Information Sciences 642 (2023) 119202

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Dynamic data-free knowledge distillation by easy-to-hard learning strategy

Jingru Li, Sheng Zhou, Liangcheng Li, Haishuai Wang, Jiajun Bu, Zhi Yu *

Zhejiang University, Zheda Rd., Hangzhou, Zhejiang, China

ARTICLE INFO

Keywords:
Data-free knowledge distillation
Curriculum learning
Knowledge distillation
Self-paced learning

ABSTRACT

Data-free knowledge distillation (DFKD) is a widely-used strategy for Knowledge Distillation (KD) whose training data is not available. It trains a lightweight student model with the aid of a large pretrained teacher model without any access to training data. However, existing DFKD methods suffer from inadequate and unstable training process, as they do not adjust the generation target dynamically based on the status of the student model during learning. To address this limitation, we propose a novel DFKD method called CuDFKD. It teaches students by a dynamic strategy that gradually generates easy-to-hard pseudo samples, mirroring how humans learn. Besides, CuDFKD adapts the generation target dynamically according to the status of student model. Moreover, we provide a theoretical analysis of the majorization minimization (MM) algorithm and explain the convergence of CuDFKD. To measure the robustness and fidelity of DFKD methods, we propose two more metrics, and experiments shows CuDFKD has comparable performance to state-of-the-art (SOTA) DFKD methods on all datasets. Experiments also present that our CuDFKD has the fastest convergence and best robustness over other SOTA DFKD methods.

1. Introduction

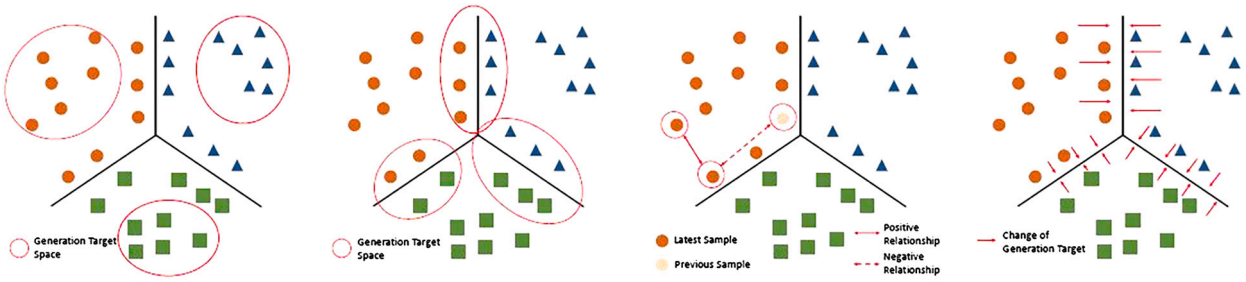
Knowledge Distillation (KD) is a popular strategy to train smaller student models by leveraging knowledge from large, pretrained teacher models, and it significantly improves the performance of small student models [17,49]. Due to privacy concerns [34] or resource constraints for training on large benchmarks like ImageNet, the training data is not accessible. To address this challenge, Data-Free Knowledge Distillation (DFKD) or Zero-Shot Knowledge Distillation (ZSKD) [40] has been proposed. It allows the training of student models without real training data, by using the teacher model's weights to reverse inference knowledge and generating pseudo samples to train the student model. This knowledge, which represents the pretrained weight of the teacher model, encapsulates prior information about the original training data and provides the target for the generation of the pseudo samples.

In Fig. 1a to 1c, different targets for the generation are set in different DFKD methods to reverse the above knowledge. Such targets include data priors [7,43,30] (Fig. 1a), adversarial samples [8,13] (Fig. 1b), and previous memory samples [14,4,5] (Fig. 1c). They have achieved remarkable success, whose trained student models are comparable in performance to those in data-driven KD. Such DFKD methods prescribe a fixed target during the process of DFKD, and finally optimize a generative model and a student model.

* Corresponding author.
E-mail addresses: jrlees@zju.edu.cn (J. Li), zhousheng_zju@zju.edu.cn (S. Zhou), liangcheng_li@zju.edu.cn (L. Li), haishuai.wang@zju.edu.cn (H. Wang), bjj@zju.edu.cn (J. Bu), yuzhirenzhe@zju.edu.cn (Z. Yu).

https://doi.org/10.1016/j.ins.2023.119202
Received 10 February 2023; Received in revised form 17 May 2023; Accepted 20 May 2023

Available online 24 May 2023
0020-0255/© 2023 Elsevier Inc. All rights reserved.



(a) Prior-based methods. (b) Boundary-based methods. (c) CMI[14]. (d) CuDFKD(Ours).

Fig. 1. Differences between CuDFKD and other methods. Black lines represent the decision boundary by the teacher model, and the dots with different shapes represent pseudo samples with different classes. a) Prior-based methods learn the generation target at the space of data prior. b) Boundary-based methods learn pseudo samples close to the decision boundary among classes. c) Memory-based methods like CMI [14] use the previously generated data samples as the generation target. d) Our CuDFKD learns a dynamic generation target from data prior to the decision boundary with time. Better viewed in color.

However, these fixed target settings of DFKD may be more challenging for the student at early learning process. We believe that a more productive method should adjust the training and generation targets with the status of the student model. This approach emulates human learning, where teachers modify their teaching strategies dynamically based on the student's learning ability. For instance, humans use a dynamic procedure to learn easy knowledge like arithmetic before hard knowledge like calculus, while a static approach is employed to learn both simultaneously as aforementioned DFKD methods do. Obviously, the former learning strategy is better for the student of the better learning experience and efficiency. Back to the field of machine learning, such dynamic target strategy is consistent with curriculum learning (CL) techniques [3,38,32]. It first learns the easiest part of the training data, followed by more difficult part, which is a **dynamic** target setting. It promises faster convergence rate and better local minima for the machine learning problem.

To propose a dynamic DFKD method, we present a brand new dynamic learning approach for DFKD based on Curriculum Learning in this paper, called **Cu**rriculum **DFKD** (CuDFKD). We introduce a dynamic and adaptive strategy for generating pseudo samples in CuDFKD, based on an easy-to-hard approach. Our method allows the learning process to adapt to the student's capability. The difficulty for different training timestamps in DFKD is defined as the divergence between the teacher and student models. We generate pseudo samples from data prior to decision boundaries, which serves as a dynamic signal indicating the level of difficulty. Fig. 1d illustrates the main idea of CuDFKD. To update the student model and generator parameters, we propose an alternative updating strategy (AOS) that is simple yet effective. Theoretical support for CuDFKD comes from self-paced learning (SPL) [38]. In practice, the dynamic modules are implemented by adjusting the gradient of adversarial samples and reweighting the generated pseudo samples in the objective function. Experimental results demonstrate that CuDFKD performs comparably to other state-of-the-art (SOTA) DFKD methods, with the added advantage of superior convergence.

The contributions of this paper are summarized as follows:

- We propose a simple but effective DFKD method by dynamically adjusting the difficulty of generated pseudo samples, called CuDFKD. It's the first method to dynamically match the target generation with the status of student model.
- We demonstrate the convergence of CuDFKD by the theory of MM and VC-dimension. We also provide a detailed analysis and discussion of the advantages of CuDFKD over memory-based and boundary-based DFKD methods.
- We first utilize two more metrics to measure the performance of DFKD. Besides, Our CuDFKD performs best in CIFAR10, CIFAR100, Tiny ImageNet over various teacher-student pairs and metrics.
- The designed dynamic module in CuDFKD is proved to be effective by the best performance for both KD training and pseudo sample generation. Notably, our proposed CuDFKD method has the fastest convergence over other DFKD methods. It preserves the best performance even in the presence of noisy teachers.

This paper is organized as follows: section 2 reviews some related work about DFKD and curriculum learning, and section 3 discusses some basic concepts about curriculum learning and self-paced learning. Section 4 outlines how to build up CuDFKD and proves its convergence by MM theory. Section 5 reports some results and discussion about CuDFKD. Finally, section 6 concludes this paper and discusses future work.

2. Related work

This section reviews the previous work of DFKD and curriculum learning and compares their differences with CuDFKD.

2.1. Data-free knowledge distillation

Data-Free Knowledge Distillation (DFKD) generates pseudo samples to optimize student model. According to the kind of target during generation pseudo samples, we categorize DFKD methods into three types, i.e., prior-based DFKD, boundary-based DFKD, and memory-based DFKD.

Prior-based DFKD methods [7,43,29,39,30] imitate the original data distribution using inversion methods. ZSKD [29,40,27] and SoftTarget [39] model the output label distribution or intermediate feature maps with simple distributions. DAFL [7], DeepInversion, and Adaptive DeepInversion [43] use different regularization terms, such as adversarial divergence or batch normalization statistics, to generate more realistic and useful pseudo data samples. Additionally, some limited-data KD models use similar targets to guide the training stage, such as [11].

Boundary-based DFKD methods [13,43,8,9,35] are motivated by the thought of adversarial learning, where generative models are optimized to maximize the gap between the output distribution of teacher and student models. [35,46] treat the pretrained teacher model as a black-box model and use a min-max game to update the generator and student models. DFQ [9] dynamically balances the sample generation between prior-based and boundary-based approaches and incorporates instance/category entropy loss for prior regularization. Qimera [8] explores the effect of boundary samples on model quantization, providing a similar perspective to KD. They learn samples near the teacher model's decision boundary, treating them as hard samples. Such methods generally produce robust student models and high performance.

During the training process, [5] points out that catastrophic forgetting may occur, whereby some learned knowledge or gradients could not be preserved, leading to the student model getting trapped in local minima. Memory-based DFKD methods are proposed to rephrase the learned knowledge from the early student model. CMI [14], PRE-DFKD [4], and MB-DFKD [5] are motivated by continual learning [25]. They use memory samples to avoid the catastrophic forgetting during KD training, achieving outstanding performance on different benchmarks. They preserve the knowledge from the early student model by setting up a memory bank [14, 5] or reconstruction [5] process, thereby assisting the learning of student model. CuDFKD, on the other hand, only uses a non-parameterized training scheduler and difficulty measurer, eliminating extra memory usage or parameter training.

DFKD can also be extended to graph data as "graph-free distillation" [10,36], which is applicable in tasks such object detection [2] and image super-resolution [48]. Additionally, some work [12] proposes faster DFKD methods by meta-learning.

2.2. Curriculum learning

Curriculum learning (CL) has been widely used to train models by an easy-to-hard strategy [3,38,32]. Bengio [3] provides a clear illustration of its convergence of it.

Automatic CL is a dynamic strategy to adjust the difficulty by the feedback of training process. Self-paced learning (SPL) [19] is the most widely used automatic CL method assigning data with different difficulties based on the training losses at each timestamp (or epoch). Several theoretical studies [24,26] provide a deep understanding of SPL and are categorized into majorization minimization (MM) [6] algorithm and concave optimization. They use transfer learning [32] and uncertainty [50] as specific representative techniques to transfer from teacher. Recently, SPL is also applied to the field of unsupervised learning [22], clustering [47], anomaly detection [44] and graph [15].

Besides, some data-driven KD methods also use CL to enhance student learning from the teacher. For example, Xiang et al. [42] use SPL for instance selection in long-tailed datasets, and Li et al. [21] use a similar uncertainty curriculum to distill models from large pretrained language models. In this work, CL provides an adaptive training target for the generation process of DFKD. Additionally, it provides a theoretical understanding of accelerating the convergence of DFKD methods and contributes to the usage of CL in knowledge distillation.

3. Preliminary on CL

Curriculum Learning (CL) is widely used in the data-driven machine learning, i.e., given a training dataset $\mathcal{D} = \{(\mathbf{x}, y)\}$, where y is the label of specific data $\mathbf{x}$. CL defines a difficulty scheduler $d(\mathbf{x}, t)$ to evaluate the difficulty of data samples and a training scheduler to find the subset $\mathbb{B}_t \in \mathcal{D}$ of training data at this difficulty. CL is designed to get the optimal parameter $\mathbf{w}^*$ by some schedulers, and the data subset series is organized to be easy-to-hard. If the difficulty scheduler function is continuous, the subset series should satisfy $\partial d(\mathbb{B}_i, t)/\partial t \geq 0$.

Instead of manually setting two schedulers, self-paced learning (SPL) [19] dynamically update difficulty by the training loss $\mathbf{L}(\mathbf{x}, y; \Theta)$. What's more, they define a reweighting factor $\mathbf{v}$ as a training scheduler, i.e.,

$$\min_{\mathbf{w}\in\mathbb{R}^d, \mathbf{v}\in[0,1]^d} \mathbf{F}(\mathbf{w}, \mathbf{v}) = \mathbf{v}(\lambda, \mathbf{L})^T \mathbf{L}(\mathbf{x}, y; \Theta) + g(\lambda, \mathbf{v}), \tag{1}$$

where the convex function $g(\lambda, \mathbf{v})$ is called a Self-Paced regularizer (SP-regularizer). It's an alternative optimization problem for parameter $\mathbf{w}$ and reweighting factor $\mathbf{v}$. Here $g(\lambda, \mathbf{v})$ is designed under some constraints. It can be explicit defined by primary functions [38,32].

4. Design and discussion of CuDFKD

In this section, we implement the SPL framework to DFKD and propose a dynamic DFKD algorithm to update all the parameters, called **CuDFKD**. To better design the dynamic module of CuDFKD, there still exist some gaps, including:

- **Q1:** How to define the loss function of the generative model $\mathbf{L}_\theta$ to set dynamic generation targets?
- **Q2:** How to design the loss function for KD learning to make the DFKD more adaptive to the student model?

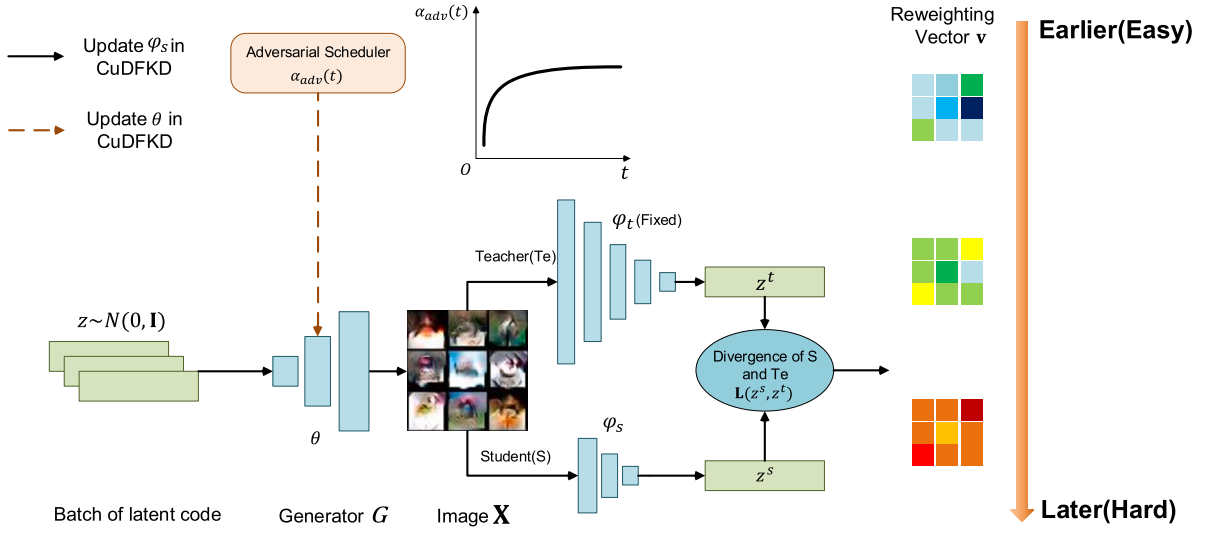


Fig. 2. The mainstream of CuDFKD. We construct a dynamic DFKD generation and training target by 1) an adversarial scheduler $\alpha_{adv}(\tau)$ to adjust the gradient of the generator G, and 2) reweighting factor $\mathbf{v}$ to reweight the generated pseudo samples. For $\mathbf{v}$, the warmer the colors are, the higher value of $\mathbf{v}$ is. Better viewed in screen.

In this work, we present an alternative updating strategy (AOS) that aims to update all parameters of CuDFKD more effectively. Specifically, we alternately train the generator θ and student model ϕ_s in each epoch. Fig. 2 provides a visual representation of the overall workflow of our approach.

4.1. Dynamic generation module

In data-driven KD, the teacher plays a crucial role in ensuring a well student [37,1], and this becomes even more critical in DFKD. For **Q1**, when updating θ, we construct a dynamic generation module, with an objective function for generating pseudo samples motivated by previous DFKD methods [9,14,43], namely,

$$\min_{\theta} \mathbf{L}_{\theta} = \mathbb{E}_{x \sim p_{\theta}}[-\alpha_{adv}(\tau)\mathcal{L}_{adv}(x) + \mathcal{L}_{\theta}(x)]. \tag{2}$$

Here $\mathcal{L}_{adv}(x) = D(f_{\phi_t}(x), f_{\phi_s}(x))$, and divergence function D is either Jensen–Shannon divergence (JS divergence) or Kullback–Leibler divergence (KL divergenece). $\mathcal{L}_{\theta}(x)$ is a loss function for regularizing the pretrained teacher model and improving the quality for the generation of pseudo samples. $\mathcal{L}_{\theta}(x)$ includes categorical entropy [9], and batch normalization statistics alignment [43]. The regularization item L_{θ} is designed by,

$$\begin{aligned} \mathcal{L}_{\theta}(x) &= \alpha_{bn}\mathcal{L}_{bn} + \alpha_{oh}\mathcal{L}_{oh} \\ &= \alpha_{bn}\sum_{l}(\mathcal{D}(\mu_{(l)}, \mu_{bn,(l)}) + \mathcal{D}(\sigma^2_{(l)}, \sigma^2_{bn,(l)})) \\ &+ \alpha_{oh}\sum_{i=1}^{C} CE(\mathbf{z}_i^t, \arg\max_j z_{i,j}^t). \end{aligned} \tag{3}$$

The objective of Equ. (3) is two-fold. The first term aims to align the intermediate BN layer statistics of the pseudo sample with those of real data, thereby ensuring statistical consistency between the two distributions. Besides, the second term $CE(\mathbf{z}_i^t, \arg\max_j z_{i,j}^t)$ serves as a cross-entropy loss between the teacher output logit $\mathbf{z}_i^t$ and the pseudo label $y_i = \arg\max_j z_{i,j}^t$, which balances the class distribution in the output space of the teacher model. This objective, represented by $\mathcal{L}_{\theta}$, facilitates the alignment of the generated distribution with the original data distribution p_{data}.

Compared with previous DFKD methods [43,9,14], our approach adjusts the difficulty of generated samples by tuning the hyper-parameter α_{adv}. Specifically, by setting $\alpha_{adv}(\tau) = 0$, we utilize only the data-prior term $\mathcal{L}_{\theta}(x)$ in Equ. (2) as the generation target, while setting $\alpha_{adv}(\tau) = \infty$ results in the sole use of the adversarial term in Equ. (2). In CuDFKD, we design the generation target as shown in Equ. (15) to dynamically balance the data-prior and decision boundary. To ensure non-decreasing difficulty during training, we employ a predefined function $\alpha_{adv}(\tau)$, whose detailed design are discussed in Section 5.6. The adjustment of the difficulty gradient is illustrated as orange dashed arrows in Fig. 2, thereby ensuring a dynamic generation target.

4.2. Dynamic training module

In this section, we design a dynamic training module of DFKD, specifically in relation to **Q2**. Therefore, we adjust the difficulty when updating ϕ_s in DFKD. As discussed in section 2.2, automatic curriculum learning (CL) is a technique that dynamically adjusts the

difficulty of the training process, thereby enhancing both training speed and convergence. By implementing CL through appropriate pacing functions or transferring from the teacher, we can further improve the performance of DFKD. To achieve this, the pretrained teacher model serves as a better scheduler for the guidance of DFKD. This approach improves the effectiveness of DFKD by ensuring that the difficulty of DFKD process is adapted in a dynamic and efficient manner.

First of all, we construct a difficulty measurer for the above dynamic information. Compared with previous CL methods [3], the dataset D is not available. Therefore, we build a difficulty measurer $d(p_\theta(x), \tau)$ with respect to the built distribution estimator $p_\theta(x)$ and timestamp τ. It should satisfy the following criteria:

- It's monotonically non-decreasing w.r.t. timestamp τ, i.e., $\frac{\partial d(p_\theta(x),\tau)}{\partial \tau} \geq 0$, which promises a easy-to-hard learning strategy.
- It guides the training of DFKD, i.e., to optimize a proper parameter by $\theta^* = \arg\min_\theta f(d(p_\theta(x), \tau), \mathbf{v}^*)$, where f is a type of loss function.

Inspired by boundary-based methods [8,9], we consider pseudo samples close to the decision boundary as hard samples. Therefore, we define a tractable difficulty scheduler for a given dataset $\mathcal{D}$ as $d(\mathcal{D}, t) = \mathbb{E}_{x\sim\mathcal{D}}[D(p_{\phi_t}(x)||p_{\phi_s}(x))]$, where ϕ_t and ϕ_s denote the parameters of the teacher and student models, respectively. Here D is used to minimize the objective function $\mathbf{F}(\mathbf{w}, \mathbf{v})$ in the DFKD method. In the setting of DFKD, the dataset D represents the distribution of the trained dynamic generation module. The dynamic strategy $\mathbf{v}^*$ is then calculated to control the level of difficulty at a fine-grained level, and it is considered as another dynamic target for the training stage of DFKD. To this end, we define $\lambda(\tau)$ as a linearly increasing function with respect to the epoch τ. The objective function for optimizing $\phi_\mathbf{s}$ can then be expressed as follows:

$$\begin{aligned}
&\min_{\phi_s} \mathbf{F}(\phi_s) \\
&= \mathbf{v}^*(\lambda, \mathbf{L})^{\mathrm{T}} \mathbb{E}_{x\sim p_\theta}[D(\frac{z_t(x)}{T}, \frac{z_s(x;\phi_s)}{T})] + g(\lambda, \mathbf{v}^*) \\
&= \sum_{i=1}^{N} v_i^*(\lambda, l_i) l_{i,\phi_s}(G_\theta(z)) + g(\lambda, v_i^*).
\end{aligned} \tag{4}$$

The logit output at the final layer of the teacher and student models is denoted by z_t and z_s, respectively. The temperature, T, defined in the original KD method [17] is used to soften the output distribution of ϕ_t and ϕ_s. The loss function $\mathbf{L}$ in Equ. (4) is given by $D(\frac{z_t(x)}{T}, \frac{z_s(x;\phi_s)}{T})$, where D can be one of KL divergence, JS divergence, or l_1-loss [9,39,14]. The second line of Equ. (4) represents the implementation of the first line. Therefore, to update ϕ_s, we compute the dynamic difficulty by $\mathbf{v}^*(\lambda, \mathbf{L})^T \mathbb{E}_{x\sim p_\theta}[D(\frac{z_t(x)}{T}, \frac{z_s(x;\phi_s)}{T})]$.

Algorithm 1: CuDFKD.

Input: A pretrained teacher model ϕ_t, Student model ϕ_s, λ-scheduler $\lambda(\tau)$, steps for training θ, n_G, steps for training ϕ_s, n_s
Output: A trained student model ϕ_s.

```
1  τ = 0;
2  while not converge do
3      for i = 1:n_G do
4          α_adv ← α_adv(τ) by Equ. (15);
5          Calculate L_θ by Equ. (2);
6          L_θ.backward();
7      end
8      for t = 1:n_s do
9          λ ← λ(τ); v ← v*(λ, L);
10         Calculate F(φ_s) by Equ. (4);
11         F(φ_s).backward();
12     end
13 end
```

4.3. Algorithm of CuDFKD

This section proposes the alternative updating strategy (AOS) algorithm for CuDFKD. It incorporates self-paced learning with various strategies to make a dynamic learning process. The algorithm is presented in Algorithm 1, where θ and ϕ_s are updated alternatively using equations (3) and (4). Specifically, the generation dynamic module and training dynamic module is represented by lines 4 and 9, respectively.

As discussed in Section 3, Algorithm 1 implements an alternative updating strategy (AOS) for three parameters: the generator θ (lines 5-6), student model ϕ_s (lines 10-11), and reweighting factor $\mathbf{v}$ (line 9). Assuming that these parameters have time costs T_g, T_s, and T_v, respectively, the time cost of an inner loop (lines 3-12) is given by $T_{inner} = O(n_G T_g + n_s(T_s + T_v))$. Furthermore, given that the hyperparameter $\lambda(\tau)$ and reweighting factor $\mathbf{v}$ are predefined primary functions, we assume that T_g and T_s are significantly larger than T_v. Therefore, we can approximate the time cost of an inner loop as $T_{inner} \approx O(n_G T_g + n_s T_s)$, which is similar to the time cost of the simplest DFKD method, DAFL [7]. Furthermore, CuDFKD does not require any additional memory during its implementation.

4.4. Does CuDFKD converge?

To ensure the convergence of the student model ϕ_s, we should first find a suitable parameter θ^* to generate pseudo samples with varying degrees of difficulty, represented by the learned distribution $p_\theta(\mathbf{x})$. The reweighting factor $\mathbf{v}^*$ is then used to fine-tune the difficulty level. Finally, following the approach proposed in [26], we analyze the convergence of CuDFKD using the following proposition:

Proposition 1 *(Convergence of CuDFKD). The objective function $\mathbf{F}(\phi_s, \mathbf{v}, \theta)$ is equal to the Majorization Minimization (MM) step for optimizing the following latent function:*

$$\mathbf{F}_\lambda(\mathbf{L}(\theta, \phi_s)) = \int_0^{\mathbf{L}} \mathbf{v}^*(\lambda, \mathbf{L}) d\mathbf{L}. \tag{5}$$

Thus using AOS promises the convergence of all parameters.

Proof. According to Theorem 1 in [26], and set parameter $\mathbf{w} = (\theta, \phi_s)$, it is proved that

$$\begin{aligned} &\mathbf{F}_\lambda(\mathbf{L}(\mathbf{w})) \le \mathbf{Q}(\mathbf{w}|\mathbf{w}^*) \\ &= F_\lambda(\mathbf{L}(\mathbf{w}^*)) + \mathbf{v}^*(\lambda, \mathbf{L}(\mathbf{w}))[\mathbf{L}(\mathbf{w}) - \mathbf{L}(\mathbf{w}^*)]. \end{aligned} \tag{6}$$

For any $\mathbf{w}^*$ and concave function $\mathbf{F}$. Thus for the *majorization step*, the θ is fixed, and the $\mathbf{v}^*$ is achieved by $\mathbf{v}^*(\lambda, \mathcal{L}(\mathbf{w})) = \arg\min_{\mathbf{v}\in[0,1]^d} \mathbf{F}(\mathbf{w}, \mathbf{v})$, where $\mathbf{F}(\mathbf{w}, \mathbf{v})$ is defined in Equ. (1).

For the *minimization step*, by setting the update step of $\mathbf{w}$ as $\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \mathbf{Q}(\mathbf{w}|\mathbf{w}^k)$, it's calculated by,

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \mathbf{v}^*(\mathbf{L}(\mathbf{w}^k), \lambda)^T \mathbf{L}(\mathbf{w}). \tag{7}$$

As the item loss function for student $\mathbf{L}(\mathbf{w})$ is defined as $\mathbb{E}_{x\sim p_\theta(\mathbf{z})}[D_{KL}(p_{\phi_t}(x)||p_{\phi_s}(x))]$, and $\mathbf{v}^*$ is fixed, it's equal to the objective function of KD. By the previous DFKD work, when setting data or diversity prior to the DFKD model, the AOS strategy promises convergence to optimize $\mathbf{w} = (\theta, \phi_s)$.

According to MM theory [6], the lower bound of the objective function on Equ. (6) is monotonically non-decreasing and thus promises convergence. □

4.5. Does CuDFKD work well?

In this section, we provide a theoretical perspective on whether the easy-to-hard strategy improves the performance of DFKD. While we observed that CuDFKD might not perform as well when distilling small neural networks, we briefly discuss the results. By the VC theory of distillation [37], the error between the teacher and student in CuDFKD is bounded by:

$$R(f_s) - R(f_t) \le O(\frac{|\mathcal{F}_s|_C}{n^\alpha}) + \epsilon_l = \mathcal{L}_{gb,sr}, \tag{8}$$

where $R()$ denotes the error of a specific function, and $f_s \in \mathcal{F}_s, f_t \in \mathcal{F}_t$ are the function set of student and teacher model, respectively. Both of them are related to approaching an unknown target function $f \in \mathcal{F}$. The parameter $\frac{1}{2} \le \alpha \le 1$ is related to the learning rate of the training model, and ϵ_l is the approximation error between the teacher and student functions. n denotes the number of data samples. In the setting of DFKD, the learning rate α is manually set by providing a learning rate lr_s for the KD training.

To avoid the student model being trapped into local minima during the early training, CuDFKD generates easy pseudo samples and utilizes the reweighting factor to adjust the difficulty level. It leads to a smaller value of $\mathcal{L}_{gb,sr}$ at an early stage of training, as the easy-to-hard strategy and the choice of divergence $D(z_t, z_s)$ result in α being close to $\frac{1}{2}$ and ϵ_l being small. As the student model grows, $|\mathcal{F}_s|_C$ also increases, increasing ϵ_l and α, making it possible for the student model to generalize better.

The above discussion presents that while updating ϕ_s, CuDFKD helps the student better approach the teacher. Similarly, the error gap between the teacher and target (ground truth) function is,

$$R(f_t) - R(f_r) \le O\left(\frac{|\mathcal{F}_t|_C}{n^{\alpha_{tr}}}\right) + \epsilon_{tr} = \mathcal{L}_{gb,tr}. \tag{9}$$

The error gap between the target and student is calculated by adding Equ. (9) and Equ. (8), i.e.,

$$R(f_s) - R(f_r) \le \mathcal{L}_{gb,tr} + \mathcal{L}_{gb,st}. \tag{10}$$

The component $\mathcal{L}_{gb,tr}$ solely depends on the generator's training performance. During the generation stage of CuDFKD, the optimal sample for the teacher is identified using the loss $\mathcal{L}_{oh}$, resulting in a low value of $\mathcal{L}_{gb,tr}$ during training. Consequently, this guarantees a lower bound on the sum of $\mathcal{L}_{gb,tr}$ and $\mathcal{L}_{gb,st}$ during training, ultimately leading to improved classification performance of the student model.

Table 1
The DFKD performance on CIFAR10 and different teacher-student pairs. All the results are achieved from our implementation. Here *WRN* is in short of *Wider ResNet*. All chosen performances are the best performance during all runs of training. The metrics are defined in section 5.1. Our CuDFKD model is bolded at the last line.

DataSet	CIFAR10											
Teacher	ResNet34[1]			VGG11			WRN-40-2			WRN-40-2		
Student	ResNet18			ResNet18			WRN-40-1			WRN-16-2		
Metric	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p
Vanilla Teacher	95.70	100.00	1.0000	92.25	100.00	1.0000	94.87	100.00	1.0000	94.87	100.00	1.0000
Vanilla Student	94.23	94.91	0.8276	94.23	92.67	0.7868	91.21	91.42	0.8046	90.42	90.91	0.7998
DAFL [7]	91.45	93.20	0.7686	81.10	82.30	0.6303	81.33	80.21	0.6445	81.55	80.21	0.6445
ZSKT [27]	91.60	93.57	0.7968	89.46	90.24	0.7193	86.07	88.42	0.6998	89.66	91.19	0.7464
ADI [43]	93.26	95.33	0.8478	90.36	93.11	0.8063	87.18	88.53	0.7746	84.50	85.61	0.7279
DFQ [9]	94.61	97.07	0.8492	90.84	93.14	0.7568	91.69	92.16	0.7380	92.01	93.50	0.7826
CMI [14]	94.85	96.46	0.8747	91.13	94.18	0.8213	92.20	93.59	0.8305	92.08	94.01	0.8460
PRE-DFKD [5]	91.65	93.53	0.8209	87.26	89.88	0.7527	86.68	88.13	0.7529	83.57	84.67	0.7146
CuDFKD (Ours)	**95.28**	**98.20**	**0.8915**	**91.61**	**96.00**	**0.8267**	**93.18**	**95.27**	**0.8440**	**92.94**	**95.15**	**0.8477**

[1] The result is implemented by ourselves in the same framework with CMI, by the same hyperparameter as the original paper for the previous DFKD methods. Actually, we cannot achieve the reported result from the original paper in CMI, and we report the highest performance over all trials of each method to promise the fairness of this experiment.

4.6. Relationship with the memory-based DFKD methods

CuDFKD shares similar concept with memory-based DFKD methods, as they both focus on improving the performance of the early student model. In this section, we explore the differences between CuDFKD and memory-based methods such as contrastive distillation (CMI) [14] and robust DFKD (PRE-DFKD) [4]. Typically, memory-based methods use past information during the generation stage to increase diversity and capture the learned information from the early steps. CMI and PRE-DFKD employ additional memory to store the generated pseudo samples from previous epochs, which helps the small model understand better. In contrast, CuDFKD uses dynamic difficulty and self-paced learning to generate pseudo samples that are more adaptable to the current state of the student model.

Some other continual learning methods, like [41,25], utilize regularization items to refine the preceding training information. Such methods assume that deep neural networks are overparameterized, and regularization helps to maintain the learned knowledge while preventing overfitting. While these methods differ from CuDFKD in their approach, they share a similar goal of improving the generalization ability of the student model. The formula is

$$L(\theta) = L_t(\theta) + \frac{\lambda}{2}\sum \Omega(\theta - \theta^*_{t-1})^2, \tag{11}$$

τ represents the current training task, while $\tau - 1$ denotes the task from the previous timestamp. The second item in Equ. (11), $\Omega(\theta - \theta^{t-1})^2$, relates to the preceding training epochs. Equ. (11) bears similarity to Equ. (4), as continual learning is a special case of self-paced learning by setting $v_{ContL}(\lambda, \mathbf{L}) = \mathbf{1}^d$ and $g_{ContL}(\lambda, \mathbf{v}) = \Omega(\theta - \theta^{t-1})^2$. It is worth noting that $g_{ContL}(\lambda, \mathbf{v})$ is convex when Ω is semi-positive definite. In continual learning, the regularizer term g must incorporate parameters from previous epochs, while in self-paced learning, $\mathbf{v}$ and g are defined by the current timestamp, τ.

With the predefined g in continual learning, CuDFKD can adjust the difficulty of samples early on in the training process, which can mitigate forgetting during short times. To enhance performance further, we aim to incorporate more information about early samples, which can better address the forgetting problem.

5. Experiments

In this section, we present an evaluation of the performance and convergence of CuDFKD on various benchmarks and teacher-student pairs. We also conduct an ablation study to verify the effectiveness of the curriculum learning strategy. Additionally, we also explore the advantages of CuDFKD over other DFKD methods from several perspectives.

5.1. Experimental setup

For the benchmarks, we use CIFAR10, CIFAR100 [18], and Tiny ImageNet [20]. The image size for CIFARs is 32×32, and for Tiny ImageNet is 64×64. For the models for distillation, we use ResNet [16], VGG [31], and Wider ResNet [45]. Please refer to Table 1 and 2 for different teacher-student pairs and other baseline methods. We use a generator to parameterize the generation process, i.e., $x = G(z)$. All experiments are implemented on NVIDIA 3090 TI GPUs. For the baselines, we compare SOTA DFKD methods as DAFL [7], ZSKT [27], ADI [43], DFQ [9], CMI [14] and PRE-DFKD [4]. Our implementation is based on the framework provided by the paper from CMI [14]. Please refer to Appendix for detailed hyperparameters, the architecture of generators, and adversarial schedulers. We run all the methods for 250 epochs in CIFAR10, and 300 epochs in CIFAR100 and Tiny ImageNet and record the peak performance during all trainings. Our implementation is presented in https://github.com/ljrprocc/DataFree.

Table 2
The DFKD performance on CIFAR100 and different teacher-student pairs. All results are achieved from our implementation. Here *WRN* is in short of *Wider ResNet*. All chosen performances are the best performance during all runs of training. The metrics are defined in section 5.1. Our CuDFKD model is bolded at the last line.

DataSet	CIFAR100											
Teacher	ResNet34[1]			VGG11			WRN-40-2			WRN-40-2		
Student	ResNet18			ResNet18			WRN-40-1			WRN-16-2		
Metric	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p	Acc@1	Agree@1	L_p
Vanilla Teacher	78.05	100.00	1.0000	71.32	100.00	1.0000	75.83	100.00	1.0000	75.83	100.00	1.0000
Vanilla Student	73.24	76.69	0.5836	73.24	72.12	0.5665	64.87	65.90	0.5160	66.61	67.85	0.5394
DAFL [7]	67.58	74.82	0.5232	64.49	72.82	0.4891	55.06	58.83	0.3847	55.48	58.85	0.3878
ZSKT [27]	56.49	61.61	0.4379	59.83	67.75	0.4935	44.35	46.56	0.3046	50.98	54.74	0.3695
ADI [43]	69.13	75.56	0.5604	68.21	76.79	0.5949	52.25	54.74	0.4029	51.54	54.48	0.3966
DFQ [9]	73.56	82.62	0.6173	69.37	80.16	0.5699	63.62	68.42	0.4366	64.02	68.51	0.4396
CMI [14]	75.16	83.46	0.6551	69.78	78.54	0.6158	66.89	71.69	**0.5525**	66.07	70.01	0.5289
PRE-DFKD [5]	75.63	87.07	0.6574	70.29	85.15	0.6632	55.70	59.01	0.4075	49.54	52.29	0.3538
CuDFKD(Ours)	**75.98**	**87.89**	**0.6815**	**71.22**	**85.85**	**0.6777**	**67.16**	**72.53**	0.5381	**66.26**	**71.44**	**0.5295**

[1] The result is implemented by ourselves in the same framework with CMI, by the same hyperparameter as the original paper for the previous DFKD methods. Actually, we cannot achieve the reported result from the original paper in CMI, and we report the highest performance over all trials of each method to promise the fairness of this experiment. Specifically, CMI [14] and DFQ [9] report about 77% in their paper, but we only get about 73.56%, 75.16% in 300 epochs.

As for the robustness and fidelity of the DFKD framework, we use metrics reported by [33,28], i.e.,

- **Average top-1 accuracy** $Acc@1$. It's widely used for the measurement of the learning performance of student accuracy. It's calculated by

$$Acc@1 = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\arg\max_j z^s_{i,j} = y_i), \tag{12}$$

- **Average top-1 agreement** $Agree@1$ [33]. It's proposed to measure the generalization of the KD framework, and it's calculated by

$$Agree@1 = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y^s_i = y^t_i), \tag{13}$$

  where $y^s_i = \arg\max_j z^s_{i,j}$ and $y^t_i = \arg\max_j z^t_{i,j}$ are the pseudo label of teacher and student output logits, respectively.
- **Average Probability Loyalty** L_p [28]. It's first proposed to measure the performance of model compression of large pretrained language models. The probability loyalty measures the robustness of the learned distribution by the student model. It's calculated by

$$L_p(P\|Q) = 1 - \sqrt{D_{\mathrm{JS}}(P\|Q)}, \tag{14}$$

  where $D_{\mathrm{JS}}(P\|Q)$ is the JS divergence between the distributions of P and Q.

Here, the commonly used metric $Acc@1$ in previous DFKD methods [4,14,13,23] only evaluates the gap between the distribution of student logits $p(z^s)$ and the label distribution $p(y)$. However, as the pretrained teacher model is the sole training target in the DFKD setup (discussed in Section 1), the generalization error of DFKD is determined by the gaps between teacher and student outputs. Therefore, alternative metrics such as $Agree@1$ and L_p can more accurately assess these gaps and provide better performance measurements.

5.2. Results and analysis

In this section, we report the result and convergence of CuDFKD. In addition, we compare the performance and convergence of CuDFKD with previous DFKD methods.

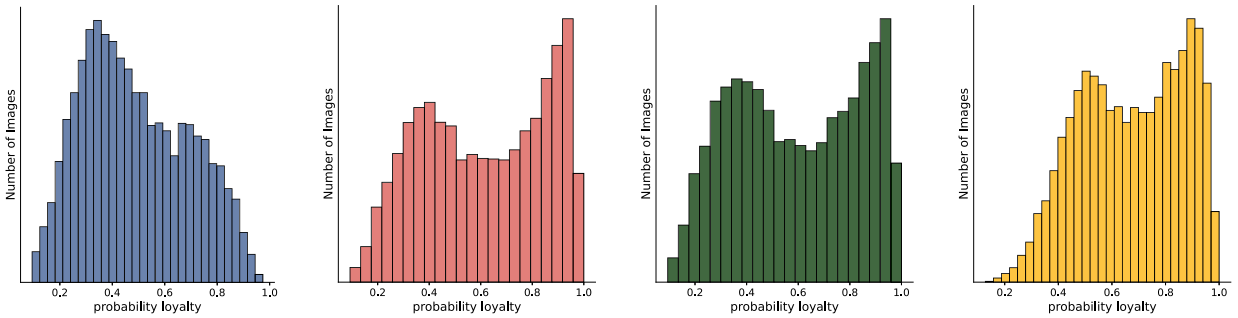
5.2.1. Results on different benchmarks

Table 1 and Table 2 present the performance of different benchmarks and teacher-student pairs. The labels *Vanilla Teacher* and *Vanilla Student* refer to the training accuracy of the teacher and student models, respectively, obtained by training them from scratch using the original training data. The metrics $Agree@1$ and L_p for the teacher model are 100.00% and 1.0000 respectively, according to their definitions. From Table 1, we observe that CuDFKD outperforms all the methods. Furthermore, memory-based methods such as CMI [14] and PRE-DFKD [4] require an extra memory bank or memory generator, while our CuDFKD only adds a scheduler for

Table 3
The result of Tiny ImageNet. All results are achieved by our reimplementation.[1] Methods marked by * mean that the validation performance does not increase for a long time.

Method	Acc@1	Agree@1	L_p
Vanilla Teacher	61.47	100.00	1.0000
Vanilla Student	43.18	44.10	0.3502
CMI*	11.65 ± 1.23	16.72 ± 1.33	0.2003 ± 0.0204
ADI*	26.00 ± 0.87	36.98 ± 0.82	0.2736 ± 0.0185
DFQ	41.30	47.07	0.3300
CuDFKD	**43.42**	**50.07**	**0.3562**

[1] The result is achieved by the same hyperparameter as the CMI original paper in CMI code implementation. As the teacher is retrained in this paper, results are different from those reported in the original paper. For the fairness of two methods, we run 3 experiments and report the error for CMI and ADI.



(a) L_p distribution of DAFL. (b) L_p distribution of CMI. (c) L_p distribution of ADI. (d) L_p distribution of CuDFKD.

Fig. 3. Distribution of probability loyalty L_p for different DFKD methods. T: VGG11, S: ResNet18, Benchmark: CIFAR100. Better viewed in color.

the training. For different teacher-student pairs in CIFAR10, we achieve the best overall performance metrics, as shown in Table 1. It is worth noting that we use the same hyperparameter settings and code implementation as the original paper by CMI.[1]

For harder benchmarks, we also implement CuDFKD on Tiny ImageNet, and the result is presented in Table 3. The larger image resolution of Tiny ImageNet from 32×32 to 64×64 required us to adjust the hyperparameters of the generator models from the original paper and freeze them. Additionally, we adjust the hyperparameters of the dynamic strategy further $\mathbf{v}^*$ and retrain the teacher model on the benchmark before performing CuDFKD. For comparison with other DFKD methods, we used the same generator architecture as CuDFKD. From Table 3, we observe that CMI and ADI did not perform well in our implementation, while CuDFKD maintained stable and optimal performance, even in the face of more challenging benchmarks.

5.2.2. Robustness at the evaluation sets

To evaluate the robustness and fidelity of CuDFKD, we compute two metrics, namely the average Top-1 agreement ($Agree@1$) and the average probability loyalty (L_p). Tables 1 and 2 show the results, from which we make the following observations:

1) Both $Agree@1$ and L_p metrics are positively correlated with $Acc@1$.

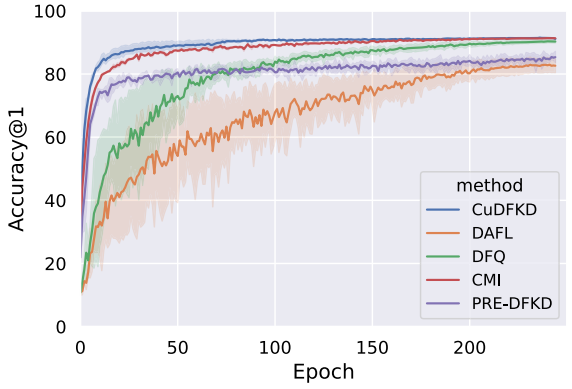
2) Among all the benchmarks and student-teacher pairs, CuDFKD achieves the best performance, which can be attributed to the full learning of the information in the teacher model, i.e., the full imitation of the teacher.

3) Although *Vanilla Student* results in a more desirable $Acc@1$, its $Agree@1$ and L_p values are not satisfactory.

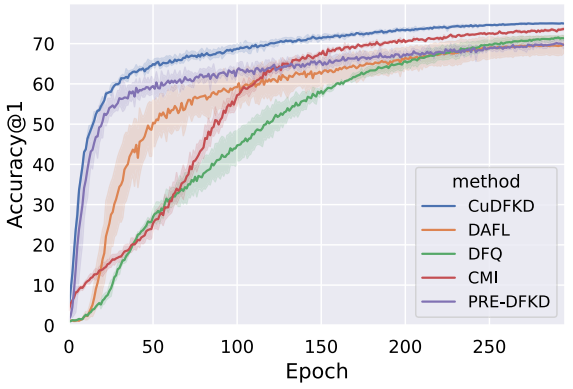
Observation 1) implies that the performance of the student model in the DFKD task relies on the degree of imitation of the teacher model. Observation 2) explains that the main reason for the superior performance of CuDFKD is the adequate learning of the teacher model. Finally, observation 3) indicates that such approximation to teacher is gradually achieved during the training of the student model. The student model approaches the target distribution $p(y)$. By measuring $Agree@1$ and L_p, we emphasize the importance of approximating the distribution determined by the teacher model for DFKD training and thus present the superiority of our CuDFKD over previous methods.

To further investigate the generalization performance of DFKDs, we visualized the distribution of L_p at the final epoch of ADI, CMI, and CuDFKD using VGG11-ResNet18 pairs on CIFAR100. The resulting distribution, as shown in Fig. 3, demonstrates that CuDFKD achieves higher probabilistic loyalty compared to ADI, DAFL, and CMI on the CIFAR100 benchmark, with a clear rightward

[1] https://github.com/zju-vipa/CMI.



(a) The convergence for CIFAR10, the running epochs is 250. T: VGG11, S: ResNet18.



(b) The convergence for CIFAR100, the running epochs is 300. T:ResNet34, S: ResNet18,

Fig. 4. Visualization of validation accuracy curves of different curves with different methods. The blue curve is our CuDFKD, respectively. Better viewed in color.

Table 4
The stability for the training of ResNet18 taught by ResNet34 on two CIFAR benchmarks. Here the μ and σ^2 achieved by CuDFKD are performed by 4 runs. The memory usage and training time tests are all performed on one NVIDIA 3090 GPU, with batch size of 256. Here we use $Acc@1$ as the metric.

DataSet	CIFAR10					CIFAR100				
	μ	σ^2	best Acc@1	GPU Memory	Time	μ	σ^2	best Acc@1	GPU Memory	Time
DAFL	62.6	17.1	92.0	**6.45G**	6.10 h	52.5	12.8	74.5	**6.45G**	**7.09 h**
DFAD [13]	86.1	12.3	93.3	-	-	54.9	12.9	67.7	-	-
ADI [43]	87.2	13.9	93.3	7.85G	25.2 h	51.3	18.2	61.3	7.85G	30.4 h
CMI [14]	82.4	16.6	94.8	12.5G	13.3 h	55.2	24.1	77.0	12.5G	22.3 h
MB-DFKD [5]	83.3	16.4	92.4	-	-	64.4	18.3	75.4	-	-
PRE-DFKD [4]	87.4	10.3	94.1	-	-	70.2	11.1	**77.1**	-	-
CuDFKD(Ours)	**94.1**	**2.88**	**95.0**	6.84G	**5.48 h**	**71.7**	**4.37**	75.2	6.84G	7.50 h

shift of the first wave peak. However, CMI generates more low L_p samples than higher one, leading to a lower mean L_p. It is expected since ADI and DAFL introduces a regularization term to the generator's loss function L_g to increase the diversity of the pseudo-samples, which trades off similarity to the teachers for the generation of diverse samples.

For CuDFKD, the difficulty function is designed based on the disagreement between teachers and students, and it changes dynamically with the training process. Therefore, it is more inclined towards the imitation and learning process for the teacher, resulting in higher $Agree@1$ and L_p performance. The performance of $Agree@1$ and L_p supports our hypothesis that $\mathcal{L}_{gb,st}$ is low, while high $Acc@1$ supports the hypothesis that $\mathcal{L}_{gb,st} + \mathcal{L}_{gb,tr}$ is low, as shown in Equ. (10). In conclusion, CuDFKD achieves comparable results among SOTA methods because it better explores the global minimum during the easy-to-hard training strategy. Note that these results were achieved at the best checkpoint during training, which we refer to as the **peak performance**. We also evaluated the convergence and performance of CuDFKD during the training stage.

5.2.3. *Convergence of CuDFKD*

In this section, we check the convergence of CuDFKD and compare it with other SOTA DFKD methods by plotting the curve of validation accuracy at each timestamp during training. We use VGG11-ResNet18 and ResNet34-ResNet18 as teacher-student pairs. Hyperparameters are those specified in the original papers of each method. The metric used for this experiment is $Acc@1$, and the same curriculum strategy as in Table 1 is used. Fig. 4 show that CuDFKD converges quickly and efficiently, achieving a 90% top-1 accuracy on CIFAR10 in less than 15 epochs and stably improving until the end of training. Moreover, the training process is stable, as the validation accuracy does not fluctuate significantly during the training process. In contrast, previous DFKD methods exhibit different stability, making it difficult to achieve optimal performance.

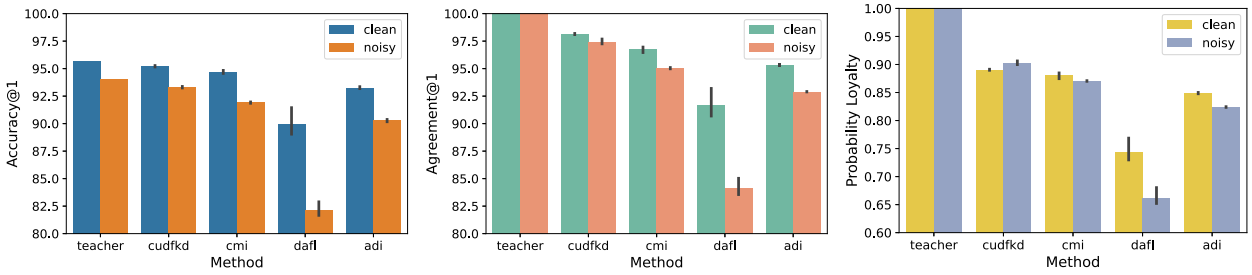
Quantitatively, we assess the stability of our framework and compare it with other SOTA DFKD methods by computing the mean $\mu[s_{acc}]$ and variance $\sigma^2[s_{acc}]$ of the validation student accuracy over all epochs, as motivated by PRE-DFKD [4]. Specifically, we calculate $\mu[s_{acc}]$ and $\sigma^2[s_{acc}]$ as the mean and variance of the validation accuracy from the 80th epoch to the end of the training, as all methods tend to converge from this epoch. Moreover, we record the peak performance during training, the used GPU memory, and training time. Table 4 presents the $\mu[s_{acc}]$ and $\sigma^2[s_{acc}]$ for all SOTA methods. Our CuDFKD shows well stability during DFKD training without requiring any memory bank or replay generators, achieving higher μ than all other methods. Furthermore, the memory usage and time cost indicate that CuDFKD uses the same memory as the simplest DFKD method as DAFL while achieving significantly better performance.

(a) Distribution of difficulty at different timestamps.

(b) Visualization of generated samples by the class at epoch 10.

(c) Visualization of generated samples by the class at epoch 300.

Fig. 5. Visualization of the distribution of difficulty at different stages while training CuDFKD. T: ResNet34, S: ResNet18, Benchmark: CIFAR100. Better viewed in color.



(a) $Acc@1$.

(b) $Agree@1$.

(c) L_p.

Fig. 6. Performance of using clean and noisy teacher for training CuDFKD on different metrics. T: ResNet34, S: ResNet18, Benchmark: CIFAR10. Better viewed in color.

5.3. Difficulty visualization at different stages

In CuDFKD, we utilize the divergence between the teacher and student models to define the difficulty level for each training sample. It is important to ensure that the easy-to-hard criterion is satisfied during training. To investigate the impact of this design on student model training, we visualize the sample difficulty distribution at different training stages, along with the corresponding images. We focus on visualizing image $\mathbf{X}$ as the difficulty, consistent with the definition of SPL. We also compute the difficulty of the generated samples by finding the label y_t with the highest score: $y_t = \arg\max_i z_i^t$. The experiments are conducted on CIFAR100, with WRN-40-2 as the teacher and WRN-16-2 as the student, using a batch size of 1024.

The visualized samples are presented in Fig. 5. Fig. 5a shows the visualization results of the difficulty distribution for the 10th epoch (distribution B) as well as for the 300th epoch (distribution A). Figs. 5b and 5c show the visualization results of the same category of pseudo-samples for the 10th epoch (Fig. 5b) as well as for the 300th epoch (Fig. 5c). We observe sharper peaks in distribution A at the beginning of training compared to distribution B, and the peaks in A are more to the left than those in B. These sharp peaks indicate that the samples are not sufficiently diverse at the beginning of the training phase, and therefore the visualization is not as effective. Similar results are obtained from the sample visualization results in Figs. 5b and 5c. For Fig. 5b, the samples between different classes already possess a certain gap, however, the intra-class samples are more similar. This setting for the samples can effectively help students learn quickly from the teacher model and achieve the learning process from easy to difficult, as explained in Section 1 and Fig. 2.

5.4. Comparable results on noisy teachers

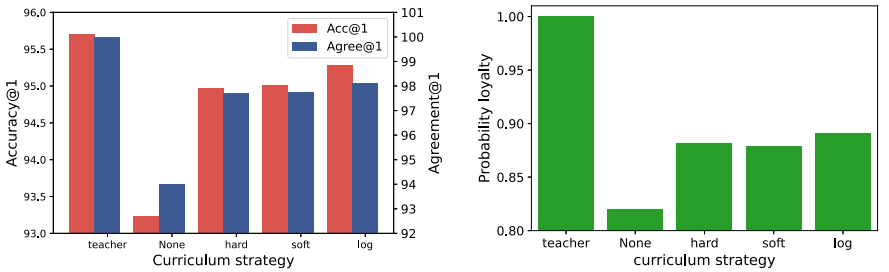
In this section, we investigate the robustness of CuDFKD in learning from a noisy teacher model. They create the noisy teacher model by training a ResNet34 model for only 200 epochs with an initial learning rate of 1.0 using a cosine scheduler and without finetuning other parameters. The teacher model achieves 94.07% top-1 accuracy but contains some incorrect classification information, which introduces noise into the distillation process. The experiments are conducted with a batch size of 768, and the results are presented in Fig. 6. In addition, we run three experiments with random seeds $\{0, 10, 20\}$ and report the error bars in the figure for all three metrics.

From Fig. 6, our CuDFKD model achieves more than 93.3% $Acc@1$ with the guidance of the noisy teacher, experiencing less than a 0.7 percentage drop compared to the teacher model. Moreover, CuDFKD improves the L_p of student guided by the noisy teacher, indicating its robustness against noise in the teacher. Such robustness is attributed to the dynamic reweighting of generated pseudo samples, which provides another form of supervision for the student.

Additionally, we also have some interesting discoveries from Fig. 6. First, DAFL suffers significant drops in all metrics due to its failure to learn adversarial samples, resulting in its lack of robustness. Second, CuDFKD and CMI perform better than ADI and DAFL in terms of the drop in accuracy, as the memory in CMI improves the interaction between the teacher and student. Last, CuDFKD reports less variance than CMI across different random seeds and even better L_P when using the noisy teacher model, indicating that

Table 5
Ablation study of different components of CuDFKD. If adding the reweighting factor $\mathbf{v}$, we mark ✓ in the $\mathbf{v}$ column. The last column t_{90} represents the cost of GPU hours for the student model to reach 90% top-1 validation accuracy. T: ResNet34, S: ResNet18, Benchmark: CIFAR10.

$\mathbf{v}$	$\alpha_{adv}(\tau)$	Acc@1	Agree@1	L_p	t_{90}
	1.0	94.61	97.07	0.8492	3.98 h
✓	1.0	94.86	97.59	0.8805	1.21 h
	Equ (15)	94.79	97.51	0.8806	3.28 h
✓	Equ (15)	**95.28**	**98.20**	**0.8915**	**0.71 h**



(a) $Acc@1$ and $Agree@1$ with different settings of $\mathbf{v}^*$s.

(b) L_p with different settings of $\mathbf{v}^*$s.

Fig. 7. The performance of CuDFKD with different settings of dynamic strategy. Teacher: ResNet34, Student: ResNet18, Benchmark: CIFAR10. Better viewed in color.

a good dynamic strategy can improve the efficiency of knowledge transfer and align the output distribution from the teacher and student models. Therefore, training DFKD on the noisy teacher model presents an interesting area for future research.

5.5. Ablation study on dynamic modules

This section investigates the impact of different dynamic modules in CuDFKD. As discussed in section 4, CuDFKD employs a dynamic learning strategy in the generation and training stages. Specifically, we adjust the hyperparameter α_{adv} to control the generation target and the reweighting factor $\mathbf{v}$ to maintain the target of KD. Table 5 presents the performance of CuDFKD with and without different dynamic modules. To assess the convergence rate, we measure the time it takes for the student model to achieve 90% validation $Acc@1$, denoted as t_{90}.

Table 5 presents the performance of CuDFKD with different dynamic modules. Compared with DFQ [9] (line 1), adding the reweighting factor $\mathbf{v}$ (line 2) leads to significant improvements in $Agree@1$ and L_p. It presents that t$\mathbf{v}$ helps the student model approach the teacher model. Furthermore, the convergence rate of DFKD significantly improves as t_{90} decreases from nearly 4 GPU hours to 1.21 GPU hours. Using the only dynamic generation target by adjusting α_{adv} (line 3) results in significant improvements in $Agree@1$ and L_p but with a slight decrease in t_{90}. Finally, combining both dynamic modules (line 4) leads to substantial improvements in all metrics, with an even greater decrease in t_{90}. We conclude that the dynamic generation target by $\alpha_{adv}(\tau)$ can help the reweighting factor $\mathbf{v}$ find the global minima of the KD loss function $\mathbf{L}$ faster. Therefore, both designed dynamic modules are crucial for CuDFKD's performance.

5.6. Hyperparmeter sensitivity

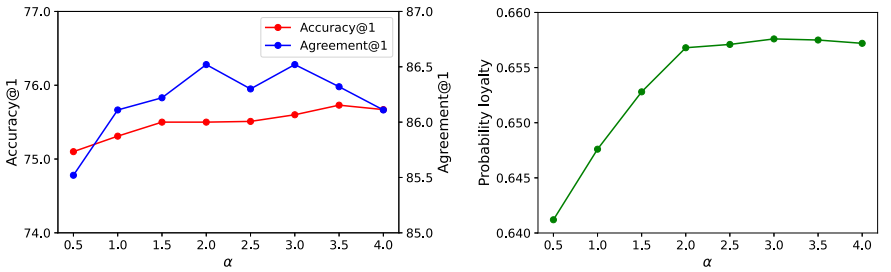
In this section, we check the effectiveness of CuDFKD and compare the performance of different hyperparameters for a given design of $\mathbf{v}^*$ and $\mathcal{L}_\theta(x)$. Furthermore, we explore various hyperparameters for the dynamic modules at different stages.

Effect of dynamic training module, $\mathbf{v}^*$. Specifically, we check different designs of $\mathbf{v}^*$. As it has the analytical solution, we design different closed forms of $\mathbf{v}^*$ s by previous work [38], i.e.,

- **Hard**: $\mathbf{v}^*(\lambda, \mathbf{L}) = \mathbf{1}(\mathbf{L} < \lambda)$;
- **Soft**: $\mathbf{v}^*(\lambda, \mathbf{L}) = \mathbf{1}(\mathbf{L} < \lambda)(1 - \mathbf{L}/\lambda)$;
- **Logarithm**: $\mathbf{v}^*(\lambda, \mathbf{L}) = \frac{1+e^{-\lambda}}{1+e^{\mathbf{L}-\lambda}}$.

The function $\mathbf{1}()$ represents the indicator function. The remaining hyperparameters are kept consistent with those used in previous experiments. All experiments are implemented on the distillation ResNet34-ResNet18 in CIFAR10, and they're summarized in Fig. 7.

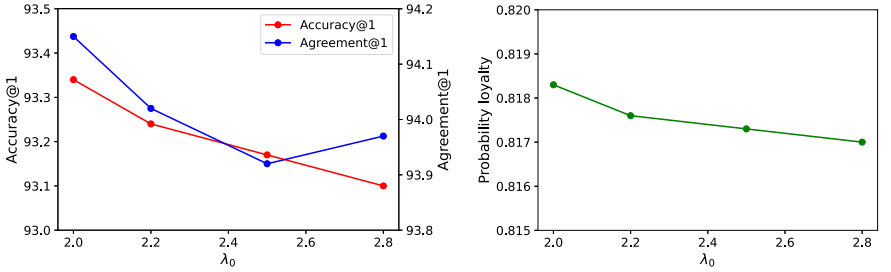
The findings meet with those of prior studies on curriculum learning [38]. As illustrated in Fig. 7, we observe the following: 1) Regardless of the dynamic strategy employed, the $Acc@1$ of the student models shows improvements. 2) Notably, adopting a softer dynamic strategy, such as the logarithmic scheduler, significantly improves learning performance. It's attributed to the logarithmic



(a) $Acc@1$ and $Agree@1$ with different αs.

(b) L_p with different settings of αs.

Fig. 8. The performance of CuDFKD with different settings of α. Teacher: ResNet34, Student: ResNet18, Benchmark: CIFAR100. Better viewed in color.



(a) $Acc@1$ and $Agree@1$ with different λ_0s.

(b) L_p with different settings of λ_0.

Fig. 9. The performance of CuDFKD with different settings of λ_0. Teacher: Noisy ResNet34, Student: ResNet18, Benchmark: CIFAR10. Better viewed in color.

scheduler's ability to circumvent hard threshold-like indicator functions, which, in turn, provide a continuous and differentiable optimization objective for the student network. Such a continuous setting enables more efficient discovery of global minimum solutions.

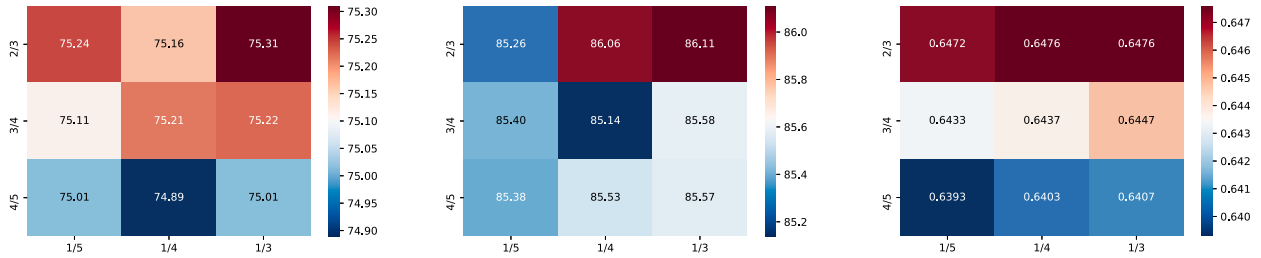
Effect of λ_0. We investigate the effect of hyperparameter $\lambda(\tau)$ on the actual dynamic strategy here as discussed in Section 4. Choosing a proper $\lambda_0 = \lambda(0)$ is crucial as it affects the initial reweighting of generated pseudo-samples and early student training. Specifically, a large λ_0 leads to overreweighting, where almost all the generated samples are included in the training. Besides, the SP-regularizer $g(\mathbf{v};\lambda)$ is close to zero, resulting in a collapse to the same training process as previous DFKD methods. On the other hand, a small λ_0 leads to underreweighting, where almost no samples are included in the training. Most of the contribution to the objective function $\mathbf{F}$ is from $g(\mathbf{v};\lambda)$, causing no optimization of the student model.

Practically, we set λ_0 from $\{2.0, 2.2, 2.5, 2.8\}$, as choosing a value less than 2.0 results in underreweighting and divergence, while a value greater than 3.0 results in overreweighting. The experiments use the logarithm strategy for $\mathbf{v}^*$, with a noisy ResNet34 as the teacher and ResNet18 as the student. The noisy teacher model is used because we solely explore the hyperparameter sensitivity of the dynamic module. The results in Fig. 9 indicate that λ_0 is slightly negatively correlated with the final performances. Besides, all the performances achieve more than 93% $Acc@1$, which is better than other SOTA methods. For the $Agree@1$ and probability loyalty L_p metrics, a similar phenomenon is observed as $Acc@1$. Obviously, using a proper interval of λ_0 leads to ideal performances for DFKD.

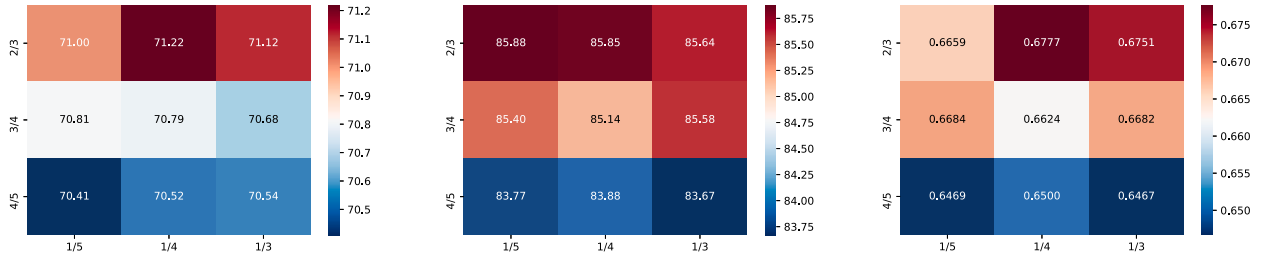
Effect of dynamic strategy of generation stage, α_{adv}. The preceding section deals with the training stage, which involves optimizing ϕ_s. To investigate the hyperparameter sensitivity in the generation stage, we focus on the hyperparameter α_{adv}, which regulates the gradient for the divergence to the weight of generator θ. α_{adv} plays a crucial role in dynamically adjusting the difficulty level during the generation stage, as outlined in Section 4.1. Our experiments are conducted on the CIFAR100 benchmark using VGG11-ResNet18, ResNet34-ResNet18, and WRN-40-2-WRN-40-1 pairs. We set the batch size to 768 to obtain better results and employed a simple scheduler for the experiments.

$$\alpha_{adv}(\tau) = \begin{cases} 0, & \tau \le k_{begin} N \\ \alpha \cdot \tau, & k_{begin} N < \tau \le k_{end} N \\ \lambda_{adv,final}, & \tau > k_{end} N. \end{cases} \tag{15}$$

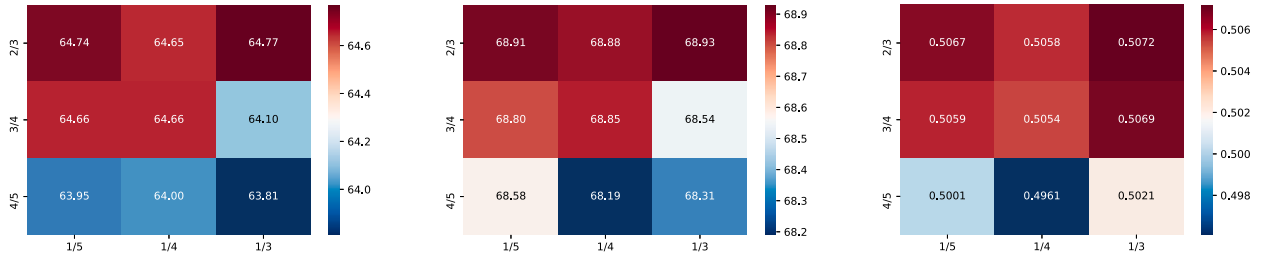
In previous work [4], epoch 0 to $k_{begin}N$ referred to as the warm-up stage. Practically, we set $k_{end} < 1$ to avoid extra noise to the pseudo samples. We search for hyperparameters by adjusting $k_{begin} \in \{1/5, 1/4, 1/3\}$ and $k_{end} \in \{2/3, 3/4, 4/5\}$, where N is the total number of training epochs. The results are presented in Fig. 10. The figure shows that k_{begin} is more sensitive than k_{end}, and all the best performances are obtained when $k_{begin} = 1/5$. In lighter networks like the wider ResNet series, the change in $Acc@1$ is



(a) Acc@1, Res34 → Res18. (b) Agree@1, Res34 → Res18. (c) L_p, Res34 → Res18.



(d) Acc@1, VGG11 → Res18. (e) Agree@1, VGG11 → Res18. (f) L_p, VGG11 → Res18.



(g) Acc@1, WRN-40-2 → WRN-16-2. (h) Agree@1, WRN-40-2 → WRN-16-2. (i) L_p, WRN-40-2 → WRN-16-2.

Fig. 10. The effect of k_{begin} (x-axis) and k_{end} (y-axis) on the performance of DFKD. All the experiments are performed on CIFAR100. The warmer the color is, the higher the metric is and thus the better performance is. Better viewed in color.

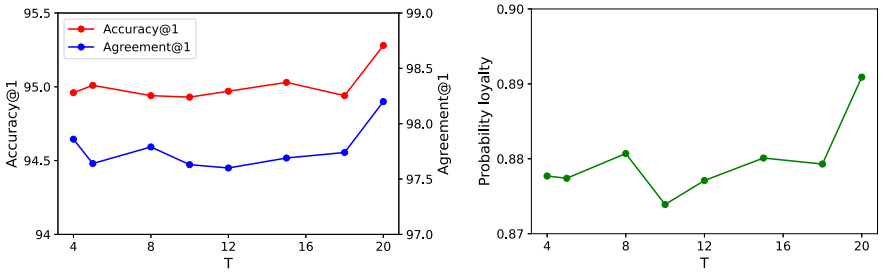
Table 6
Ablation study of $\alpha_{adv}(\tau)$. ∞ represents we do not add $\mathcal{L}_\theta(x)$ in Equ. (2). T: ResNet34, S: ResNet18, Benchmark: CIFAR10.

$\alpha_{adv}(\tau)$	Acc@1	Agree@1	L_p
0	93.63	95.98	0.8462
∞	94.92	97.64	0.8823
Equ (15)	**95.28**	**98.20**	**0.8915**

more pronounced than in $Agree@1$ and L_p. It suggests that the training epochs can be treated as warm-up steps in previous DFKD work [9,4], and it provides a prior for tuning the gradient. To test the effect of $\alpha_{adv}(\tau)$, we evaluated CuDFKD with two extreme cases: $\alpha_{adv}(\tau) = 0$ and $\alpha_{adv}(\tau) = \infty$. The results are presented in Table 6. We observe that when setting the generation target as the decision boundary, the performance can be greatly improved (line 2). The dynamic strategy also leads to significant improvement in the student model's performance (line 3).

Gradient of adversarial item α. The gradient parameter α in Equ. (15) plays a important role in controlling the variation of difficulty during generation. It significantly affects the difficulty scheduler in CuDFKD. To explore its sensitivity, we conduct experiments on CIFAR100 with α values in the range of [0.05, 0.4] using a step size of 0.05. The results are presented in Fig. 8. As α increases, all three metrics follow a similar pattern of initially increasing and decreasing later. When α is too small, the change in difficulty setting of CuDFKD is insufficient to produce a notable effect, resulting in a performance close to that of static DFKD. Conversely, when α is too large, the generated samples tend to focus on samples near the decision boundary in the later stages of training, making it more challenging to capture image information for classification.

Temperature T **for KD.** Previous KD frameworks used the temperature T to smooth the output distribution of the model and facilitate alignment between teacher and student models across the distribution. However, we observed that different model ar-



(a) $Acc@1$ and $Agree@1$ with different Ts.

(b) L_p with different settings of T.

Fig. 11. The performance of CuDFKD with different settings of temperature T. Teacher: ResNet34, Student: ResNet18, Benchmark: CIFAR10. Better viewed in color.

chitectures require different T s for optimal results. To investigate this further, we conducted several comparison experiments by varying the temperature T and comparing the resulting performance across three metrics. The results are presented in Fig. 11. Our findings indicate that for ResNet and VGG architectures, larger temperature values than those commonly used in data-driven KD (i.e., $T = 4$ [17]) yield better results. Additionally, heavier models require smoother output distributions to achieve optimal KD performance.

6. Conclusion

This paper proposes a novel dynamic Data-Free Knowledge Distillation (DFKD). Specifically, the dynamic module is realized by the thought of self-paced learning (SPL). In contrast to prior DFKD methods, CuDFKD utilizes dynamic difficulty setting, with the timestamp of the training process as target information for generating pseudo samples. By applying SPL, we dynamically adjust the difficulty of the generated samples, which improves the balance between the data-prior and decision boundary and makes the DFKD model more adaptive to the student model. We also analyze the properties of CuDFKD and demonstrate its ability to converge all parameters due to MM theory.

In our experiment, we compare the $Acc@1$, $Agree@1$ and L_p of CuDFKD with several memory-based methods, and present that CuDFKD achieves comparable results. Moreover, CuDFKD's convergence is fast and stable, and it performs the best across a wide range of hyperparameter settings. Overall, we offer a brand new perspective on how to perform DFKD dynamically and highlight the types of knowledge that can improve the training of KD. CuDFKD is a promising approach that provides an effective solution to the challenges posed by previous DFKD methods and has the potential to be applied in various settings.

7. Limitation and future work

Despite its comparable results, CuDFKD has some limitations that leaves for future work. Firstly, the scheduler $\alpha_{adv}(\tau)$ design is currently manual, as we choose the best validation accuracy among all experiments. We plan to develop more automatic strategies to optimize this parameter in the future. Secondly, the curriculum setting of matching $\mathbf{v}^*(\lambda, \mathbf{L})$ and $d(p_\theta(x), \tau)$ can be made more flexible to achieve better performance. Additionally, we aim to explore the integration of sequential deep generative models with CuDFKD to control the difficulty of generated samples at different timestamps. These avenues of research will improve the applicability and effectiveness of CuDFKD.

CRediT authorship contribution statement

Jingru Li: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Sheng Zhou:** Methodology, Supervision, Writing – review & editing. **Liangcheng Li:** Project administration, Resources, Supervision, Writing – review & editing. **Haishuai Wang:** Methodology, Supervision, Writing – review & editing. **Jiajun Bu:** Funding acquisition, Supervision, Writing – review & editing. **Zhi Yu:** Funding acquisition, Supervision, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Table 7
Hyperparameter setting in the experiment.

DataSet	CIFAR10	CIFAR100	Tiny ImageNet
Batch Size	768	768	512
Dim of z	512	512	512
lr_G	0.001	0.001	0.001
lr_s	0.1	0.1	0.1
α_{bn}	1	1	1
α_{oh}	20	20	20
k_{begin}	0.25	0.25	0.33
k_{end}	0.75	0.75	0.67
α	1e-4	2e-5	2e-5
$\frac{\partial\lambda(\tau)}{\partial\tau}$	0.5	0.2	0.2
λ_0	2.0	1.0	1.0
loss type	kl	l_1	l_1
T	20	5	5

Acknowledgements

This work is supported by Alibaba-Zhejiang University Joint Institute of Frontier Technologies, The National Key R&D Program of China (No. 2021YFB2701100), the National Natural Science Foundation of China (No. 61972349, 62106221) and the Fundamental Research Funds for the Central Universities (No. 226-2022-00064).

Appendix A

A.1. Implementation details

In our details for implementation, we design a generator to represent the deep generative model, i.e., $x = G(z)$. The detailed hyperparameter design is in Table 7. We use Pytorch and code forked by CMI [14]. Code can be found in the GitHub link in section 5. Noticing that the hyperparameter selection in Table 7 is the default set of hyperparameters in CuDFKD, we change one of them to some specific value when performing hyperparameter searching in section 5.6.

In this paper's table values, we test different sets of nearby hyperparameters in Table 7 and report the best value in the table.

Implementation on CIFARs. For the experiments of CIFARs, we design a generator motivated by DCGAN and other work by 2 upsampling layers. The structure is: Linear - Upsampling2 - Conv3 - BN - LeakyReLU - Upsampling2 - Conv3 - BN - LeakyReLU - Conv3 - Tanh.

Implementation on Tiny ImageNet. For the experiments of CIFARs, we design a generator motivated by DCGAN and other work by three upsampling layers. The structure is: Linear - Upsampling2 - Conv3 - BN - LeakyReLU - Upsampling2 - Conv3 - BN - LeakyReLU - Conv3 - BN - LeakyReLU - Upsampling2- Conv3 - Tanh.

In the experiments, we find that the number of upsampling layers L greatly affects the final performance of DFKD. When L is too large, the interpolation operation can greatly affect the generation quality. On the other hand, if L is too small, the hidden units for the linear projection will be large and thus lose a lot of information. In CuDFKD, we set $L = 2, 3$ for a better generation of pseudo samples.

For the implementation of previous DFKD methods, we use the single generator in the setting of DFQ and CMI, without any meta-data setting described in [23].

References

[1] D. Bang, J. Lee, H. Shim, Distilling from professors: enhancing the knowledge distillation of teachers, Inf. Sci. 576 (2021) 743–755.
[2] A. Banitalebi-Dehkordi, Knowledge distillation for low-power object detection: a simple technique and its extensions for training compact models using unlabeled data, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 769–778.
[3] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 41–48.
[4] K. Binici, S. Aggarwal, N.T. Pham, K. Leman, T. Mitra, Robust and resource-efficient data-free knowledge distillation by generative pseudo replay, preprint, arXiv:2201.03019, 2022.
[5] K. Binici, N.T. Pham, T. Mitra, K. 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, 2022, pp. 663–671.
[6] R.E. Caflisch, Monte Carlo and quasi-Monte Carlo methods, Acta Numer. 7 (1998) 1–49.
[7] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, Q. Tian, Data-free learning of student networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3514–3522.
[8] K. Choi, D. Hong, N. Park, Y. Kim, J. Lee, Qimera: data-free quantization with synthetic boundary supporting samples, Adv. Neural Inf. Process. Syst. 34 (2021).
[9] Y. Choi, J. Choi, M. El-Khamy, J. Lee, Data-free network quantization with adversarial knowledge distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 710–711.
[10] X. Deng, Z. Zhang, Graph-free knowledge distillation for graph neural networks, preprint, arXiv:2105.07519, 2021.
[11] G. Fang, Y. Bao, J. Song, X. Wang, D. Xie, C. Shen, M. Song, Mosaicking to distill: knowledge distillation from out-of-domain data, Adv. Neural Inf. Process. Syst. 34 (2021) 11920–11932.

[12] G. Fang, K. Mo, X. Wang, J. Song, S. Bei, H. Zhang, M. Song, Up to 100x faster data-free knowledge distillation, in: AAAI Conference on Artificial Intelligence, 2022.
[13] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, M. Song, Data-free adversarial distillation, preprint, arXiv:1912.11006, 2019.
[14] G. Fang, J. Song, X. Wang, C. Shen, X. Wang, M. Song, Contrastive model inversion for data-free knowledge distillation, preprint, arXiv:2105.08584, 2021.
[15] M. Gong, H. Zhou, A. Qin, W. Liu, Z. Zhao, Self-paced co-training of graph neural networks for semi-supervised node classification, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[17] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, preprint, arXiv:1503.02531, 2015.
[18] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[19] M. Kumar, B. Packer, D. Koller, Self-paced learning for latent variable models, Adv. Neural Inf. Process. Syst. 23 (2010).
[20] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231 (7) (2015) 3.
[21] L. Li, Y. Lin, S. Ren, P. Li, J. Zhou, X. Sun, Dynamic knowledge distillation for pre-trained language models, preprint, arXiv:2109.11295, 2021.
[22] W. Li, H. Chen, T. Li, J. Wan, B. Sang, Unsupervised feature selection via self-paced learning and low-redundant regularization, Knowl.-Based Syst. 240 (2022) 108150.
[23] L. Luo, M. Sandler, Z. Lin, A. Zhmoginov, A. Howard, Large-scale generative data-free distillation, preprint, arXiv:2012.05578, 2020.
[24] Z. Ma, S. Liu, D. Meng, Y. Zhang, S. Lo, Z. Han, On convergence properties of implicit self-paced objective, Inf. Sci. 462 (2018) 132–140.
[25] M. Mazur, Ł. Pustelnik, S. Knop, P. Pagacz, P. Spurek, Target layer regularization for continual learning using Cramer-Wold distance, Inf. Sci. 609 (2022) 1369–1380.
[26] D. Meng, Q. Zhao, L. Jiang, A theoretical understanding of self-paced learning, Inf. Sci. 414 (2017) 319–328.
[27] P. Micaelli, A.J. Storkey, Zero-shot knowledge transfer via adversarial belief matching, Adv. Neural Inf. Process. Syst. 32 (2019).
[28] S.I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, H. Ghasemzadeh, Improved knowledge distillation via teacher assistant, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 5191–5198.
[29] G.K. Nayak, K.R. Mopuri, V. Shaj, V.B. Radhakrishnan, A. Chakraborty, Zero-shot knowledge distillation in deep networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 4743–4751.
[30] D. Nguyen, S. Gupta, K. Do, S. Venkatesh, Black-box few-shot knowledge distillation, in: European Conference on Computer Vision, 2022.
[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, arXiv:1409.1556, 2014.
[32] P. Soviany, R.T. Ionescu, P. Rota, N. Sebe, Curriculum learning: a survey, Int. J. Comput. Vis. (2022) 1–40.
[33] S. Stanton, P. Izmailov, P. Kirichenko, A.A. Alemi, A.G. Wilson, Does knowledge distillation really work?, Adv. Neural Inf. Process. Syst. 34 (2021) 6906–6919.
[34] J.B. Truong, P. Maini, R. Walls, N. Papernot, Data-free model extraction, preprint, arXiv:2011.14779, 2020.
[35] J.B. Truong, P. Maini, R.J. Walls, N. Papernot, Data-free model extraction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4771–4780.
[36] C. Wang, Z. Wang, D. Chen, S. Zhou, Y. Feng, C. Chen, Online adversarial distillation for graph neural networks, preprint, arXiv:2112.13966, 2021.
[37] L. Wang, K.J. Yoon, Knowledge distillation and student-teacher learning for visual intelligence: a review and new outlooks, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[38] X. Wang, Y. Chen, W. Zhu, A survey on curriculum learning, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[39] Z. Wang, Data-free knowledge distillation with soft targeted transfer set synthesis, preprint, arXiv:2104.04868, 2021.
[40] Z. Wang, Zero-shot knowledge distillation from a decision-based black-box model, in: International Conference on Machine Learning, PMLR, 2021, pp. 10675–10685.
[41] Z. Wang, Z. Zhang, C.Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, T. Pfister, Learning to prompt for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 139–149.
[42] L. Xiang, G. Ding, J. Han, Learning from multiple experts: self-paced knowledge distillation for long-tailed classification, in: European Conference on Computer Vision, Springer, 2020, pp. 247–263.
[43] H. Yin, P. Molchanov, J.M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N.K. Jha, J. Kautz, Dreaming to distill: data-free knowledge transfer via deepinversion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8715–8724.
[44] G. Yu, S. Wang, Z. Cai, X. Liu, C. Xu, C. Wu, Deep anomaly discovery from unlabeled videos via normality advantage and self-paced refinement, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 13987–13998.
[45] S. Zagoruyko, N. Komodakis, Wide residual networks, preprint, arXiv:1605.07146, 2016.
[46] J. Zhang, C. Chen, J. Dong, R. Jia, L. Lyu, Qekd: query-efficient and data-free knowledge distillation from black-box models, preprint, arXiv:2205.11158, 2022.
[47] Q. Zhang, Z. Kang, Z. Xu, S. Huang, H. Fu, Spaks: self-paced multiple kernel subspace clustering with feature smoothing regularization, Knowl.-Based Syst. 253 (2022) 109500.
[48] Y. Zhang, H. Chen, X. Chen, Y. Deng, C. Xu, Y. Wang, Data-free knowledge distillation for image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7852–7861.
[49] B. Zhao, Q. Cui, R. Song, Y. Qiu, J. Liang, Decoupled knowledge distillation, preprint, arXiv:2203.08679, 2022.
[50] Y. Zhou, B. Yang, D.F. Wong, Y. Wan, L.S. Chao, Uncertainty-aware curriculum learning for neural machine translation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 6934–6944.