95.36
Single Layer	CRD [15]	88.89 (↓)	88.41 (↓)	87.52 (↓)	84.75 (↓)	89.56 (↓)
Multiple Layers	VID [12]	93.10	94.02	93.81	92.49	95.26
Multiple Stages	**SCKD**	94.62 (↑)	94.83 (↑)	94.11 (-)	92.96 (-)	95.38 (↑)

Table 3. Top-1 test accuracy (%) of on the **CIFAR100** for diffrent architecture. All results are the average over 3 trials.

Distillation	Teacher Acc	ResNet32x4 79.42	WRN40-2 75.61	VGG13 74.64	ResNet50 79.34	ResNet32x4 79.42
Manner	Student Acc	ShuffleNetV1 70.50	ShuffleNetV1 70.50	MobileNetV2 64.6	MobileNetV2 64.6	ShuffleNetV2 71.82
Logits	KD [11]	74.07	74.83	67.37	67.35	74.45
Single Layer	FitNet [8]	73.59	73.73	64.14	63.16	73.54
	PKT [22]	74.10	73.89	67.13	66.52	74.69
	RKD [23]	72.28	72.21	64.52	64.43	73.21
	CRD [15]	75.11	76.05	69.73	69.11	75.65
Multiple Layers	AT [10]	71.73	73.32	59.40	58.58	72.73
	VID [12]	73.38	73.61	65.56	67.57	73.40
Multiple Stages	**SCKD**	75.26 (↑)	75.86 (↓)	69.68 (-)	69.37 (↑)	76.13 (↑)

5. REFERENCES

[1] François Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[2] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[3] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[4] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[5] Jonathan Frankle and Michael Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[6] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[8] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[9] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[10] Sergey Zagoruyko and Nikos Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[11] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[12] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai, "Variational information distillation for knowledge transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171.

[13] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 3779–3787.

[14] Frederick Tung and Greg Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1365–1374.

[15] Yonglong Tian, Dilip Krishnan, and Phillip Isola, "Contrastive representation distillation," *arXiv preprint arXiv:1910.10699*, 2019.

[16] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[18] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[22] Nikolaos Passalis and Anastasios Tefas, "Probabilistic knowledge transfer for deep representation learning," *CoRR, abs/1803.10837*, vol. 1, no. 2, pp. 5, 2018.

[23] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.