Distilling Knowledge via Knowledge Review
Supplementary File

Pengguang Chen[1], Shu Liu[2], Hengshuang Zhao[3], Jiaya Jia[1]
The Chinese University of Hong Kong[1] SmartMore[2] University of Oxford[3]
{pgchen, leojia}@cse.cuhk.edu.hk liushuhust@gmail.com hengshuang.zhao@eng.ox.ac.uk

1. Detailed Network Architecture and Training Hyper-parameters

For a fair comparison, we choose the network architecture following recent works [1, 4]. We use ResNet20, ResNet32, ResNet56, and ResNet110 to represent CIFAR-style ResNet, which contains three groups of basic blocks with the channels of 16, 32, and 64. Specially, we refer to ResNet8×4 and ResNet32×4 as networks with ×4 channels, which are 64, 128, and 256 respectively. ResNet18, ResNet34, and ResNet50 are the standard ResNet architecture provided by the PyTorch. WRN-d-r represents the wide resnet the with depth d and width factor r. We remove some downsample layers of MobileNet and ShuffleNet series for the CIFAR dataset, which are the same as previous work [4]. We use the widen factor of 0.5 for MobileNetV2 on CIFAR dataset, and 1.0 for others. For VGG series networks, we use the original network with batchnorm.

For the CIFAR-100 dataset, we train our models on one GPU with a batchsize of 128. The learning rate for ResNet, WRN, and VGG is initialized as 0.1. The learning rate for MobileNet and ShuffleNet is initialized as 0.02. We train models for 240 epochs, the learning rate is decayed by 10 at 150, 180 and 210 epochs. The weight decay is 5e-4. The λ is empirically set to 5.0 for our models. For the ImageNet dataset, we train our models on 4 GPUs with a batchsize of 256. The learning rate is initialized as 0.1. The models are trained for 100 epochs. The learning rate is decayed by 10 at 30, 60 and 90 epochs. The weight decay is 1e-4. The λ is empirically set to 1.0.

For the object detection task, we train all models on 4 GPUs. The models are trained for 180,000 iterations with a batch size of 8. The learning rate is initialized as 0.01 and decay by 10 at 120,000 and 160,000 iterations. For the KD [2] methods, we empirically adjust the loss weight to 20.0 for better performance. For the FitNet [3], we empirically adjust the loss weight to 0.1. For FGFI, we following the author's official code and set the loss weight to 0.01. We set λ to 0.5.

2. Implementation Details for Object Detection

A detector can be divided into two parts roughly, the backbone and the headers, just like the neural network for classification can be divided into the feature extractor and the classifier. Our method is used on the feature extractor's intermediate features for the classification task. On object detection, the process is similar, we can consider the backbone as the feature extractor, and apply our method on the backbone's intermediate features. There is only one difference, modern detectors will use the intermediate features for detection, but this will not affect the process of our knowledge distillation method. The application on instance segmentation is similar to object detection.

For KD [2] on detection, we can directly use the KL divergence between the teacher's and the student's RoI classification logits, just like on the classification tasks. However, there is a problem, the proposal for the teacher and the student is different, so the meanings of the logits is different, and directly distill between them is meaningless. To address this problem, we simply change the process of the teacher detector. Instead of the teacher detector's proposal, we use the student detector's proposal as the input for the teacher's RoI head to align the meaning of RoI classification logits. In that case, the original KD algorithm can successfully improve the student detector's performance, although the improvement is limited. We conjecture the reason is the detector has many losses to optimize the whole network, but KD only guide the classification branch.

For FitNet [3] on detection, the process is straightforward, we directly use FitNet on the backbone's second stage features just like in classification tasks. The performance of FitNet is better than KD but still not significant.

FIFG [5] is a knowledge distillation method directly designed for object detection, it utilizes the character of detection tasks and transfers the knowledge between features through fine-grained regions. The original paper uses a primitive baseline, so we transfer the official code to the new strong baseline. FIFG performs better than previous methods. Our method is not

designed for detection, but still outperforms FIFG by a large margin, which demonstrates the applicability and superiority of our method.

References

[1] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *ICCV*, 2019. 1

[2] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. 1

[3] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 1

[4] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 1

[5] Tao Wang, Li Yuan, Xiaopeng Zhang, and Jiashi Feng. Distilling object detectors with fine-grained feature imitation. In *CVPR*, 2019. 1