Supplementary Material for *"Knowledge Distillation Layer that Lets the Student Decide"*

1 Extended Empirical Study

In this section, we provide comparisons of our method to additional KD methods, along with the evaluation of our method on Tiny-ImageNet. We also provide more ablations to offer deeper insights into the efficacy of our method.

1.1 Results on More Datasets and Methods

In this section, we provide the extended results in Tabs. 1 to 3 for the evaluations on Tiny-ImageNet [13], ImageNet [4] and CIFAR-100 [12], respectively. We compare our method against KD [9], FitNet [16], AT [25], AB [8], FSP [23], SP [20], VID [1], CRD [19], DKD [27], SimKD [3], TDD [18] and QUEST [11]. Overall, the results demonstrate the effectiveness of our methods letKD-2 and letKD-1 for all the datasets by being the first and the second best compared to other KD alternatives except with the SimKD method in RN32x4-RN8x4 on CIFAR-100 for which the relevant discussion is provided in the main paper.

Table 1: Average top-1 accuracies on CIFAR-100 over 5 trials. **Bold**: best in its category.

Archs. →	Homogeneous						Heterogeneous			
Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	RN56 RN20	RN110 RN20	RN110 RN32	RN32x4 RN8x4	WRN-40-2 SNV1	RN32x4 SNV1	RN32x4 SNV2	RN50 MNV2
Methods ↓	75.61 73.26	75.61 71.98	72.34 69.06	74.31 69.06	74.31 71.14	79.42 72.50	75.61 70.50	79.42 70.50	79.42 71.82	79.34 64.60
KD	74.92	73.54	70.66	70.67	73.08	73.33	74.83	74.07	74.45	67.35
FitNet	73.58	72.24	69.21	68.99	71.06	73.50	73.73	73.59	73.54	63.16
AT	74.08	72.77	70.55	70.22	72.31	73.44	73.32	71.73	72.73	58.58
AB	72.50	72.38	69.47	69.53	70.98	73.17	73.34	73.55	74.31	67.20
FSP	72.91	-	69.95	70.11	71.89	72.62	-	-	-	-
SP	73.83	72.43	69.67	70.04	72.69	72.94	74.52	73.48	74.56	68.08
VID	74.11	73.30	70.38	70.16	72.61	73.09	73.61	73.38	73.40	67.57
CRD	75.48	74.14	71.16	71.46	73.48	75.51	76.05	75.11	75.65	69.11
CRD +KD	75.64	74.38	71.63	71.56	73.75	75.46	76.27	75.12	76.05	69.54
DKD	76.24	74.81	71.97	-	74.11	76.32	76.70	76.45	77.07	70.35
SimKD	76.06	74.92	68.95	69.35	72.15	**78.08**	76.95	77.18	77.78	68.91
TDD	75.01	74.04	71.53	-	-	-	75.60	-	-	68.37
TDD +CRD	75.71	74.35	71.88	-	-	-	76.34	-	-	69.22
QUEST	76.10	74.58	71.84	71.89	74.08	75.88	76.75	76.28	77.09	69.81
letKD-1	76.29 ∓0.15	75.01 ∓0.09	72.44 ∓0.24	72.68 ∓0.31	74.40 ∓0.14	76.70 ∓0.06	76.93 ∓0.16	76.65 ∓0.24	77.75 ∓0.17	69.97 ∓0.18
letKD-2	**76.56 ∓0.22**	**75.19 ∓0.13**	**73.27 ∓0.16**	**73.38 ∓0.14**	**74.62 ∓0.20**	**77.09 ∓0.18**	**77.08 ∓0.12**	**77.30 ∓0.12**	**77.95 ∓0.06**	**70.39 ∓0.23**

Table 2: Top-1 and top-5 accuracies on ImageNet. Setting **(a)**: Teacher and student models are selected as RN34-RN18. Setting **(b)**: Teacher and student models are selected as RN50-MNV2. **Bold**: best in its category.

Setting		Teacher	Student	KD	AT+KD	DKD	QUEST	**letKD-1**	**letKD-2**
(a)	Top-1	73.31	69.75	70.66	70.70	71.70	71.67	72.33	**72.38**
	Top-5	91.42	89.07	89.88	90.00	90.41	90.67	91.06	**91.15**
(b)	Top-1	76.13	68.87	68.58	69.56	72.05	72.54	73.78	**73.98**
	Top-5	92.86	88.76	88.98	89.33	91.05	91.13	91.81	**92.00**

Table 3: Average top-1 accuracies on Tiny-ImageNet over 3 trials. **Bold**: best in its category.

Archs. →	Homogeneous			Heterogeneous	
Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	RN56 RN20	WRN-40-2 SNV1	RN50 MNV2
Methods ↓	61.26 57.17	61.26 56.25	58.34 52.66	61.26 60.52	68.97 58.35
KD	59.16	57.75	53.04	64.80	58.68
FitNet	57.75	-	51.73	-	57.55
AT	58.71	57.41	54.01	63.90	50.91
FSP	57.33	-	53.55	-	-
SP	55.69	53.74	54.03	64.62	58.11
VID	58.51	57.45	53.20	63.58	57.50
TDD	59.22	58.42	54.45	65.27	59.09
TDD +CRD	59.53	59.20	54.85	65.50	59.72
QUEST	59.86	59.13	54.53	65.23	59.81
letKD-1	61.42 $\mp$0.36	59.75 $\mp$0.50	55.54 $\mp$0.33	65.70 $\mp$0.42	60.69 $\mp$0.17
letKD-2	**62.21** **$\mp$0.17**	**60.59** **$\mp$0.27**	**57.35** **$\mp$0.46**	**66.15** **$\mp$0.50**	**61.15** **$\mp$0.46**

1.2 More Ablations

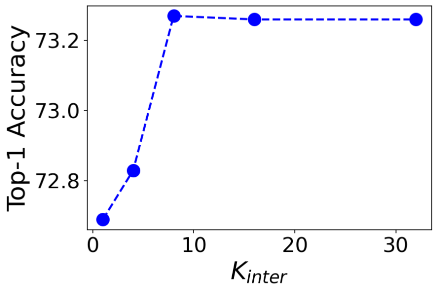

Figure 1: Effect of K_{inter}

Hyperparameters. Our KD layer introduces 4 additional hyperparameters which are $\{\alpha_{inter}, \alpha_{penult}\}$ from Fig. 2 in the main paper, K_{penult} the number of cluster centers used in the penultimate layer, and K_{inter} the number of sub-classes for each class used in the intermediate layer. We use $K_{penult} = 4096$ to be directly comparable with [ ]. For K_{inter}, through our analysis on CIFAR-100 with RN56-RN20, plotted in Fig. 1, we observe a relatively stable performance for $K_{inter} > 8$. Hence, we set $K_{inter} = 8$ for the rest of the experiments. Finally, for the selection of $\{\alpha_{inter}, \alpha_{penult}\}$, since we use normalized convolution with a learnable scale for the 1x1 to jointly learn it (§ 2.2), we set $(\alpha_{inter}, \alpha_{penult}) = (1, 1)$ in all models except for RN50-MNV2 in CIFAR-100 and Tiny-ImageNet. Essentially, in both letKD-1 and letKD-2 experiments, *(i)* for CIFAR-100, we arrange $\alpha_{penult} = 0.1$ and $\alpha_{inter} = 0.2$, *(ii)* for Tiny-ImageNet, we arrange $\alpha_{penult} = 0.5$ and $\alpha_{inter} = 1$.

Table 4: Effect of our KD layer on intermediate layer classification performance on CIFAR-100 with RN56-RN20

letKD-2 ($\alpha_{inter} = 0$)	letKD-2 ($\alpha_{inter} = 1$)	
	x	$\hat{x}$
52.71	51.71	56.36

Feature geometry. Towards the understanding of the impact of the proposed KD layer for intermediate layer (*i.e.*, lower level) supervision, we measure the classification capacity of the student trained with our methods in Tab. 4 with α_{inter} representing the inclusion of our layer. In this table, x and $\hat{x}$ represent the input and the output of the KD layer as in Fig. 2 in the main paper. To obtain the classification scores, we perform global average pooling to the extracted intermediate features of the trained student and fit a linear classifier using LDA. This analysis is required to see whether the student is able to exploit the teacher's supervision according to its own discrimination capacity. As seen from Tab. 4, the student attains 52.71 % accuracy when it is trying to imitate the teacher ($\alpha_{inter} = 0$). When our layer is included ($\alpha_{inter} = 1$), even though the input features x perform 1% poorer compared to $\alpha_{inter} = 0$, the output features $\hat{x}$ strongly show the advantage of letting the student freely exploit the teacher rather than directly forcing to imitate it.

Effect of the KD layer. To further validate exploiting the teacher's knowledge to shape the intermediate features, we analyze the effect of enhancing the student's features with the weighted combinations of the learned semantic vectors. Namely, we set $\alpha = 0$ in Fig. 2 in the main paper to lift the knowledge-based feature transform. With the RN56-RN20 pair, we evaluate all possible settings in CIFAR-100 and provide the results averaged over 5 trials in Tab. 5. We observe that $\alpha = 1$ consistently improves the performance whereas $\alpha = 0$ consistently degrades, especially in the intermediate layer. Based on these results, we can conclude that the student can effectively decide to exploit the semantically meaningful information coming from the teacher to shape the embedding space rather than directly imitating the feature space. The addition of our layer substantially improves the performance, especially when multi-layer supervision is involved. Nevertheless, it is important to quantitatively show the computational cost of the inclusion of our KD layer. The computation rises merely by adding 1x1-BN-ReLU-1x1 convolution block, resulting in about 2% longer sec per image in intermediate layers and 2%-20% in the penultimate layer, depending on the number of cluster centers, K, ranging from 64 to 4096. Therefore, our proposed method remains computationally feasible.

Table 5: Effect of the included parts in letKD-2

Inter.	α_{inter}	Penult.	α_{penult}	Top-1 Acc.
✓	0	-	0	70.64
✓	1	-	0	70.80
-	0	✓	0	71.84
-	0	✓	1	72.44
✓	0	✓	0	71.70
✓	0	✓	1	72.13
✓	1	✓	0	72.78
✓	1	✓	1	73.27

Source of effectiveness of the KD layer. To reduce the confounding of factors other than the proposed method, we examine whether the performance increase is coming from the method or the capacity increase introduced by our KD layer. Hence, in Tab. 6, we compare the performance of the three methods as FitNet, FitNet equipped with our KD layer at the penultimate layer with and without supervision. The experiments are done using CIFAR-100 with the teacher-student pair selected as RN110-RN32 and they are averaged over 5 trials. We trained FitNet using the stage-2 outputs and included our KD layer at the output of the penultimate layer. We highlight the inclusion of supervision (*i.e.*, distillation loss) using the notations "with" and "without". These results show that even though the capacity of the student is marginally increased due to our KD layer, the major contribution to the performance occurs upon combining it with our supervision.

Table 6: Effect of the impact of the KD layer on performance improvement considering the capacity increase

Methods	Top-1 Acc
FitNet	71.59
FitNet+KD layer without supervision	71.80
FitNet+KD layer with supervision	73.36

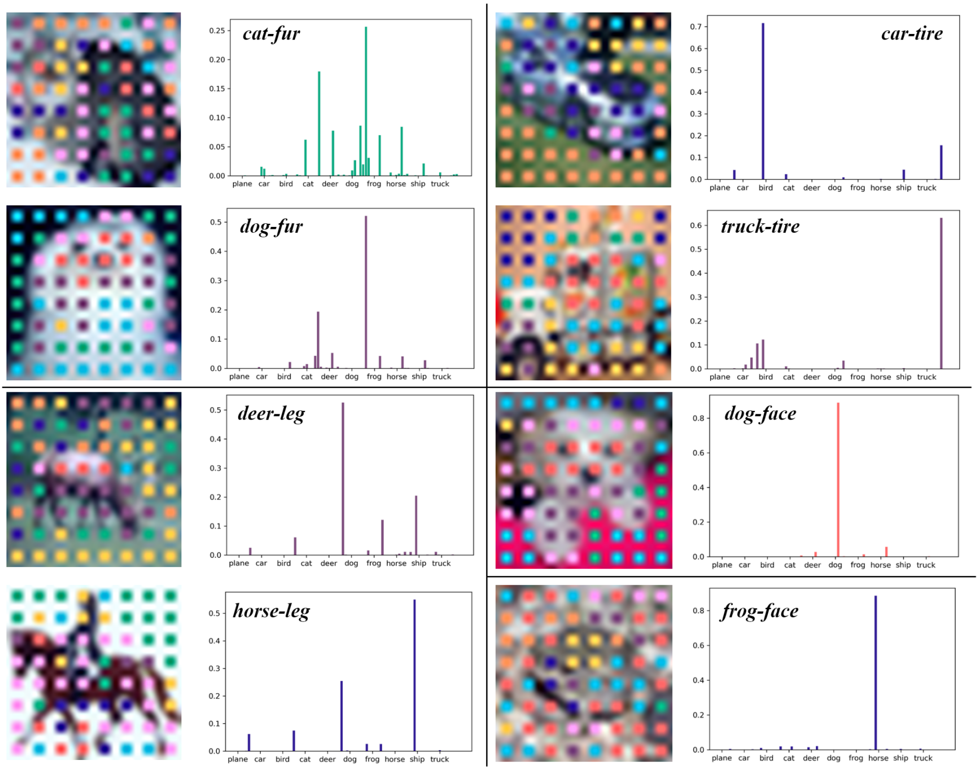

Figure 2: Sample images with the teacher sub-class annotations marking the center of each spatial location. Each color corresponds to a distinct sub-class. On the right of each image, the histogram for the sub-class assignments are plotted, where x-axis corresponds to subclass indices. The indices of the sub-classes associated with its super-class lie on the right of the index ticked by the class label.

Transferred knowledge. To support the effectiveness of the proposed intermediate layer supervision, we demonstrate the extracted information from the teacher on inter-category relations between the different sub-classes and category-specific patterns for the images in Fig. 2 using CIFAR-10 with RN56. We mark the center of each patch with respect to its sub-class (denoted by its color). We also provide annotation of a sub-class ($p_{\mathcal{T}}(i) = p(h_2(z_i) \mid h_1(z_i, y))$ in § 4.2.2 in the main paper) on the right, which shows how discriminating that patch for the teacher for the main task. We observe shared entities such as *tire* for *truck* and *car*, *leg* for *horse* and *deer*, *fur* for *cat* and *dog*. We are also able to observe discriminative patterns such as *face* of a *cat* and a *dog*. Through exploiting this information, the templates of the student (*i.e.*, 1x1 kernels) adapt to those patterns that the teacher finds useful to discriminate categories. Besides uniquely differentiating discriminative patterns, the student also learns to acknowledge the shared entities by combining them with their learned semantic features. That way, the uninformative patterns such as *fur* that have less peaky distribution can be exploited to represent coarse categories, *e.g.*, *belonging to animals*, or can be completely discarded by the student if their matching scores drop below the average due to following an almost uniform distribution.

2 Empirical Study Details

In the following sections, we detail our experimental setup, including the utilized datasets, and fully disclose our implementation specifics.

2.1 Reproducibility

We provide full details of our experimental setup and recapitulate the implementation details for the sake of complete transparency and reproducibility. Code is available at: letKD Framework

2.2 Experimental Setup

Datasets. We adopted three benchmark image classification datasets to extensively evaluate our method, which are also widely used in KD literature. These datasets include CIFAR-100 [12], which contains 100 classes with 50K training images and 10K test images of size 32x32; Tiny-ImageNet [13], which contains 200 classes with 500 training images, 25 validation images and 25 test images of size 64x64 for each class; ImageNet [4], which consists over 1.2 million images for training and 50K images for validation which are distributed over 1000 classes. For data augmentation, we use standard operations including normalization that are commonly used in other KD algorithms [11].

Implementation Details. We perform our experiments on various architectures including ResNet (RN) [6, 7], Wide ResNet (WRN) [24], MobileNet (MNV2) [17] and ShuffleNet (SNV1/V2) [14, 26]. We adopt the framework implemented by [11] in PyTorch [15] to make a fair and unbiased evaluation of our method as well as comparisons with the other invented methods. For the results in Tabs. 1 to 3, we use the results of the relevant methods from [11, 18, 27]. For SimKD [3], we use their official implementation to obtain the corresponding results.

Specifically, for all datasets, we adopt the SGD optimizer with 0.9 Nesterov momentum. For CIFAR-100 and Tiny-ImageNet, we trained for 240 epochs in which the learning rate is divided by 10 at 150th, 180th and 210th epochs. For heterogeneous (MNV2 and SNV1/V2 as students) trainings, we set the initial learning rate as 0.01 and for other architectures, we set the initial learning rate as 0.05.

For all evaluations on ImageNet, we trained for 100 epochs with an initial learning rate 0.1, which is divided by 10 at 30th, 60th and 90th epochs.

Formulation of 1x1 convolution. In the proposed KD layer including 1x1-BN-ReLU-1x1 block, we employ 1x1 operations as *normalized* 1x1 *convolutions* with a learnable scale, *i.e.*, the kernels are $\ell 2$ normalized and scaled. For us, the two *normalized convolutions* serve distinct purposes. For the first 1x1, we utilize this by first normalizing our input features and kernels since we want to measure the cosine similarity between them (*i.e.*, their alignment). For the second 1x1, we try to decrease the dependency of the hyperparameter $\{\alpha_{inter}, \alpha_{penult}\}$ through the normalized and scaled outputs to have a more stable learning. Owing to this mechanism, we ease the process of adding the importance and attention gathered from 1x1-BN-ReLU-1x1 to the features at the shortcut connection.

Heterogeneous distillation cases. For heterogeneous cases, we transform the features of the teacher before obtaining the soft assignments by applying an additional average pooling operation before quantization (*K-means* operation) to align its spatial dimensions with the predictions of the student. For instance, at the penultimate layer, RN32x4 and WRN-40-2 have 8x8 feature maps, SNV1/V2 and RN50 have 4x4 feature maps, and MNV2 has 2x2 feature maps. Hence, the spatial dimensions between the teacher and the student should be aligned to apply the distillation loss properly.

3 Implementations with Pseudo-codes

Algorithm 1 TEACHER PENULTIMATE LAYER KD

offline:

input: $X = \{x_i\}_{i \in [\mathcal{X}_T]}$, K_{penult}, θ_t //all training images, # of cluster centers, //parameters of the teacher

$F \leftarrow f_t^{(-1)}(X; \theta_t)$ //teacher's features at the penultimate layer, //$F \in \mathbb{R}^{[\mathcal{X}_T].w.hxd}$

$\{\rho_k\}_{k \in [K_{penult}]} \leftarrow \text{KMeans}(F, K_{penult})$ //clustering operation to the teacher's features

return $\{\rho_k\}_{k \in [K_{penult}]}$ //cluster centers in the quantized space

online:

input: $X = \{x_i\}_{i \in [b]}, \{\rho_k\}_{k \in [K_{penult}]}, \theta_t$ //batch of images, cluster centers, parameters //of the teacher

$F \leftarrow f_t^{(-1)}(X; \theta_t)$ //teacher's features at the penultimate layer, //$F \in \mathbb{R}^{[b].w.hxd}$

$d \leftarrow [\|F - \rho_k\|_2^2]_{k \in [K_{penult}]}$ //distance of each spatial location to the //cluster centers

$p_{\mathcal{T}} \leftarrow \text{softmax}(d)$ //soft assignments of the teacher for each //spatial location

return $p_{\mathcal{T}}$ //soft assignments of the teacher

Algorithm 2 TEACHER INTERMEDIATE LAYER KD

offline:

input: $(X, Y) = (\{x_i\}, \{y_i\})_{i \in [\mathcal{X}_{\mathcal{T}}]}$, K_{inter}, θ_t, C //all training image-label pairs, # of sub-classes, //parameters of the teacher, # of classes

$F \leftarrow f_t^{(l')}(X; \theta_t)$ //teacher features at l'th layer

$W_{\text{LDA}}, b_{\text{LDA}} \leftarrow \text{LDA}(F, Y)$ //obtain weight and bias for LDA

$F_{\text{LDA}} \leftarrow \text{Conv1x1}(F, W_{\text{LDA}}, b_{\text{LDA}})$ //apply LDA, $F_{\text{LDA}} \in \mathbb{R}^{[b].w.h \times d_{\text{LDA}}}$

for $c = 1 : C$ **do**

$F_{\text{LDA}}^c \leftarrow F_{\text{LDA}}[y = c]$ //obtain the LDA features belonging to class c

$\rho_c, P^c \leftarrow \text{KMeans}(F_{\text{LDA}}^c, K_{inter})$ //obtain clusters and predictions through //K-means clustering

for $k = 1 : K_{inter}$ **do**

$F_{\text{LDA}}^{k,c} \leftarrow F_{\text{LDA}}[P^c = k]$ //obtain LDA features belonging to sub-class k //in class c

$\text{prot}^{k,c} \leftarrow \text{mean}(F_{\text{LDA}}^{k,c})$ //obtain representative prototype for sub-class //k in class c

$\text{prot} \leftarrow \{\text{prot}^{k,c}\}_{k \in [K_{inter}]}^{c \in [C]}$

$\{s_{\mathcal{T}}^{k,c}\}_{k \in [K_{inter}]}^{c \in [C]} \leftarrow \text{NNSearch}(F_{\text{LDA}}, \text{prot})$ //apply NN Search between features and //prototypes

$s_{\mathcal{T}} \leftarrow \{s_{\mathcal{T}}^{k,c}\}_{k \in [K_{inter}]}^{c \in [C]}$ //scores for all sub-classes, //$s_{\mathcal{T}} \in \mathbb{R}^{K_{inter}.C \times K_{inter}.C}$

return $s_{\mathcal{T}}, W_{\text{LDA}}, b_{\text{LDA}}, \{\rho_c^k\}_{c \in [C]}^{k \in [K_{inter}]}$ //scores, LDA parameters, cluster centers

online:

input: $(X, Y) = (\{x_i\}, \{y_i\})_{i \in [b]}, s_{\mathcal{T}}, W_{\text{LDA}}, b_{\text{LDA}}, \{\rho_c^k\}_{c \in [C]}^{k \in [K_{inter}]}, \theta_t$ //batch of image-label pairs, scores, LDA //parameters, cluster centers, parameters of the //teacher

$F_{\text{LDA}} \leftarrow \text{Conv1x1}(f_t^{(l')}(X; \theta_t), W_{\text{LDA}}, b_{\text{LDA}})$ //apply LDA to the teacher's features at l'th layer //$F_{\text{LDA}} \in \mathbb{R}^{[b].w.h \times d_{\text{LDA}}}$

$k^* \leftarrow \min([\|F_{\text{LDA}} - \rho_Y^k\|_2^2]_{k \in [K_{inter}]})$ //assign the closest sub-class cluster index for //class Y to each spatial location

$p_{\mathcal{T}} \leftarrow s_{\mathcal{T}}(k^*)$ //assign the score of the selected cluster

return $p_{\mathcal{T}}$ //soft assignments of the teacher

Algorithm 3 STUDENT KD SUPERVISION

online:

input: $X = \{x_i\}_{i\in[b]}$, $p_\mathcal{T}$, θ_s //batch of images, soft assignments of the teacher, //parameters of the student

$F \leftarrow f_s^{(l)}(X;\theta_s)$ //student's features at the lth (penultimate or //intermediate) layer

$\hat{F}, p_\mathcal{S} \leftarrow \text{KDLayer}(F;\theta_s)$ //see Fig. 2 in the main paper

$\mathcal{L}_{KD} \leftarrow \text{KLDiv}(p_\mathcal{T}, p_\mathcal{S})$ //distillation loss using teacher's soft assignments //and student's predictions

return $\hat{F}, \mathcal{L}_{KD}$ //output features of the KD layer, distillation loss

References

[1] 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 (CVPR)*, June 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Defang Chen, Jian-Ping Mei, Hailin Zhang, Can Wang, Yan Feng, and Chun Chen. Knowledge distillation with the reused teacher classifier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11933–11942, June 2022.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Ada Gorgun, Yeti Z. Gurbuz, and Aydin Alatan. Feature embedding by template matching as a resnet block. In *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[8] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[11] Himalaya Jain, Spyros Gidaris, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Quest: Quantized embedding space for transferring knowledge. *European Conference on Computer Vision (ECCV)*, 2020.

[12] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[13] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[14] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: practical guidelines for efficient cnn architecture design. In *European Conference on Computer Vision (ECCV)*, pages 122–138, 2018.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[16] 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.

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[18] Jie Song, Haofei Zhang, Xinchao Wang, Mengqi Xue, Ying Chen, Li Sun, Dacheng Tao, and Mingli Song. Tree-like decision distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13488–13497, June 2021.

[19] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[20] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[21] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[22] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[23] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138, 2017.

[24] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[25] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

[26] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[27] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11953–11962, June 2022.

Appendix

BN-ReLU as a Soft Maximizer

To strengthen our approximation of BN-ReLU as a soft maximizer considering the problem:

$$p_{|i} = \underset{p,q\geqslant 0}{\arg\max}\, q\,\mu + \Sigma_k p_k\, \omega_k^{\intercal} x_i \quad \text{s.to} \quad q + \Sigma_k p_k = 1 \tag{3.1}$$

we start with the explanation of the overall process, where x_i represents a local region i in a feature map x, $\{\omega_k \in \mathbb{R}^d\}_{k\in[K]}$ are the 1x1 kernels, and μ is a threshold enabling to zero out the embedding vector if no kernel matches with at least μ similarity. As we state in the paper, we make this problem differentiable by employing entropy smoothing to the problem in (3.1) as:

$$p_{|i} = \underset{p,q\geqslant 0}{\arg\max}\, q\,\mu + p^{\intercal} a_{|i} - \tfrac{1}{\epsilon}(q \log q + p^{\intercal} \log p) \quad \text{s.to} \quad q + \Sigma_k p_k = 1 \tag{3.2}$$

and obtain a soft-max solution:

$$p_{k|i} = \tfrac{\exp(\epsilon a_{k|i})}{\exp(\epsilon\mu) + \Sigma_{k'} \exp(\epsilon a_{k'|i})} \tag{3.3}$$

where $a_{k|i} = \omega_k^{\intercal} x_i$ and ϵ controls the smoothness of $p_{|i}$. As can be seen from (3.3), apart from the temperature parameter ϵ, we only need to add an additional dimension with the value μ to the channels of $a_{|i}$ to mimick the threshold in (3.1). Yet, the problem here is finding proper μ and ϵ values. Indeed, BN-ReLU is shown to mitigate that problem in [5] and the equivalence of the solution in (3.3) and BN-ReLU (up to a scale) is empirically validated.

We now derive an alternative equivalence to rather explicitly show that replacing soft-max with BN-ReLU inherently makes the model learn these parameters while performing a scaled version of soft-max.

Note that BN [11] and its successor counterparts [2, 21, 22] perform activity normalization using some batch statistics as:

$$\mathrm{BN}(a_k) = \gamma_k \frac{a_k - \mathbb{E}[a_k]}{\sqrt{\mathrm{Var}(a_k)}} + \beta_k = \gamma_k \hat{a}_k + \beta_k \tag{3.4}$$

which can be interpreted as *whitening* its input with a learnable scale and bias, where $\mathbb{E}[a_k]$ and $\mathrm{Var}[a_k]$ are calculated using the whole batch.

To employ BN-ReLU as a replacement of (3.2), we first consider the whitened version of our activations, *i.e.*, $\hat{a}_k = \frac{a_k - \mathbb{E}(a_k)}{\sqrt{\mathrm{Var}(a_k)}}$, to be used in (3.3) and use a scale γ_k to make their values around 0 as $a'_k = \gamma_k \hat{a}_k$. Then, when we apply soft-max to $a'_{k|i}$, we can use the first order Taylor series expansion to approximate the unnormalized soft-max operation applied to them as:

$$e^{a'_{k|i}} \approx 1 + a'_{k|i} + \frac{a'^{2}_{k|i}}{2!} + \frac{a'^{3}_{k|i}}{3!} + \ldots \approx 1 + a'_{k|i} + \mathrm{err} \tag{3.5}$$

where err is an error owing to the higher order terms. When we consider the expression in (3.3) with the inclusion of temperature ϵ, we can say that for certain k's, $p_{k|i}$ will go to zero if $\exp(\epsilon a_{k|i})$ is way smaller than the denominator (sum of all exponential terms including the effect of the threshold parameter μ) of (3.3). Moreover, since the soft-max formulation is linearized in (3.5), the output might be negative. Hence, if we employ the solution in (3.5), the condition that should be satisfied for $p_{k|i}$ to be non-zero and non-negative would be:

$$1 + a'_{k|i} + \mathrm{err} > \mathrm{th} \rightarrow 1 + a'_{k|i} + \mathrm{err} - \mathrm{th} > 0 \tag{3.6}$$

where the terms $\{1, \mathrm{th}, \mathrm{err}\}$ can be combined into a single term β (involving the effect of μ) as:

$$a'_{k|i} + \beta_k > 0 \rightarrow \gamma_k \hat{a}_{k|i} + \beta_k > 0 \tag{3.7}$$

Internally, the constraint defined in (3.7) mimicks the function ReLU. When this constraint is satisfied, the terms $\gamma_k \hat{a}_{k|i} + \beta_k$ of the unnormalized soft-max would be counted as the corresponding outputs. Hence, when we consider the relationship between (3.7) and (3.4), if we use BN+ReLU as a replacement of (3.3), we can simply employ $\hat{p}_k = \max\{0, \gamma_k \hat{a}_k + \beta_k\}$, *i.e.*, ReLU, to zero-out the assignment vector and let BN learn the proper parameters, (β_k, γ_k), using the batch statistics to assess the poor matching scores. For the pixels with non-zero activations after BN-ReLU, we can obtain the normalized assignment vector as $\hat{p}_k/\eta$ where $\eta := \Sigma_k \hat{p}_k$. That being said, we empirically find that, absorbing η into α (Eq. 4.1 in the main paper) and α into γ_k, are useful to adaptively put more emphasis on the teacher's knowledge according to the matching scores.