Outline

In this supplementary file, we first provide more results and discussions in Sec. A: results of MobileNetV2 as student in Sec. A.1, detailed proof of Theorem 1 in Sec. A.2, comparison of previous feature-based KD methods in Sec. A.3, an example of local transformations breaking the original relative magnitude in Sec. A.4, and discussion on computation cost in Sec. A.5. Further, we offer the elaborated implementation details for the KDEP setups and downstream task setups in Sec. B.

A. More Results

A.1. Main Results: MobileNetV2 as Student.

Due to the length limit of the main paper, we show the results of MobileNetV2 as student in Table S.1. Similar results have been achieved with MobileNetV2 (MNV2) as student compared to R18 as student, which shows the generalization of the proposed KDEP method.

A.2. Detailed Proof of Theorem 1

Given two independent random variables with normal distribution $T \sim N(0, \sigma^2)$ and $S \sim N(0, \sigma_s^2)$, then $F(\sigma) = \mathbb{E}[(T-S)^2]$ is monotonically increasing ($\sigma > 0$).

Proof 1 *(Detailed version)*

$$
\begin{aligned}
F(\sigma) &= \mathbb{E}[(T-S)^2] \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(t-s)^2 \cdot P(t,s)\mathrm{d}t\mathrm{d}s \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(t-s)^2 \cdot P(t)\cdot P(s)\mathrm{d}t\mathrm{d}s \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(t-s)^2 \cdot \frac{1}{2\pi\sigma\sigma_s}e^{\frac{-s^2}{2\sigma_s^2}}e^{\frac{-t^2}{2\sigma^2}}\mathrm{d}t\mathrm{d}s \\
&= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{+\infty}e^{\frac{-t^2}{2\sigma^2}}\Big(\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma_s}(t-s)^2\cdot e^{\frac{-s^2}{2\sigma_s^2}}\mathrm{d}s\Big)\mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{+\infty}e^{\frac{-t^2}{2\sigma^2}}(t^2+\sigma_s^2)\mathrm{d}t = \sigma^2+\sigma_s^2
\end{aligned}
$$

$$\frac{\mathrm{d}F(\sigma)}{\mathrm{d}\sigma} = 2\sigma > 0 \Rightarrow \textit{monotonically increasing}$$

A.3. Compare other feature-based KD methods.

Here, we show the KDEP results of some traditional feature-based KD methods that are developed for distilling knowledge to improve student's performance for a specific task instead of its transferability. Also, these methods all require task label loss which violates our setting of an unlabeled dataset. Hence, we don't include them in our paper to compare for fairness. As shown in Table S.2, previous feature-based KD methods largely rely on logit KD loss and task label loss, and perform inferiorly with only feature-level clues.

Method	Data	Epoch	Time (/h)	Classification (Acc %) Caltech	DTD	CUB	CIFAR	Avg
rand. init.	-	-	-	51.77	57.34	60.44	76.66	61.55
SP. b.	10%	90	4.2	66.58	65.72	71.09	78.60	70.50
KDEP	10%	90	4.3	74.34	71.84	74.24	81.06	75.37
SP. b.	100%	9	4.2	68.09	67.514	73.33	78.95	71.97
KDEP	100%	9	4.3	74.48	72.51	74.76	81.47	75.81
SP. b.	10%	180	8.4	69.23	67.33	73.62	79.50	72.42
KDEP	10%	180	8.6	75.56	73.29	75.28	81.98	**76.53**
SP. b.	100%	18	8.4	71.83	69.60	74.64	80.13	74.05
KDEP	100%	18	8.6	76.06	73.14	76.00	82.15	**76.83**
SP. b.	10%	900	42	69.93	67.59	72.74	79.83	72.52
KDEP	10%	900	43	77.66	73.05	76.06	82.53	**77.32**
SP. o.	100%	90	42	76.43	72.26	76.34	81.91	76.74
KDEP	100%	90	43	78.57	73.94	76.40	82.73	**77.91**

Method	Data	Epoch	Time (/h)	Segmentation (mIoU %) Cityscapes	VOC12	ADE20K	Avg
rand. init.	-	-	-	40.33	39.23	23.07	34.21
SP. b.	10%	90	4.2	60.92	62.60	29.17	50.89
KDEP	10%	90	4.3	63.50	67.28	31.46	54.08
SP. b.	100%	9	4.2	61.42	64.31	29.61	51.78
KDEP	100%	9	4.3	63.53	68.17	32.00	54.57
SP. b.	10%	180	8.4	61.68	64.65	29.95	52.09
KDEP	10%	180	8.6	64.32	68.73	31.92	**54.99**
SP. b.	100%	18	8.4	62.23	65.82	29.55	52.53
KDEP	100%	18	8.6	64.23	69.32	32.41	**55.32**
SP. b.	10%	900	42	61.87	64.95	31.07	52.63
KDEP	10%	900	43	63.89	70.45	32.23	**55.52**
SP. o.	100%	90	42	64.16	69.48	31.69	55.11
KDEP	100%	90	43	64.72	71.07	32.39	**56.06**

Method	Data	Epoch	Time (/h)	Detection VOC0712 AP	VOC0712 AP_{50}	VOC0712 AP_{75}	COCO AP	COCO AP_{50}	COCO AP_{75}
rand. init.	-	-	-	31.4	56.7	30.2	25.8	43.6	26.9
SP. b.	10%	90	4.2	42.5	71.3	44.0	27.7	46.4	29.0
KDEP	10%	90	4.3	**46.7**	75.7	49.0	**29.7**	48.8	31.2
SP. b.	100%	9	4.2	43.0	71.9	44.0	27.9	46.4	29.0
KDEP	100%	9	4.3	**47.1**	76.3	49.6	**30.2**	49.8	31.9
SP. b.	10%	180	8.4	43.0	71.4	45.0	27.6	46.2	29.0
KDEP	10%	180	8.6	**47.0**	75.9	49.9	**30.0**	49.4	31.5
SP. b.	100%	18	8.4	43.7	72.6	45.4	27.9	46.6	29.3
KDEP	100%	18	8.6	**47.2**	76.3	50.0	**30.4**	50.1	32.0
SP. b.	10%	900	42	44.0	73.1	45.6	28.7	47.9	30.1
KDEP	10%	900	43	**47.0**	76.0	49.8	**29.7**	49.2	31.4
SP. o.	100%	90	42	45.5	75.0	47.6	29.6	49.1	31.3
KDEP	100%	90	43	**46.8**	76.5	49.4	**30.0**	49.6	31.4

Table S.1. **KDEP *vs.* SP, R50 $\rightarrow$ MNV2, fine-tuned on various tasks.** KDEP refers to our SVD+PTS method.

Method	SP	FitNet [11]	AT [7]	NST [6]	AB [5]	Heo [4]	Ours
Acc	71.05	72.43	67.84	67.05	71.66	72.98	75.74

Table S.2. Compare feature-based KD with 10% data and 90 epochs. Acc: averaged top-1 accuracy over 4 classification tasks.

A.4. Example: Breaking Relative Magnitude.

In Sec. 3.3 of our paper, we argue that Scale Normalization (SN) and Std Matching (SM) are local transformations that transform channel-wisely, which may break the original relative magnitude between channels. Here, we provide a toy example as an illustration.

For instance, we have a three channel penultimate layer with target Std=[4, 3, 2] and after SVD Std=[50, 5, 1]. For a feature after SVD that is [10, 2, 2], with SN we have [0.2, 0.4, 2], and with SM we have [0.8, 1.2, 4], both losing the original relative magnitude.

A.5. Computation cost.

In most of our experimental results, we provide the training time of KDEP and supervised pre-training, where only unnoticeable extra training time is added. Moreover, the GPU memory usage during KDEP is also similar to supervised pre-training since we do not require gradients for the teacher. Yet, our teacher model is R50, and additional computation costs may increase when using larger teachers that inquires more inference time and GPU memory. Still, the pre-training time could be largely reduced compared with supervised pre-training.

B. Implementation Details

We implement our method using the PyTorch [9] framework and use SGD with momentum of 0.9 for all our experiments. All experiments are conducted using four 32G V100 GPUs.

B.1. KDEP Setups

For the KDEP procedure, we use an initial learning rate (lr) of 0.3 for R50→R18 and 0.1 for R50→MNv2. Batch size is set to 512. For data augmentation, we use RandomResizedCrop(224) and RandomHorizontalFlip. In order to reduce the burden of hyper-parameter tuning (*e.g.* weight decay), we multiply our feature-based loss (refer to Eq. 1 in our paper) by a loss weight w, which matches the feature-based loss to the loss scale of supervised pretraining. For reproduction, we provide the loss weight of different teacher-student pairs in Table S.3.

Teacher→Student	w
Standard SP R50→R18	20
MS R50→R18	3
SWSL R50→R18	1
MEAL V2 R50→R18	1
Swin-B→R18	3
MS R50→MNV2	3

Table S.3. Value of loss weight w for different T-S pairs.

For the 10% ImageNet data setting, we sample 10% images from each class of the original 1000 class in ImageNet-1K. We set the training epochs to 90 or 180, and drop the lr by a factor of 10 at 1/3 and 2/3 of total epochs. When using all of ImageNet data, we train for 9 or 18 epochs to verify fast convergence, and for 90 epochs to further boost performance, where 90 epochs with all ImageNet data is the standard supervised pretraining schedule [12]. We use weight decay $\in \{1e\text{-}4, 4e\text{-}4, 5e\text{-}4\}$ according to the length of the training schedule, shown in Table S.4.

Data	Epoch	Weight decay
10%	90	$5e$-4
10%	180	$4e$-4
100%	9	$5e$-4
100%	18	$4e$-4
100%	90	$1e$-4

Table S.4. Weight decay for different KDEP training scedules.

B.2. Downstream Task Setups

For all downstream tasks, we use the same schedule and evaluation protocols for all models for a fair comparison.

For image classification, we initialize the backbone with the distilled weights and add a linear classifier with random initialization. We train the network for 150 epochs with batch size of 64, weight decay of $5e$-4, an initial learning rate $\in \{0.01, 0.001\}$ which drops by a factor of 10 at 1/3 and 2/3 of total epochs. Again, we use RandomResizedCrop(224) and RandomHorizontalFlip for data augmentation.

For semantic segmentation, we also initialize the backbone with the distilled weights, and add a PSP module [13] and a segmentation head after the backbone. We use batch size of 16, an initial learning rate of 0.01, weight decay of $1e$-4, crop size of 512, and deploy an polynomial learning rate annealing procedure [1]. For data augmentation, we use random scaling, random horizontal flipping, random rotation, and random Gaussian blur. The number of epochs is 50, 100, 200 for VOC12, ADE20K, and Cityscapes, respectively, following previous standard [13].

For object detection, we experiment with the Faster R-CNN [10] detector and backbones are also initialized by the distilled weights. Unless noted, all the setups follow the evaluation protocols in MOCO [2]. For ResNet18, we use a backbone of R18-C4 (similar to R50-C4 [3]) for both VOC and COCO experiments. For MobileNetV2, we equip it with a FPN [8] backbone. 1x schedule is applied for COCO.

B.3. Parametric/Non-parametric Methods.

In our experiments, we experiment with three 1×1 conv variants for parametric methods: Post-ReLU, Pre-ReLU and the one in [4]. Specifically, we add a 1×1 convolutional layer and a Batch Normalization layer either after (Post-ReLU) or before (Pre-ReLU) the ReLU activation function. We also experiment with the Pre-ReLU 1×1 conv method equipped with Margin ReLU of teacher's feature and Partial L_2 loss function as in [4].

For non-parametric methods, we explore channel selection (CS.var, CS.rand), interpolation, and SVD. For channel selection methods, we experiment with two methods: selecting the top-D_s channels with largest variances (CS.var) or random selecting D_s channels (CS.rand). For interpolation method, we use the default nearest-neighbor interpolation in PyTorch [9]. For SVD, we calculate the singular vectors offline and use the top-D_s principal components to transform the teacher's features during the online KDEP process.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

[2] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

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

[5] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

[6] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv:1707.01219*, 2017.

[7] Nikos Komodakis and Sergey Zagoruyko. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[9] 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. *NIPS*, 2019.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

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

[12] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *arXiv:2007.08489*, 2020.

[13] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.