A. Appendix

A.1. Details about the reformulation in Sec 3.1

Details of the mathematical derivation in Sec 3.1 of the manuscript are as follows (notations are the same in Sec 3.1 of the manuscript):

$$\begin{aligned}\text{KD} &= \text{KL}(\mathbf{p}^{\mathcal{T}}||\mathbf{p}^{\mathcal{S}}) \\ &= \sum_{i=1}^{C} p_i^{\mathcal{T}} \log(\frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}}) \\ &= p_t^{\mathcal{T}} \log(\frac{p_t^{\mathcal{T}}}{p_t^{\mathcal{S}}}) + \sum_{i=1,i\neq t}^{C} p_i^{\mathcal{T}} \log(\frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}}).\end{aligned} \tag{8}$$

According to Eqn.(1) and Eqn.(2) of the manuscript, we have $\hat{p}_i = p_i/p_{\backslash t}$. Thus, we can rewrite Eqn.(8) to:

$$\begin{aligned}\text{KD} &= p_t^{\mathcal{T}} \log(\frac{p_t^{\mathcal{T}}}{p_t^{\mathcal{S}}}) + \sum_{i=1,i\neq t}^{C} p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}\hat{p}_i^{\mathcal{S}}}) \\ &= p_t^{\mathcal{T}} \log(\frac{p_t^{\mathcal{T}}}{p_t^{\mathcal{S}}}) + \sum_{i=1,i\neq t}^{C} p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}} (\log(\frac{\hat{p}_i^{\mathcal{T}}}{\hat{p}_i^{\mathcal{S}}}) + \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}})) \\ &= p_t^{\mathcal{T}} \log(\frac{p_t^{\mathcal{T}}}{p_t^{\mathcal{S}}}) + \sum_{i=1,i\neq t}^{C} p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}} \log(\frac{\hat{p}_i^{\mathcal{T}}}{\hat{p}_i^{\mathcal{S}}}) \\ &+ \sum_{i=1,i\neq t}^{C} p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}}).\end{aligned} \tag{9}$$

Since $p_{\backslash t}^{\mathcal{T}}$ and $p_{\backslash t}^{\mathcal{S}}$ are irrelevant to the class index i, we have:

$$\begin{aligned}\sum_{i=1,i\neq t}^{C} p_{\backslash t}^{\mathcal{T}}\hat{p}_i^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}}) &= p_{\backslash t}^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}}) \sum_{i=1,i\neq t}^{C} \hat{p}_i^{\mathcal{T}} \\ &= p_{\backslash t}^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}}).\end{aligned} \tag{10}$$

Then,

$$\text{KD} = \underbrace{p_t^{\mathcal{T}} \log(\frac{p_t^{\mathcal{T}}}{p_t^{\mathcal{S}}}) + p_{\backslash t}^{\mathcal{T}} \log(\frac{p_{\backslash t}^{\mathcal{T}}}{p_{\backslash t}^{\mathcal{S}}})}_{\text{KL}(\mathbf{b}^{\mathcal{T}}||\mathbf{b}^{\mathcal{S}})} + p_{\backslash t}^{\mathcal{T}} \underbrace{\sum_{i=1,i\neq t}^{C} \hat{p}_i^{\mathcal{T}} \log(\frac{\hat{p}_i^{\mathcal{T}}}{\hat{p}_i^{\mathcal{S}}})}_{\text{KL}(\hat{\mathbf{p}}^{\mathcal{T}}||\hat{\mathbf{p}}^{\mathcal{S}})}. \tag{11}$$

According to the definition of KL-Divergence, Eqn.(11) can be rewritten as (which is the same as Eqn.(5) of the manuscript):

$$\text{KD} = \text{KL}(\mathbf{b}^{\mathcal{T}}||\mathbf{b}^{\mathcal{S}}) + (1 - p_t^{\mathcal{T}})\text{KL}(\hat{\mathbf{p}}^{\mathcal{T}}||\hat{\mathbf{p}}^{\mathcal{S}}) \tag{12}$$

A.2. Implementation: Experiments in Sec 4

CIFAR-100: Our implementation for CIFAR-100 follows the practice in [33]. Teachers and students are trained for 240 epochs with SGD. As the batch size is 64, the learning rates are 0.01 for ShuffleNet [21] and MobileNet-V2 [30], 0.05 for the other series (*e.g*. VGG [32], ResNet [9] and WRN [42]). The learning rate is divided by 10 at 150, 180 and 210 epochs. The weight decay and the momentum are set to 5e-4 and 0.9. The weight for the cross-entropy loss is set to 1.0. The temperature is set as 4 and α is set as 1.0 for all experiments. The proper value of β could be different for different teachers, and the details and discussions are in the next section. And we utilize a 20-epoch linear warmup for all experiments since the value of β could be high leading to a large initial loss.

ImageNet: Our implementation for ImageNet follows the standard practice. We train the models for 100 epochs. As the batch size is 512, the learning rate is initialized to 0.2 and divided by 10 for every 30 epochs. Weight decay is 1e-4 and the weight for the cross-entropy loss is set to 1.0. We set temperature as 1 and α as 0.5 for all experiments. Strictly following [1, 33], for distilling networks of the same architecture, the teacher is ResNet-34 model, the student is ResNet-18, and β is set to 0.5. For different series, the teacher is ResNet-50 model, the student is MobileNet-V2, and β is set to 2.0.

MS-COCO: Our implementation for MS-COCO follows the settings in [1]. We use the two-stage method Faster R-CNN [27] with FPN [19] as the feature extractors. ResNet [9] models and MobileNet-V2 [30] are selected as teachers and students. All students are trained with the 1x scheduler (schedulers and task-specific loss weights follow Detectron2 [39]). We employ the DKD loss on the R-CNN head, and set α as 1.0, β as 0.25, and temperature as 1 for all experiments.

Results of compared methods are reported in their original papers or reproduced by previous works [1, 33].

A.3. Guidance for tuning β

We suppose that the importance of NCKD in knowledge transfer could be related to the confidence of the teacher. Intuitively, the more confident the teacher is, the more valuable the NCKD could be, and the larger β should be applied. However, NCKD could increase the gradient contributed by logits of non-target classes. Thus, an improper large β could harm the correctness of the student's prediction. If the logit value of the target class is much higher than all non-target classes, the teacher could be regarded as more confident and a large $beta$ could be more reasonable. Thus, we suppose that the value of β could be related to the logit value *gap* between the target and all non-target classes. Specifically, the *gap* between the logit of the target class (*i.e*., z_t, where z represents the output logit and t represents the target class) and the max logit among non-target classes could be reliable guidance for tuning β, which can be denoted as $z_t - z_{max}$, where $z_{max} = \max(\{z_i | i \neq t\})$.

β	ResNet56	WRN-40-2	ResNet32x4
1.0	76.02	75.94	74.95
2.0	**76.32**	76.25	75.64
4.0	75.91	76.17	75.82
6.0	75.62	**76.70**	76.34
8.0	75.33	76.44	**76.45**
10.0	75.35	76.21	76.32
$z_t - z_{max}$	5.36	7.24	8.40

Table A.1. Accuracy(%) on CIFAR-100 [16] with different β and different teachers. The *gap* ($z_t - z_{max}$) is also reported.

We report experimental results on CIFAR-100 [16] to verify this conjecture. We select ResNet56, WRN-40-2 and ResNet32×4 as teachers and ShuffleNet-V1 as the student, and apply different β. Both top-1 accuracy (%) and the *gap* $z_t - z_{max}$ (averaged over all training samples) are reported. As shown in Table A.1, the best value of β could be positively proportional to the *gap*, which we suppose could be guidance of tuning β and a direction for further research. Based on this, the value of β for each teacher in Table 6 and Table 7 of the manuscript is set as follows (in Table A.2):

teacher	$z_t - z_{max}$	β
ResNet56	5.36	2.0
ResNet110	6.73	2.0
WRN-40-2	7.24	6.0
VGG13	8.25	6.0
ResNet50	8.53	8.0
ResNet32×4	8.40	8.0

Table A.2. The value of β for different teachers in Table 6 and Table 7 of the manuscript.

A.4. Implementation: Experiments in Sec 3.2

In this part, we report the implementation details of the experiments in Sec 3.2 of the manuscript.

Basic settings. We set the loss term of KD and CE as 1.0 and 1.0, respectively (instead of the default $0.1CE + 0.9KD$ setting in [33]). The setting in [33] follows the loss form proposed by [12], which assumes that the sum of all terms' weights should be 1.0. However, the NCKD loss is target-irrelevant, which means the target-relevant loss could be 0.1 if we utilize the original setting when only applying NCKD. Based on this, we set the loss weight of all terms (*e.g.*, KD, TCKD, NCKD and CE) as 1.0 for all experiments in Sec 3.2 of the manuscript.

Strong augmentation. We employ the AutoAugment [5] to reveal the effectiveness of TCKD in Sec 3.2 of the manuscript. Specifically, we add the CIFAR AutoAugment policy[7] after applying the default augmentation (random crop and horizontal flip). Then we train the teacher and the student with the same augmentation policy.

[7]https://github.com/DeepVoltaire/AutoAugment

Noisy labels. We also perform experiments on noisy training data to verify that TCKD conveys the knowledge about sample "difficulty". Specifically, we follow the settings of [7, 35], utilizing the symmetric noise type[8]. We train a teacher network on the noisy training data and select the best epoch to distill the student (on the same training data).

A.5. Explanation about why TCKD brings performance drop in Table 1

In Table 1 of the manuscript, we reveal that singly applying TCKD could bring performance drop sometimes. An explanation for this phenomenon is that the high temperature (T=4) will lead to a great gradient to increase the non-target classes' logits, which will harm the correctness of the student's prediction. Without NCKD, the information about the class similarity (or the prominent dark knowledge) is not available, so that TCKD's gradient could do no good but lead to performance drop (since TCKD could bring marginal performance gain on easy-fitting training data). To verify that the large temperature is not proper when singly applying TCKD, we perform experiments with different temperatures (T) in the table below. Results in Table A.3 show that the performance is almost the same as the vanilla training baseline (73.26 in Table 1 of the manuscript) when the temperature is set as 1. And the performance drop is positively related to the temperature.

T	1	2	3	4
top-1	73.24	73.05	71.69	70.96

Table A.3. Accuracy (%) with different temperature(T) when only applying TCKD. The teacher and the student are set as WRN-40-2 and WRN-16-2, respectively.

A.6. How to employ DKD on detectors

In this paper, we employ our DKD on the two-stage object detector Faster R-CNN. We only employ our DKD on the R-CNN head. Specifically, given a student network, we utilize the labels assigned to the proposals (generated by the RPN module) as the target class(if *IoU(proposal)* < 0.5, the target class is set as "background"). Then, we use a teacher network to get the R-CNN prediction logits of the *same proposals* (locations are the same, while the features are from the teacher's backbone). Thus, we can employ our DKD by minimizing the KL-Divergence (*i.e.*, TCKD and NCKD) between the student's logits and the teacher's.

A.7. Implementation: Experiments in Sec 4.2

Training efficiency. We report the training time of each distillation method in Figure 2 of the manuscript. The

[8]https://github.com/bhanML/Co-teaching/blob/master/data/cifar.py

training time (per batch) is the sum of (1) the data processing time (*e.g.*, including the time to sample the contrast examples in [33]), (2) the network forward time and the gradient backward time and (3) the memory updating time (*e.g.*,including the time to update the contrast memory in [33]). We also report the number of extra parameters for each method. Besides the learnable parameters (*e.g.*, the connectors in [10] and the ABF modules in [1]), we also calculate the extra dictionary memory(*e.g.*, the contrast memory in [33]).

Feature transferability. We perform linear probing experiments to verify the feature transferability of our DKD in Sec 4.2 of the manuscript. We use the WRN-16-2 distilled from a WRN-40-2 teacher as the feature extractor (only using the feature generated by the final global average pooling module), then train linear fully-connected (FC) layers as classifier modules for STL-10 and Tiny-ImageNet datasets (the feature extractor is fixed during training). We train the FC via an SGD optimizer with 0.9 momentum and 0.0 weight decay. The number of total epochs is set as 40, and the learning rate is set to 0.1 for a 128 batch size and divided by 10 for every 10 epochs.