De-confounded Data-free Knowledge Distillation for Handling Distribution Shifts

Supplementary Material

In this supplementary material, we provide more details of our method, organized as follows:

- In Section 1, we provide the detailed training settings and illustrate how KDCI combines with existing DFKD methods, and show the algorithm process, corresponding to Section 4.3 of the main body.
- In Section 2, we qualitatively assess students' learning progress about vanilla DFKD methods and their KDCI-based version to verify the positive effect of KDCI on the existing DFKD method.
- In Section 3, we analyze the possible reasons for the difference in performance improvement, corresponding to Section 4 of the main body.
- In Section 4, we provide more observable visualization results as more sufficient evidence, corresponding to Section 4.7 of the main body.
- In Section 5, we discuss the significant differences between our KDCI and other methods focusing on data distribution.
- In Section 6, we discuss the broader impact and potential limitations.
- In Section 7, we provide the detailed experimental settings for the used baseline methods, corresponding to Section 4.2 of the main body.

1. Additional Training Details & Algorithm Process of Combining KDCI with Existing DFKD Methods

1.1. Training Details

We provide the detailed experimental settings for our KDCI framework. Our KDCI and reproducible methods are implemented through PyTorch [11]. All models are trained on RTX 3090 GPUs. **For CIFAR-10 and CIFAR-100**, all training settings (*e.g.*, loss function, optimizer, batch size, learning rate, etc) of the reported methods are consistent with the released codebase. The results are shown in Table 1 of the main body. **For Tiny-ImageNet**, initially, we try to find a unified teacher model for the Tiny-ImageNet dataset in open-sourced projects. However, one problem is that the teacher model pre-trained on Tiny-ImageNet seems confidential, so finding an open-source unified model is difficult. In this case, we train the unified renset-34 teacher model for 200 epochs on the original training data. During the teacher's training, we use the SGD optimizer with the momentum as 0.9, weight decay as $5e{-}4$, the batch size as 128, and cosine annealing learning rate with an initial value of 0.1. The teacher model can converge without additional tuning. Based on this pre-trained teacher, we train all students for 200 epochs. For the student, we use the SGD optimizer with the momentum as 0.9, the weight decay as $1e{-}4$, the batch size as 256, the cosine annealing learning rate with an initial value of 0.2 for Fast [10], and 0.1 for DeepInv [1] & DFND [5]. The results are shown in Table 2 of the main body. **For ImageNet**, We choose the same pre-trained resnet-50 model with [14] and unify the teacher model of different baseline methods. For Fast, we test directly on the open-source project. For DeepInv, we reproduce the corresponding results with the specified backbone pair. For DFND, we select 600k samples from the unlabeled FlickerlM dataset. The teacher's backbone is different from the original paper. The different backbones may cause the

Algorithm 1 Training process of generation-based methods combined with our KDCI

```
Input: A pre-trained teacher model $\boldsymbol{T}$, a generator $g$, a student model $\boldsymbol{S}$, distillation epochs $T$, batch size $N_m$, the iterations of generator $g$ in each epoch $Tg$, the iterations of student $f_s$ in each epoch $Ts$, the confounder size $N$.
for epoch $= [1, \dots, T]$ do
    // Generation stage
    for generator iterations $= [1, \dots, Tg]$ do
        Randomly sample noises and labels $(\mathrm{z}, y)$
        Synthesize a mini-batch training data $\boldsymbol{X} = g(\mathrm{z}, y)$
        Update generator $g$ with the generator loss
    end for
    Synthesize training data $\boldsymbol{X} = g(\mathrm{z}, y)$. Obtain the prediction feature $M = \left\{m_j \in \mathbb{R}^d\right\}_{j=1}^{N_m}$
    Prototype clustering for $M$. Calculate the number of the prediction features in $i$-th cluster $N_i$, the feature cluster $\sum_{k=1}^{N_i} m_k^i$ and the subcenter $\boldsymbol{z}_i = \frac{1}{N_i}\sum_{k=1}^{N_i} m_k^i$.
    Construct a confounder dictionary $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_N]$ and calculate the prototype proportion $P_s(\boldsymbol{z}_i) = N_i / N_m$
    // Distillation stage
    for student iterations $= [1, \dots, Ts]$ do
        Synthesize training data $\boldsymbol{X} = g(\mathrm{z}, y)$. Get models's predictions $\boldsymbol{T}(\boldsymbol{X})$ and $\boldsymbol{S}(\boldsymbol{X})$
        Calculate the prior information: $F(\boldsymbol{z}) = \sum_{i=1}^{N} \lambda_i \boldsymbol{z}_i P_s(\boldsymbol{z}_i)$
        Compensate the student's predictions: $\boldsymbol{S}'(\boldsymbol{X}) = \phi(\boldsymbol{S}(\boldsymbol{X}), F(\boldsymbol{z}))$
        Update the student $\boldsymbol{S}$ with $KD\langle \boldsymbol{T}(\boldsymbol{X}), \boldsymbol{S}'(\boldsymbol{X}) \rangle$
    end for
end for
Output: The student model $\boldsymbol{S}$.
```

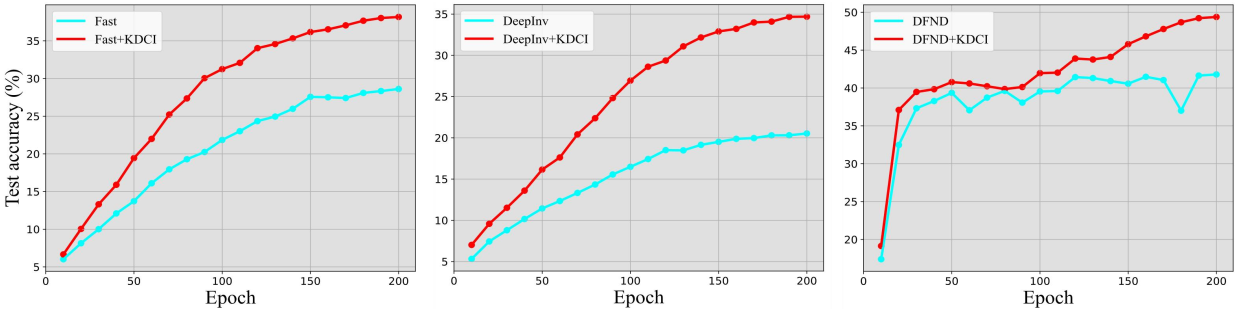


Figure 1. The test accuracy on Tiny-ImageNet dataset across different local training epochs $E = \{10, 20, \ldots, 200\}$. Our KDCI framework improves the performance of baselines consistently.

results we reproduce to differ from the original paper. The results are shown in Table 1 of the supplementary material. For the implementation of our KDCI, the hidden dimension d_n is set to 256. And d_h equals the hidden dimension d and the number of classes. By default, $\phi(\cdot)$ uses feature addition. For various baseline methods, the settings are shown in Section 7 of the supplementary material.

1.2. Algorithm Process

In the existing DFKD task, the generation-based and sampling-based method processes are different. Therefore, the way KDCI combines these methods and the hyperparameter settings are also slightly different. For the generation-based process, the generator and student models are updated alternately, which means the student's training data is updated in each epoch. We use a mini-batch of synthetic training data to construct the confounder dictionary, and the dictionary will be updated as the generator is updated. For the sampling-based process, existing methods select unlabeled data according to the preferences of the teacher model. Then, the student relies on these unlabeled data for data-based knowledge distillation training. We use all sampled data to construct the confounder dictionary. During subsequent student training, the dictionary is fixed. For a clearer understanding, we describe the above process as Algorithm 1 and 2, respectively.

2. Vanilla DFKD Methods vs. Their KDCI-based Versions

In the main body, we have compared the quantitative results of vanilla DFKD methods and their KDCI-based versions. To observe the positive effect of KDCI on the existing DFKD methods more clearly, we visualize the student's test accuracy on the Tiny-ImageNet dataset. The results are shown in Figure 1. KDCI can consistently help students from the beginning of training to the end, which verifies its effectiveness.

Algorithm 2 Training process of sampling-based methods combined with our KDCI

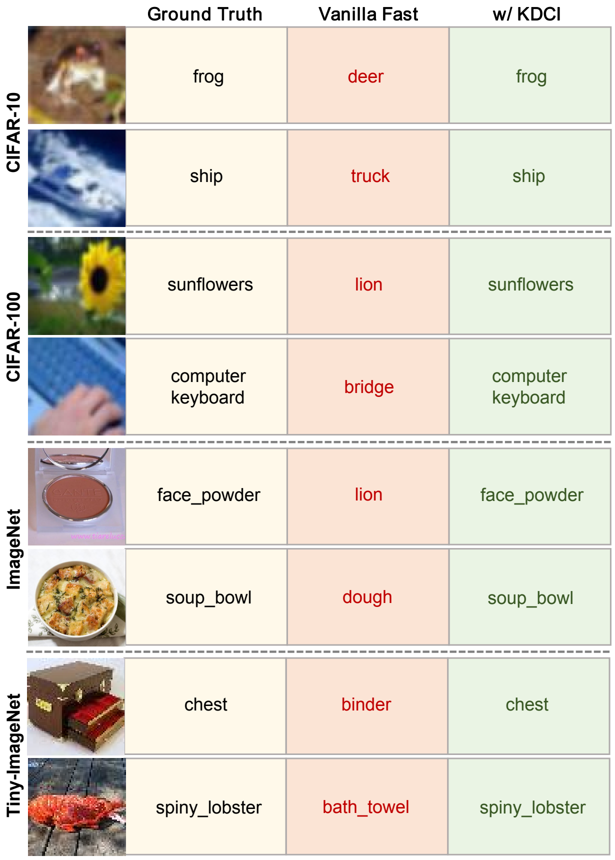
```
Input: A pre-trained teacher model $\boldsymbol{T}$, a student model $\boldsymbol{S}$, unlabeled training dataset $D = \{x_j\}_{j=1}^{n}$, distillation epochs $T$, batch size $m$, number of batches $M$, the number of sampled data $N_m$, the confounder size $N$.
// Sampling stage
Sample the training data $\{x_j\}_{j=1}^{N_m}$ from $D$. Obtain the prediction feature set $M = \left\{m_j \in \mathbb{R}^d\right\}_{j=1}^{N_m}$
Prototype clustering for $M$. Calculate the number of the prediction features in $i$-th cluster $N_i$, the feature cluster $\sum_{k=1}^{N_i} m_k^i$ and the subcenter $\boldsymbol{z}_i = \frac{1}{N_i}\sum_{k=1}^{N_i} m_k^i$.
Construct a confounder dictionary $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N]$ and calculate the prototype proportion $P_s(\boldsymbol{z}_i) = N_i/N_m$
// Distillation stage
for epoch $= [1, \ldots, T]$ do
    for mini-batch $= [1, \ldots, M]$ do
        Sample a mini-batch training data:
        $\boldsymbol{X} = \{x_i\}_{i=1}^{m}$ from $\{x_j\}_{j=1}^{N_m}$
        Get teacher and student predictions $\boldsymbol{T}(\boldsymbol{X})$ and $\boldsymbol{S}(\boldsymbol{X})$
        Calculate the prior information:
        $F(\boldsymbol{z}) = \sum_{i=1}^{N} \lambda_i \boldsymbol{z}_i P_s(\boldsymbol{z}_i)$
        Compensate the student's predictions:
        $\boldsymbol{S'}(\boldsymbol{X}) = \phi(\boldsymbol{S}(\boldsymbol{X}), F(\boldsymbol{z}))$
        Update the student $\boldsymbol{S}$ with $KD\langle \boldsymbol{T}(\boldsymbol{X}), \boldsymbol{S'}(\boldsymbol{X})\rangle$
    end for
end for
Output: The student model $\boldsymbol{S}$.
```

3. Analyses of Difference in Performance Improvements

Judging from the experimental results, KDCI has different gains for different DFKD methods on different datasets. We think such observations arise from various factors.

- By default, we choose the teacher model itself to extract the confounding dictionary. The prediction feature set provided by teachers of different backbones has different expressiveness, which affects the compensation degree of backdoor adjustment for bias during the causal intervention. The tests in Lines 513-531 and Table. 5 of the main body also verify this conclusion.

- The degree of distribution shift of synthetic data on distinct datasets is different. More complex datasets may degrade the generation quality for generation-based methods, resulting in more significant distribution shifts. KDCI tends to be more effective for more sophisticated datasets.
- Different baseline methods with different training losses are influential. Observations such as Section 4.4 of the main body suggest that methods that already incorporate prior likelihood knowledge of the data may weaken the KDCI gain.
- In addition, there may be many underlying factors. Nevertheless, KDCI, as a model-agnostic general framework, has promising and competitive improvements and gains for various models as a whole. We believe that a deeper exploration of the relevant mechanisms is a promising perspective. For this topic, we leave it to future work.



	Ground Truth	Vanilla Fast	w/ KDCI
CIFAR-10	frog	deer	frog
	ship	truck	ship
CIFAR-100	sunflowers	lion	sunflowers
	computer keyboard	bridge	computer keyboard
ImageNet	face_powder	lion	face_powder
	soup_bowl	dough	soup_bowl
Tiny-ImageNet	chest	binder	chest
	spiny_lobster	bath_towel	spiny_lobster

Figure 2. Qualitative results of the vanilla and KDCI-based version on CIFAR-10, CIFAR-100, ImageNet, and Tiny-ImageNet.

4. More Visual Evidence

To further verify the effectiveness, we provide more case studies of causal intervention. As shown in Figure 2, we visualize some test instances corrected by our KDCI compared to the vanilla version (Fast) on four kinds of datasets (*i.e.*, CIFAR-10, CIFAR-100, ImageNet, and Tiny-ImageNet). The vanilla version sometimes confuses some test instances due to shape or color. Our KDCI can repair these prediction shifts to enhance student performance.

5. Discussion with Other Works that Address Distribution Shifts

Several DFKD works already address distribution shifts in adversarial contexts [2, 3, 7, 12]. The works reveal distribution shift issues in the DFKD task from different aspects, but our method is significantly different from these works. Specifically, the differences between our KDCI and others are as follows:

- **Applicability.** These existing works tacitly use the same motivation, *i.e.*, as the generator gets updated, the distribution of synthetic data will change, causing the student to forget the knowledge it acquired at previous steps. However, such motivation does not apply to sampling-based methods. After selecting the training samples, they will not change during the entire student training process. Our motivation comes from the observed distribution shifts between the substitution data and can cover the two methods mentioned.
- **Economy.** Existing methods often rely on substantial additional computational and storage costs, *e.g.*, the need to store and maintain an additional dynamic collection of generated samples [3], the need for additional generator architectures to memorize knowledge of past generated data (an additional Variational Autoencoder (VAE) [2] or Exponential Moving Average generator [7]), and additional memory bank or additional loss calculation and gradient update [12]. In contrast, our method only needs to compute and store a small number of matrix computation results. Compared with the update of the models, the computational cost of the clustering process is basically negligible.
- **Plug-and-play.** Existing works are to propose new methods. Undoubtedly, these methods can provide a potential reference for other DFKD methods, but whether they can be easily combined with existing DFKD methods and improve overall performance is still unknown. Our proposed technique is model-agnostic, as a plug-and-play paradigm that integrates well with existing works. A large number of experiments have proved this conclusion.

6. Further Discussion

6.1. Broader Impact

The positive impact of this work: the proposed KDCI module can suppress the distribution shifts between the substitution and original data in the DFKD task, preventing the potential discrimination of the student's learning. While the pre-trained model for extracting prior knowledge uses the teacher itself, our method does not require additional dependencies and auxiliary information. The negative impacts of this work: students may be forced to identify minority groups for malicious purposes with customized biased teacher models. Therefore, we have to make sure that the DFKD technique is used for the right purpose.

6.2. Limitations

Since there are countless methods with insights for the DFKD task, other ways of classifying forms may also be reasonable. In this paper, we simply divide the source of the substitution data into generation-based and sampling-based methods. Similarly, it is impossible to cover all DFKD methods, so only open-source and representative methods are selected as the baseline. Nevertheless, the existing performance improvement is enough to prove the positive impact of KDCI on students.

In addition, since what we propose is a framework rather than a specific method, the test on the effectiveness of KDCI relies on the experimental setting of the existing DFKD methods. Currently, the mainstream open-source DFKD methods rarely use real-life medical or facial datasets for testing, so we only follow the mainstream experimental settings. Following the consensus of peers is necessary to increase the impact of our work. In this work, we select datasets that are widely used and accepted by the vast majority of DFKD methods. Following previous data paradigms is beneficial for acceptance by the relevant research community and enhances the persuasiveness of our method.

7. Experimental Setup of the Baseline DFKD Methods

DAFL. DAFL [4] is a data-free generation method. We keep the generator loss from the original as: $\mathcal{L}_{GEN} = \mathcal{L}_{oh} + \alpha\mathcal{L}_{a} + \beta\mathcal{L}_{ie}$. The knowledge distillation loss is: $\mathcal{L}_{KD} = D_{KL}(\mathcal{N}_S(x), \mathcal{N}_T(x))$. Following the original settings, we set $\alpha = 1e-3$, $\beta = 20$. We use SGD with the weight decay of $5e-4$, the momentum of 0.9, and the initial learning rate set as 0.1.

Fast. Fast [10] is a fast data-free generation method via feature sharing. We keep the generator loss from the original as: $\mathcal{L}_{GEN} = \alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{adv} + \gamma\mathcal{L}_{feat}$. The knowledge distillation loss is: $\mathcal{L}_{KD} = D_{KL}(\mathcal{N}_S(x), \mathcal{N}_T(x))$. We set $\alpha = 0.4$, $\beta = 1.1$, and $\gamma = 10$, which are the same as the original settings. We use the Adam Optimizer with a learning rate of $1e-3$ to update the generator and the SGD optimizer with a momentum of 0.9 and a learning rate of 0.1 for student training.

CMI. CMI [9] is a model inversion method with contrastive learning. We keep the generator loss from the original as: $\mathcal{L}_{GEN} = \alpha\mathcal{L}_{bn} + \beta\mathcal{L}_{cls} + \gamma\mathcal{L}_{adv} + \delta\mathcal{L}_{cr}$. The knowledge distillation loss is: $\mathcal{L}_{KD} = D_{KL}(\mathcal{N}_S(x), \mathcal{N}_T(x))$. We set $\alpha = 1$, $\beta = 0.5$, $\gamma = 0.5$, and $\delta = 0.8$. We use the Adam Optimizer with a learning rate of $1e-3$ to update the generator and the SGD optimizer with a momentum of 0.9 and a learning rate of 0.1 for student training.

DeepInv. DeepInv [13] is a model inversion method that combines prior knowledge and adversarial training. We keep the inversion loss from the original as: $\mathcal{L}_{GEN} = \alpha_{tv}\mathcal{R}_{tv} + \alpha_{l2}\mathcal{R}_{l2} + \alpha_f\mathcal{R}_{feature} + \alpha_c\mathcal{R}_{compete}$. The knowledge distillation loss is: $\mathcal{L}_{KD} = D_{KL}(\mathcal{N}_S(x), \mathcal{N}_T(x))$. We set $\alpha_{tv} = 2.5e-5$, $\alpha_{l2} = 3e-8$, $\alpha_f = 0.1$ and $\alpha_c = 10$, which are the same as the original setting. Besides, we set the number of iterations as 1000 and use Adam for optimization with a learning rate of 0.05.

DFND. DFND [5] is a sampling-based method using open-world unlabeled data as the substitution data. Following the original, we select 600k data with the highest teacher confidence from the ImageNet dataset [6] as the sampled data and resize them to the resolution of the corresponding dataset. We use the same noisy distillation loss $\mathcal{L}_{KD} = \mathcal{H}_{CE}(Q(\mathcal{N}_S(x)), \hat{y}) + \lambda D_{KL}(\mathcal{N}_S(x), \mathcal{N}_T(x))$, and λ is set as 4. The student network is optimized using SGD and the initial learning rate is set as 0.1 Weight decay and momentum are set as $5e-4$ and 0.9, respectively.

Mosaick. Mosaick [8] is a sampling-based method using out-of-domain (OOD) unlabeled data as the substitution data. We select 600k data with the lowest teacher confidence from the ImageNet dataset [6] as the OOD data. Following the original settings, we use Adam for optimization, with hyper-parameters $lr = 1e-3$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$ for the generator and discriminator. The distillation loss is $\mathcal{L}_{KD} = \lambda D_{KL} - \lambda\mathcal{R}(G, D, T)$ The student network is optimized using SGD, and the initial learning rate is set as 0.1. Weight decay and momentum are set as $1e-4$ and 0.9, respectively.

References

[1] Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework. *arXiv preprint arXiv:1905.07072*, 2019. 1

[2] Kuluhan Binici, Shivam Aggarwal, Nam Trung Pham, Karianto Leman, and Tulika Mitra. Robust and resource-efficient data-free knowledge distillation by generative pseudo replay. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 6089–6096, 2022. 3

[3] Kuluhan Binici, Nam Trung Pham, Tulika Mitra, and Karianto Leman. Preventing catastrophic forgetting and distribution mismatch in knowledge distillation via synthetic data. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 663–671, 2022. 3

[4] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3514–3522, 2019. 4

[5] Hanting Chen, Tianyu Guo, Chang Xu, Wenshuo Li, Chunjing Xu, Chao Xu, and Yunhe Wang. Learning student networks in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6428–6437, 2021. 1, 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009. 4

[7] Kien Do, Thai Hung Le, Dung Nguyen, Dang Nguyen, Haripriya Harikumar, Truyen Tran, Santu Rana, and Svetha Venkatesh. Momentum adversarial distillation: Handling large distribution shifts in data-free knowledge distillation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:10055–10067, 2022. 3

[8] Gongfan Fang, Yifan Bao, Jie Song, Xinchao Wang, Donglin Xie, Chengchao Shen, and Mingli Song. Mosaicking to distill: Knowledge distillation from out-of-domain data. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:11920–11932, 2021. 4

[9] Gongfan Fang, Jie Song, Xinchao Wang, Chengchao Shen, Xingen Wang, and Mingli Song. Contrastive model inversion for data-free knowledge distillation. *arXiv preprint arXiv:2105.08584*, 2021. 4

[10] Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. Up to 100x faster data-free knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 6597–6604, 2022. 1, 4

[11] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 1

[12] Gaurav Patel, Konda Reddy Mopuri, and Qiang Qiu. Learning to retain while acquiring: Combating distribution-shift in adversarial data-free knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7786–7794, 2023. 3

[13] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8715–8724, 2020. 4

[14] 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, 2022. 1