Supplementary Material: Knowledge Distillation Meets Self-Supervision

Guodong Xu[1[0000−0002−7026−1375]], Ziwei Liu[1[0000−0002−4220−5958]], Xiaoxiao Li[1[0000−0002−0376−5135]], and Chen Change Loy[2[0000−0001−5345−1591]]

[1] The Chinese University of Hong Kong
{xg018, zwliu, lx015}@ie.cuhk.edu.hk
[2] Nanyang Technological University
ccloy@ntu.edu.sg

1 Network Architectures

We adopt resnet [6], ResNet [6], WideResNet [18], MobileNet [9], vgg [15] and ShuffleNet [16,20] as the network backbones. For resnet, We use resnet-d to represent CIFAR-style resnet with three groups of basic blocks, each with 16, 32 and 64 channels, respectively. resnet8×4 and resnet32×4 indicate a 4× wider network (with 64, 128 and 256 channels for each block). For ResNet, ResNet-d represents ImageNet-style ResNet with Bottleneck blocks and more channels. For WideResNet (wrn), wrn-d-w represents wide ResNet with depth d and width factor w. For MobileNet, following [17], we use a width multiplier of 0.5. For vgg, ShuffleNetV1 and ShuffleNetV2, we adapt their architectures to CIFAR100 [12] dataset from their original ImageNet [4] counterparts.

2 Data Processing

2.1 Data Augmentation of Normal Images

CIFAR100. We pad the original image with size 4 and randomly crop it into 32×32. We apply horizontal flip with a probability of 0.5 and normalize three channels with the mean and standard deviation derived from the whole dataset.
ImageNet. The images are randomly cropped, into sizes ranging from 0.08 to 1.0 of the original image size and into aspect ratios ranging from 3/4 to 4/3 of the original image aspect ratio. The cropped patch is resized to 224×224 and flipped with a probability of 0.5. Finally, it is normalized by the mean and standard deviation in a channel-wise manner.

2.2 Image Transformations in Self-Supervision

In SSKD, the input images contain both normal images and transformed version. We select four kinds of transformations to construct the transformation pool, namely: 1) color dropping, 2) rotation ($\pm 90°, 180°$), 3) cropping + resizing, 4) color jitter.

Color dropping converts an RGB image to a gray image through $L = 0.229R + 0.587G + 0.114B$, where L is the gray value. Rotation is specifically performed at three angles, i.e., $\pm 90°$ and $180°$, because rotation at these angles do not introduce artifacts (black regions at the image edge). Cropping + resizing first randomly crops a patch from the augmented image with size 0.08 to 1.0 and aspect ratio 3/4 to 4/3 and then resizes it to a fixed resolution (32×32 in CIFAR100 and 224×224 in ImageNet). For color jitter, we jitter the brightness, contrast and saturation with a factor sampled from 0.6 to 1.4 and jitter the hue with a factor sampled from -0.1 to 0.1.

3 Implementations of different self-supervision tasks

In ablation study, we investigate the effects of different self-supervision tasks, i.e., Contrastive [2], Exemplar [5], Jigsaw [13] and Rotation [11]. The implementation details of Contrastive are introduced in the main paper. Here we introduce the details when combining other three self-supervision tasks with knowledge distillation.

Exemplar. Exemplar treats each instance of the dataset as a separate class. It applies heavy image transformations to the original image and forces the network to classify transformed image correctly. When combining it with KD, we adopt the same transformations as discussed in Sec A2.2. The SS module is a classifier with class number equalling to the dataset size. In student's training, we transfer the logits of this classifier from teacher to student.

Jigsaw. Jigsaw splits the original image into several non-overlapping patches and permutes them. It re-organizes these patches into image format and forces the network to recognize the permutation pattern. When combining it with KD, we split each image into $2 \times 2 = 4$ patches. Thus, the SS module is a classifier with 24 ($4! = 24$) classes. In student's training, we transfer the logits of this classifier.

Rotation. Rotation first rotates the image with four angles, i.e., $0°, \pm 90°, 180°$ and then forces the network to classify the rotation angle correctly. When combining it with KD, the SS module is a 4-way classifier. In student's training, we transfer the logits of this classifier from teacher to student.

4 Training Hyperparameters

4.1 Hyperparameters in Competing Methods

The losses of all competing methods can be uniformly written as:

$$L = \alpha_1 L_{ce} + \alpha_2 L_{kd} + \alpha_3 L_{distill}, \tag{1}$$

where L_{ce}, L_{kd} and $L_{distill}$ are cross-entropy loss, conventional KD loss and specially designed loss in each method, respectively.

CIFAR100 We re-run all the competing methods using the implementation of CRD [17]. We set $\alpha_1 = 0.1$, $\alpha_2 = 0.9$. For α_3 of different methods, we use the

Table 1. Linear Classification Accuracy (%) on STL10 and TinyImageNet. We use wrn40-2 and ShuffleNetV1 as teacher and student networks, respectively. The competing methods include KD [8], FitNet [14], AT [19], FT [10], and CRD [17]

	Student	Teacher	KD	FitNet	AT	FT	CRD	Ours
CIFAR100→STL10	71.58	71.01	73.25	73.77	73.47	73.56	74.44	**74.74**
CIFAR100→TinyImageNet	32.43	27.74	32.05	33.28	33.75	33.69	34.30	**34.54**

values reported in CRD. Note that CRD sets the α_2 in all methods to be 0, so the results of competing methods in our experiments are better than those in CRD.

ImageNet We do not run the competing methods, but copy their results from CRD directly.

4.2 Hyperparameters in SSKD

CIFAR100 We set the temperature τ in L_{kd} and L_T to be 4 and τ in L_{ss} to be 0.5. For student training, we set $\lambda_1 = 0.1, \lambda_2 = 0.9, \lambda_3 = 3.0, \lambda_4 = 5.0$ in Eq 8. We train all the models for 240 epochs. The initial learning rate is 0.05 and is decayed by a factor of 10, respectively, at 150, 180 and 210 epochs. We run experiments on a TITAN-X-Pascal GPU with a batch size of 64. An SGD optimizer with a 5×10^{-4} weight decay and 0.9 momentum is adopted.

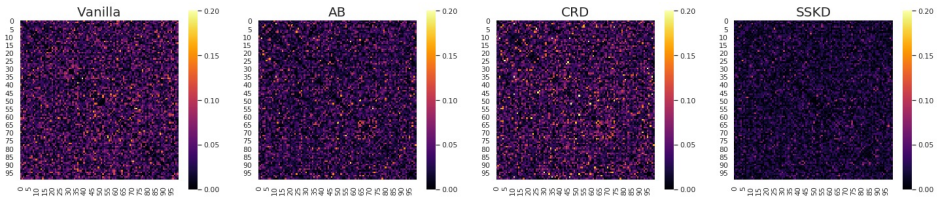
ImageNet We use the same temperatures and loss weights as those in CIFAR100 experiments, except that λ_1 is set to 1.0. We train all the models for 100 epochs. The initial learning rate is 0.1 and is decayed by 10, respectively, at 30, 60 and 90 epochs. We train models with eight parallel GPUs with a total batch size of 256. The optimizer parameters are the same as those in CIFAR100 experiments.

5 Evaluation of Learned Representations

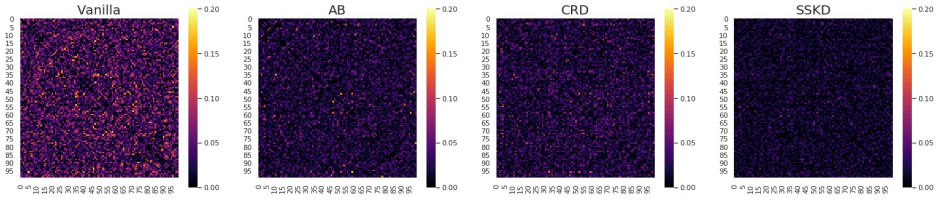
Following CRD [17], we also investigate the qualities of student representations. A good feature extractor should generate linear separable features. Hence, we use the fixed backbone of student (trained on CIFAR100) to extract features of STL10 [3] and TinyImageNet [1], and then train a linear classifier. We select wrn40-2 and ShuffleNetV1 as teacher and student networks, respectively. As shown in Table 1, SSKD achieves the highest accuracies on both STL10 and TinyImageNet.

6 Visualization of Correlation Difference

We compute the difference (L1 error) between the correlation matrices of the teacher's and student's classifier weights and visualize the difference through heatmap (Fig. 1). We select four methods: vanilla student without distillation



(a) Teacher: ResNet50, Student: MobileNetV2



(b) Teacher: resnet32×4, Student: ShuffleNetV2

Fig. 1. The difference between correlation matrices of teacher's and student's classifier weights (of CIFAR100). The correlation matrices are computed using normalized weights. We select two teacher-student pairs and four methods: Vanilla student, AB [7], CRD [17] and our SSKD. SSKD achieves significant matching between student's and teacher's correlations, indicating that it captures the teacher's correlation structures best

and students trained by AB [7], CRD [17] and our SSKD. It is clear that SSKD obtains the smallest difference on both two teacher-student pairs, indicating that SSKD captures the teacher's correlation structure best.

References

1. http://tiny-imagenet.herokuapp.com/ 3
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020) 2
3. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. vol. 15, pp. 215–223 (2011) 3
4. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009) 1
5. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. arXiv preprint arXiv:1406.6909 (2014) 2
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016) 1
7. Heo, B., Lee, M., Yun, S., Choi, J.Y.: Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In: AAAI. pp. 3779–3787 (2019) 4
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015) 3
9. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 1
10. Kim, J., Park, S., Kwak, N.: Paraphrasing complex network: Network compression via factor transfer. In: Advances in Neural Information Processing Systems. pp. 2760–2769 (2018) 3
11. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 2
12. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009) 1
13. Noroozi, M., Favaro, P.: Unsupervised learning of visual representions by solving jigsaw puzzles. In: The European Conference on Computer Vision (2016) 2
14. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014) 3
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 1
16. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 1
17. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. In: International Conference on Learning Representations (2020) 1, 2, 3, 4
18. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016) 1
19. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: International Conference on Learning Representations (2017) 3
20. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 1