Learning to Retain while Acquiring: Combating Distribution-Shift in Adversarial Data-Free Knowledge Distillation

Gaurav Patel[†]   Konda Reddy Mopuri[‡]   Qiang Qiu[†]

[†]Purdue University   [‡]Indian Institute of Technology Hyderabad

{gpatel10, qqiu}@purdue.edu, krmopuri@ai.iith.ac.in

Abstract

Data-free Knowledge Distillation (DFKD) has gained popularity recently, with the fundamental idea of carrying out knowledge transfer from a Teacher neural network to a Student neural network in the absence of training data. However, in the Adversarial DFKD framework, the student network's accuracy, suffers due to the non-stationary distribution of the pseudo-samples under multiple generator updates. To this end, at every generator update, we aim to maintain the student's performance on previously encountered examples while acquiring knowledge from samples of the current distribution. Thus, we propose a meta-learning inspired framework by treating the task of Knowledge-Acquisition (learning from newly generated samples) and Knowledge-Retention (retaining knowledge on previously met samples) as meta-train and meta-test, respectively. Hence, we dub our method as Learning to Retain while Acquiring. Moreover, we identify an implicit aligning factor between the Knowledge-Retention and Knowledge-Acquisition tasks indicating that the proposed student update strategy enforces a common gradient direction for both tasks, alleviating interference between the two objectives. Finally, we support our hypothesis by exhibiting extensive evaluation and comparison of our method with prior arts on multiple datasets.

1. Introduction

The primary goal of Data-Free Knowledge Distillation (DFKD) is to acquire a trustworthy alternative dataset for knowledge distillation. The quality of knowledge transfer depends on the caliber of these alternative samples. Noise optimization [11,22,31] and generative reconstruction [4,10,20] are the two primary ways to replace the original training data used in the distillation process with synthetic or pseudo samples. Adversarial DFKD methods [10,19,20] investigate an adversarial exploration framework to seek pseudo-samples. In such a paradigm, the Generator ($\mathcal{G}$) is used to create pseudo-samples as surrogates to perform knowledge distillation/transfer, and the Teacher-Student ($\mathcal{T}$-$\mathcal{S}$) setup acts as a joint discriminator to penalize and update generator parame-

Figure 1. The *top-section* represents the learning evolution of a generic Adversarial Data-Free Knowledge Distillation framework; the color-intensity variation signifies the change in the distribution of the pseudo-samples, the student network, and the generator network over the learning epochs. Under the variation in the distribution of the pseudo-samples, the *bottom-section* shows the learning curves for cases when the student accuracy degrades (shown in Red), which is undesirable, and when the student accuracy is maintained, if not improved, as proposed (shown in Green).

ters ($\theta_\mathcal{G}$) (Figure 1). In the adversarial framework, the generator explores the input space to find suitable pseudo-samples as the distillation progresses. Consequently, the distribution of the generated samples consistently keeps changing during the process due to the generator updates [26]. From the student network's perspective, at each iteration, the pseudo samples seem to be generated from different generator parameters ($\theta_\mathcal{G}$). Hence, the convergence of the student network gets hampered due to successive distributional alterations over time [29], as depicted by the red curve in Figure 1. This observation hints that updating the student network, solely using the samples generated from the current generator parameters is not adequate to generalize the student. Moreover, the student forgets the knowledge acquired previously and decelerates the knowledge distillation. Therefore, the generator, apart from exploring the input space, seeks

Figure 2. Student update strategies: (a) Typical student update by optimizing the *Knowledge-Acquisition* loss ($\mathcal{L}_{Acq}$) with the batch pseudo samples ($\hat{x}$), produced by the generator ($\mathcal{G}$) [5, 10, 20]. (b) Student update with simultaneous optimization of the *Knowledge-Acquisition* loss ($\mathcal{L}_{Acq}$) and *Knowledge-Retention* loss ($\mathcal{L}_{Ret}$) on the batch of pseudo samples ($\hat{x}$) and memory samples ($\hat{x}_m$) obtained from the generator ($\mathcal{G}$) and the memory ($\mathcal{M}$), respectively [2, 3]. (c) The proposed student update strategy, which treats $\mathcal{L}_{Acq}$ as meta-train and $\mathcal{L}_{Ret}$ as meta-test, and implicitly imposes the alignment between them.

to compensate for the loss of knowledge in future iterations. Additionally, in a practical setting, during the distillation process, high variation in the student network's classification accuracy is undesirable, especially when the validation data is not available, since that prevents the user from tracking the student's accuracy over time, and selecting the distilled model parameters with the highest accuracy.

To circumvent the above-discussed problem, existing methods maintain a memory buffer to rehearse the examples from previously encountered distributions while learning with current examples. Binci *et al.* [3] introduce *Replay* based methods to explicitly retrain/replay on a limited subset of previously encountered samples while training on the current examples. Then, carrying forward, they use Generative-Replay [28] to transfer the learned examples to an auxiliary generative model (VAE), and sample from the VAE's decoder in subsequent iterations [2]. Nonetheless, the performance of these methods is upper bounded by joint training on previous and current examples [7]. Although, recent works have focused on modeling the memory, we seek to work towards effectively utilizing the samples from memory.

In this paper, we aim to update the student network parameters (θ_S) such that its performance does not degrade on the samples previously produced by the generator network ($\mathcal{G}$), aspiring towards *Learning to Retain while Acquiring*. Thus, we propose a meta-learning inspired strategy to achieve this goal. We treat the task of *Knowledge-Acquisition* (learning from newly generated samples) and *Knowledge-Retention* (learning from previously encountered samples from memory) as meta-train and meta-test, respectively. Hence, in the proposed approach, the student network acquires new information while maintaining performance on previously encountered samples. By doing so, the proposed strategy (Figure 2c) implicitly aligns *Knowledge-Acquisition* and *Knowledge-Retention*, as opposed to simply combining them [2, 3] without any coordination or alignment (Figure 2b), which leaves them to potentially interfere with one another.

Additionally, analyzing the proposed meta-objective, we discover that (in Section 3.4) the latent alignment factor as

the dot product between the gradients of the *Knowledge-Acquisition* and *Knowledge-Retention* objectives, suggesting that the meta-objective enforces a common gradient direction for both tasks, encouraging the alignment between the task-specific gradients. Thus, the proposed method simultaneously minimizes the loss and matches the gradients corresponding to the individual tasks (*Knowledge-Acquisition* and *Knowledge-Retention*), enforcing the optimization paths to be same for both tasks.

Moreover, the proposed student update strategy is scalable to different deep architectures as the gradient alignment is implicit, and memory-agnostic, making no assumptions about the replay scheme employed. Nonetheless, recent works on gradient alignment, have shown great empirical advantages in Zero-Shot Learning [25], Distributed/Federated Learning [6] and Domain Generalization [27]. Our method extends the advantages of gradient alignment to memory-based Adversarial DFKD, thus strengthening the empirical findings in these works.

Finally, to demonstrate the advantages of the proposed student update strategy, we evaluate and compare against current non-memory [4, 5, 10] and memory-based [2, 3] Adversarial DFKD methods, and observe substantial improvement in the student learning evolution.

In summary, our contributions are as follows:

- We propose a novel meta-learning inspired student update strategy in the Adversarial DFKD setting, that aims to maintain the student's performance on previously encountered examples (*Knowledge-Retention*) while acquiring knowledge from samples of the current distribution (*Knowledge-Acquisition*).
- We theoretically identify (in Section 3.4) that the proposed student update strategy enforces an implicit gradient alignment between the *Knowledge-Acquisition* and *Knowledge-Retention* tasks.
- Finally, we evaluate our method and compare against various Adversarial DFKD methods, on multiple student architectures and replay schemes (Memory Buffer and Generative Replay).

Figure 3. An illustration of the proposed DFKD framework. The framework consists of the Generator ($\mathcal{G}$), Teacher ($\mathcal{T}$), Student ($\mathcal{S}$) and the Memory ($\mathcal{M}$). $\mathcal{G}$ and $\mathcal{S}$ are updated alternatively, similar to the GAN [13] framework with the generator loss ($\mathcal{L}_\mathcal{G}$) optimizing $\mathcal{G}$, and the *Knowledge-Acquisition* loss ($\mathcal{L}_{Acq}$) and the *Knowledge-Retention* loss ($\mathcal{L}_{Ret}$) optimizing the student ($\mathcal{S}$). We use $\mathcal{M}$ in a generalized way to denote any type of replay schemes (Memory Buffer or Generative Replay in our case).

2. Related Work

Adversarial Data-Free Knowledge Distillation: In the Adversarial Data-Free Knowledge Distillation paradigm, A generative model is trained to synthesize pseudo-samples that serve as queries for the Teacher ($\mathcal{T}$) and the Student ($\mathcal{S}$) [5, 10, 20]. ZSKT [20] attempts data-free knowledge transfer by first training a generator in an adversarial fashion to look for samples on which the student and teacher do not match well. To improve the model discrepancy measure, it adopts the Kullback–Leibler (KL) divergence, and introduces attention transfer [34] to aid knowledge transfer. Moreover, DFAD [10] recommends Mean Absolute Error (MAE) as a model discrepancy function to prevent decayed gradients on converged samples. Furthermore, the adversarial framework was extended by Choi *et al.* [5] in the context of model quantization, by proposing adversarial data-free quantization (DFQ), and introducing additional regularization terms that match the mean and standard deviation of the generated pseudo-samples with the teacher model's batchnorm statistics, and imposes batch categorical entropy maximization, such that sample from each class appear equally in the generated batch. Fang *et al.* recently introduced FastDFKD [9], an effective method with a meta generator to speed up the DFKD process, delivering a 100-fold increase in the knowledge transfer rate.

Handling Distribution Shift in Adversarial DFKD: To counter the distribution mismatch and the catastrophic forgetting phenomenon in the adversarial framework [26], Binici *et al.* [3] suggested maintaining a dynamic collection of synthetic samples throughout training iterations to prevent catastrophic forgetting in DFKD. Moreover, in their latest work [2], they introduce generative pseudo-replay [28] in which an auxiliary generative model simultaneously learns the distribution of the samples produced by the generator ($\mathcal{G}$). Throughout the training process, examples are generated from the auxiliary generator to replay during training.

Nonetheless, these works have focused on modeling the memory buffer. A related line of research maintains an exponentially moving average (EMA) of the generator model $\mathcal{G}$ [8] to replace the typical Memory-Buffer and Generative Replay. Nonetheless, our work focuses on the effective utilization of the samples obtained from the memory.

3. Methodology

In Section 3.1, we first provide a brief overview of Adversarial DFKD. Then, in Section 3.2, we discuss the data-free knowledge distillation objective. In Sections 3.3 and 3.4, we elaborate on the proposed student update strategy. Lastly, in Section 3.5, we discuss the adopted generator update strategy used for the baselines and the proposed framework.

3.1. Adversarial Data-Free Knowledge Distillation

In the Adversarial DFKD framework, a generator ($\mathcal{G}$) is used to create pseudo-samples as surrogates to perform knowledge distillation/transfer, and the teacher-student ($\mathcal{T}$-$\mathcal{S}$) setup acts as a joint discriminator to penalize and update generator parameters ($\theta_\mathcal{G}$) in an adversarial manner. After updating $\theta_\mathcal{G}$, random samples are generated and used to minimize the $\mathcal{T}$-$\mathcal{S}$ discrepancy by updating the student parameters ($\theta_\mathcal{S}$). The generator and the student are optimized alternatively up until a fixed number of pre-defined iterations. In essence, the goal of DFKD is to craft a lightweight student model ($\mathcal{S}$) by harnessing valuable knowledge from the well-trained Teacher model ($\mathcal{T}$) in the absence of training data. A general overview of Adversarial DFKD framework is illustrated in Figure 1.

3.2. Goal of Data Free Knowledge Distillation

A student model ($\mathcal{S}$) is trained to match the teacher's ($\mathcal{T}$) predictions on its unavailable target domain ($\mathcal{D}_\mathcal{T}$) as part of the distillation process. Let, $p_\mathcal{S}(x) = \text{Softmax}(\mathcal{S}_{\theta_\mathcal{S}}(x))$ and $p_\mathcal{T}(x) = \text{Softmax}(\mathcal{T}_{\theta_\mathcal{T}}(x)), \forall x \in \mathcal{D}_\mathcal{T}$, denote the predicted

Figure 4. Learning evolution of the proposed method compared to the prior arts (MB-DFKD [3] and PRE-DFKD [2]) that employ replay. The plots visualize the Accuracy (%) evolution (*top-row*) and the Cumulative Mean Accuracy (%) evolution (*bottom-row*) of the ResNet-18 [14] student network on SVHN [23], CIFAR10 [16], CIFAR100 [16], and Tiny-ImageNet [17] datasets. The proposed method is in **Blue**.

Student and Teacher probability mass across the classes, respectively. We seek to find the parameters $\theta_{\mathcal{S}}$ of the student model that will reduce the probability of errors (P) between the predictions $p_{\mathcal{S}}(x)$ and $p_{\mathcal{T}}(x)$:

$$\min_{\theta_{\mathcal{S}}} P_{x \sim \mathcal{D}_{\mathcal{T}}} \left(\arg\max_i p_{\mathcal{S}}^i(x) \neq \arg\max_i p_{\mathcal{T}}^i(x) \right),$$

where the superscript i denotes the ith probability score of the predicted masses, $p_{\mathcal{S}}(x)$ and $p_{\mathcal{T}}(x)$.

However, DFKD suggests minimizing the student's error on a pseudo dataset $\mathcal{D}_{\mathcal{P}}$, since the teacher's original training data distribution $\mathcal{D}_{\mathcal{T}}$ is not available. Typically, a loss function, say $\mathcal{L}_{KD}$, that gauges disagreement between the teacher and the student is minimized $\forall \hat{x} \in \mathcal{D}_{\mathcal{P}}$ by optimizing the student parameters ($\theta_{\mathcal{S}}$):

$$\min_{\theta_{\mathcal{S}}} \mathbb{E}_{\hat{x} \sim \mathcal{D}_{\mathcal{P}}}[\mathcal{L}_{KD}(\mathcal{T}_{\theta_{\mathcal{T}}}(\hat{x}), \mathcal{S}_{\theta_{\mathcal{S}}}(\hat{x}))]. \quad (1)$$

We use the Mean Absolute Error (MAE) to define loss measure ($\mathcal{L}_{KD}$) as suggested in [10]. Suppose, given the predicted logits (pre-softmax predictions) $t(\hat{x}) = \mathcal{T}_{\theta_{\mathcal{T}}}(\hat{x})$ from the teacher model and the predicted logits $s(\hat{x}) = \mathcal{S}_{\theta_{\mathcal{S}}}(\hat{x})$ from the student model, we define $\mathcal{L}_{KD}$ as:

$$\mathcal{L}_{KD}(t(\hat{x}), s(\hat{x})) = \|t(\hat{x}) - s(\hat{x})\|_1. \quad (2)$$

3.3. Learning to Retain while Acquiring

Our novelty resides in the student update strategy, as described earlier. *Knowledge-Retention* and *Knowledge-Acquisition* relate to two independent goals and can therefore be considered as two discrete tasks. In fact, by aligning these two goals, we empirically observe that, they can cooperate to retain the knowledge on previously encountered examples while acquiring knowledge from newly generated samples. The proposed method utilizes a meta-learning-inspired optimization strategy to effectively replay the samples from memory, say $\mathcal{M}$.

In the typical Adversarial DFKD setup, the student update objective with the generated pseudo samples ($\hat{x}$), *i.e.*, the *Knowledge-Acquisition* task ($\mathcal{L}_{Acq}$) (Figure 2a), is formulated as:

$$\min_{\theta_{\mathcal{S}}} \mathcal{L}_{Acq}(\theta_{\mathcal{S}}) = \min_{\theta_{\mathcal{S}}} \mathbb{E}_{\hat{x}}[\mathcal{L}(\mathcal{T}_{\theta_{\mathcal{T}}}(\hat{x}), \mathcal{S}_{\theta_{\mathcal{S}}}(\hat{x}))], \quad (3)$$

where $\mathcal{L}$ is the MAE (2) between the teacher and the student logits, and $\hat{x} = \mathcal{G}(z), z \sim \mathcal{N}(0, I)$, denotes the batch of randomly generated samples.

Moreover, to alleviate the distribution drift during knowledge distillation in the adversarial setting, previous works have maintained a memory buffer, to store $\hat{x}$ and replay at later iterations to help the student recall the knowledge [3]. At a fixed frequency, batches of samples from current iterations are updated in the memory [3]. Also, recent works [2] have explored using the Generative Replay [28] strategy to simultaneously store the samples, as they are encountered, in a generative model (VAE [15]), and later replay by generating samples from the VAE decoder. Hence, the optimization objective for the *Knowledge-Retention* ($\mathcal{L}_{Ret}$) can be defined as follows:

$$\min_{\theta_{\mathcal{S}}} \mathcal{L}_{Ret}(\theta_{\mathcal{S}}) = \min_{\theta_{\mathcal{S}}} \mathbb{E}_{\hat{x}_m}[\mathcal{L}(\mathcal{T}_{\theta_{\mathcal{T}}}(\hat{x}_m), \mathcal{S}_{\theta_{\mathcal{S}}}(\hat{x}_m))],$$
$$\hat{x}_m \sim \mathcal{M}, \quad (4)$$

where, with a slight abuse of notation, $\mathcal{M}$ denotes a Memory Buffer or Generative Replay or any other replay scheme. Thus, the overall optimization objective to update $\theta_{\mathcal{S}}$, while considering both the new samples (generated from the latest $\theta_{\mathcal{G}}$) and the old samples (sampled from $\mathcal{M}$) (Figure 2b) is defined as:

$$\min_{\theta_{\mathcal{S}}} \mathcal{L}_{Acq}(\theta_{\mathcal{S}}) + \mathcal{L}_{Ret}(\theta_{\mathcal{S}}). \quad (5)$$

However, the objective in (5) attempts to simultaneously optimizes $\mathcal{L}_{Ret}$ and $\mathcal{L}_{Acq}$ but does not seek to align the objectives, which leaves them to potentially interfere with one another. Say, we denote the gradients of *Knowledge Acquisition* and *Knowledge Retention* tasks with $\nabla \mathcal{L}_{Acq}(\theta)$ and $\nabla \mathcal{L}_{Ret}(\theta)$, respectively. If $\nabla \mathcal{L}_{Acq}(\theta)$ and $\nabla \mathcal{L}_{Ret}(\theta)$ point in a similar direction or are said to be aligned, *i.e.*, $\nabla \mathcal{L}_{Acq}(\theta).\nabla \mathcal{L}_{Ret}(\theta) > 0$, then taking a gradient step along $\nabla \mathcal{L}_{Acq}(\theta)$ or $\nabla \mathcal{L}_{Ret}(\theta)$ improves the model's performance on both tasks. This is however not the case when $\nabla \mathcal{L}_{Acq}(\theta)$ and $\nabla \mathcal{L}_{Ret}(\theta)$ point in different directions, *i.e.*, $\nabla \mathcal{L}_{Acq}(\theta).\nabla \mathcal{L}_{Ret}(\theta) \leq 0$, and the weight parameters obtained, may not be optimal for both the tasks simultaneously. Hence, the intended effect of having the gradient directions aligned is to obtain student parameters ($\theta_{\mathcal{S}}$) that have optimal performance on both $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$.

4

The proposed meta-learning inspired approach, seeks to align the two tasks, $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$. We take cues from Model-Agnostic Meta-learning (MAML) [12], and adapt it to Adversarial DFKD. At each iteration of the parameter update, we pose *Knowledge-Acquisition* and *Knowledge-Acquisition* as meta-train and meta-test, respectively. Which means, we perform a single gradient descent step using the current samples ($\hat{x}$) produced from the generator network ($\mathcal{G}$), on the parameters $\theta_{\mathcal{S}}$ and obtain $\theta_{\mathcal{S}}{}'$ as $\theta_{\mathcal{S}}' = \theta_{\mathcal{S}} - \alpha\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}})$, where $\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}})$ denotes gradient of $\mathcal{L}_{Acq}$ at $\theta_{\mathcal{S}}$. Then we optimize on the batch of samples ($\hat{x}_m$) obtained from the memory ($\mathcal{M}$), with the parameters $\theta_{\mathcal{S}}'$, and then make a final update on $\theta_{\mathcal{S}}$. Thus, formally, the overall meta-optimization objective, with the task of *Knowledge Acquisition* serving as meta-train and the task of *Knowledge Retention* as meta-test (Figure 2c), can be defined as follows:

$$\min_{\theta_{\mathcal{S}}} \mathcal{L}_{Acq}(\theta_{\mathcal{S}}) + \mathcal{L}_{Ret}(\theta_{\mathcal{S}}') = \min_{\theta_{\mathcal{S}}} \mathcal{L}_{Acq}(\theta_{\mathcal{S}}) + \\ \mathcal{L}_{Ret}(\theta_{\mathcal{S}} - \alpha\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}})). \quad (6)$$

3.4. How does the Proposed Student Update Strategy Promote Alignment?

In this subsection, we analyze the proposed objective (6) to understand how it results in the desired alignment between the *Knowledge-Acquisiton* and *Knowledge-Retention* tasks. We assume that meta-train and meta-test tasks provide us with losses $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$; in our case, $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$ are the same function computed on the batches $\hat{x}$ and $\hat{x}_m$, respectively. We utilize Taylor's expansion to elaborate the gradient of $\mathcal{L}_{Ret}$ at a point θ displaced by ϕ_{θ}, as described in Lemma 1,

Lemma 1. *If $\mathcal{L}_{Ret}$ has Lipschitz Hessian, i.e., $\|\nabla^2\mathcal{L}_{Ret}(\theta_1) - \nabla^2\mathcal{L}_{Ret}(\theta_2)\| \leq \rho\|\theta_1 - \theta_2\|$ for some $\rho > 0$, then:*

$$\nabla\mathcal{L}_{Ret}(\theta + \phi_{\theta}) = \nabla\mathcal{L}_{Ret}(\theta) + \nabla^2\mathcal{L}_{Ret}(\theta)\phi_{\theta} \\ + \mathcal{O}(\|\phi_{\theta}\|^2).$$

For instance, when $\phi_{\theta} = -\alpha\nabla\mathcal{L}_{Acq}(\theta)$, we have,

$$\nabla\mathcal{L}_{Ret}(\theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)) = \nabla\mathcal{L}_{Ret}(\theta) \\ - \alpha\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}(\theta) \\ + \mathcal{O}(\alpha^2).$$

Theorem 1. *If $\theta' = \theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)$, denotes a single gradient descent step on θ with the objective $\mathcal{L}_{Acq}(\theta)$, where α is a scalar, and $\nabla\mathcal{L}_{Acq}(\theta)$ denotes the gradient of the objective at θ, then:*

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\nabla^2\mathcal{L}_{Ret}(\theta).\nabla\mathcal{L}_{Acq}(\theta) \\ - \alpha\nabla^2\mathcal{L}_{Acq}(\theta).\nabla\mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2).$$

Proof. Please refer to the Supplemental Material (Theorem 1.

While optimizing the objective defined in (6) using stochastic gradient descent, we would need to compute the gradient of the $\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')$ w.r.t $\theta_{\mathcal{S}}$. Therefore, utilizing Theorem 1 we express $\frac{\partial\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')}{\partial\theta_{\mathcal{S}}}$ as:

$$\frac{\partial\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')}{\partial\theta_{\mathcal{S}}} = \nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}) \\ - \alpha\nabla^2\mathcal{L}_{Ret}(\theta_{\mathcal{S}}).\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}}) \\ - \alpha\nabla^2\mathcal{L}_{Acq}(\theta_{\mathcal{S}}).\nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}) + \mathcal{O}(\alpha^2), \quad (7)$$

using the product rule $\nabla a.b + \nabla b.a = \nabla(a.b)$, we get:

$$\frac{\partial\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')}{\partial\theta_{\mathcal{S}}} = \nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}) \\ - \alpha\nabla\underbrace{(\nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}).\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}}))}_{Gradient\ Alignment} + \mathcal{O}(\alpha^2). \quad (8)$$

From the analysis above, we observe that the gradient of $\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')$ at $\theta_{\mathcal{S}}$ (in (8)) produces the gradient of the gradient-product. This indicates, when optimizing $\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')$ (in (6)), the gradient of $\mathcal{L}_{Ret}(\theta_{\mathcal{S}}')$ at $\theta_{\mathcal{S}}$, has a negatively scaled gradient of the gradient-product term $\nabla(\nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}).\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}}))$ (derived in (8)), suggesting that the overall-gradients minimize $\mathcal{L}_{Ret}(\theta_{\mathcal{S}})$ and maximize $\nabla\mathcal{L}_{Ret}(\theta_{\mathcal{S}}).\nabla\mathcal{L}_{Acq}(\theta_{\mathcal{S}})$. Hence, optimizing (6) enforces the updates on $\mathcal{L}_{Ret}(\theta_{\mathcal{S}})$ and $\mathcal{L}_{Acq}(\theta_{\mathcal{S}})$ to seek a common direction, by maximizing the gradient-product.

3.5. Generator Update in Adversarial Exploration-based DFKD

In the absence of the training dataset ($\mathcal{D}_{\mathcal{T}}$), the generator ($\mathcal{G}$) is utilized to obtain pseudo-samples ($\hat{x}$) and perform knowledge-distillation, *i.e.*, $\hat{x} = \mathcal{G}(z)$, $z \sim \mathcal{N}(0, I)$. The generator is learned to maximize the disagreement between Teacher network ($\mathcal{T}_{\theta_{\mathcal{T}}}$) and the Student network ($\mathcal{S}_{\theta_{\mathcal{S}}}$). Additionally, for the generated data $\hat{x}$ to mimic similar responses from the teacher as the real data, we include a prior loss $\mathcal{L}_{\mathcal{P}}$ [4] to be minimized alongside maximizing the discrepancy (D). Hence, we update the generator parameters ($\theta_{\mathcal{G}}$) by maximizing the following objective:

$$\max_{\theta_{\mathcal{G}}} \mathbb{E}_{z \sim \mathcal{N}(0,I)}[D(\mathcal{T}_{\theta_{\mathcal{T}}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z)), \mathcal{S}_{\theta_{\mathcal{S}}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z))) \\ - \mathcal{L}_{\mathcal{P}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z))]. \quad (9)$$

Typically the disagreement function (D) for the generator is identical to the teacher-student disagreement term [5, 10]. Instead, for teacher-student discrepancy maximization we use the Jensen-Shannon (JS) ($\mathcal{L}_{JS}$) divergence. Our motivation

Table 1. Distillation results of the Adversarial DFKD methods on four image classification benchmark datasets, SVHN [23], CIFAR10 [16], CIFAR100 [16], Tiny-ImageNet [17]. Primarily we compare against the replay-based methods, present at the bottom panel of each dataset. The best numbers from our method on the Bank-based replay are highlighted in **Navy**, and for the one with Generative replay are highlighted in **Maroon**. The first five columns represent the $\mu[\mathcal{S}_{Acc}]$ and the $\sigma^2[\mathcal{S}_{Acc}]$ at different epoch percentiles (described in Section 4.1). The last column represents the maximum test accuracy (Acc$_{max}$ (%)) attained by the student network in the entire training period. Note: For SVHN and Tiny-ImageNet we were unable to reproduce the results of DFQ [5] in our setting. DFKD* denotes the baseline method without any replay.

Method	> 0th Percentile		> 20th Percentile		> 40th Percentile		> 60th Percentile		> 80th Percentile		Acc$_{max}$ (%)
	$\mu\uparrow$	$\sigma^2\downarrow$	$\mu\uparrow$	$\sigma^2\downarrow$	$\mu\uparrow$	$\sigma^2\downarrow$	$\mu\uparrow$	$\sigma^2\downarrow$	$\mu\uparrow$	$\sigma^2\downarrow$	
SVHN [23] $\mathcal{T}_{Acc}=97.45\%,\ \mathcal{S}_{Acc}=97.26\%,\ (\mathcal{E}_{max})=200$											
DAFL [4]	85.15	241.70	90.49	8.71	91.75	3.73	92.88	0.84	93.59	0.20	94.23
DFAD [10]	92.52	93.42	94.10	0.07	94.13	0.07	94.14	0.08	94.11	0.07	94.68
DFKD* [3]	91.84	8.59	92.53	6.76	93.43	4.39	94.61	1.72	95.72	0.04	95.97
MB-DFKD [3]	89.51	21.13	89.28	24.31	88.65	30.63	87.14	38.67	81.67	15.24	94.11
PRE-DFKD [2]	94.08	2.11	94.53	1.11	94.97	0.66	95.42	0.26	95.85	0.04	96.10
Ours-1 (Memory Bank)	92.78	7.44	93.69	2.21	94.25	1.54	94.94	0.62	95.59	0.05	95.88
Ours-2 (Generative Replay)	94.48	1.64	94.79	0.71	95.09	0.51	95.47	0.27	95.91	0.03	96.15
CIFAR10 [16] $\mathcal{T}_{Acc}=95.72\%,\ \mathcal{S}_{Acc}=95.23\%,\ (\mathcal{E}_{max})=200$											
DAFL [4]	62.78	629.13	72.51	296.44	80.57	110.18	87.05	22.40	90.76	2.04	92.23
DFAD [10]	85.88	152.39	89.91	11.95	91.46	5.66	92.66	0.07	92.85	0.04	93.21
DFQ [5]	81.25	98.86	84.41	37.40	87.08	19.59	89.72	6.15	91.87	0.67	92.90
DFKD* [3]	83.57	107.77	86.94	18.42	88.67	11.83	90.67	4.09	92.32	0.38	93.02
MB-DFKD [3]	84.29	95.74	87.25	16.47	88.81	11.00	90.71	3.79	92.31	0.35	93.03
PRE-DFKD [2]	87.10	45.69	88.88	10.86	90.24	6.26	91.75	1.84	92.92	0.14	93.41
Ours-1 (Memory Bank)	85.53	91.33	88.58	12.66	89.96	8.26	91.66	2.86	93.09	0.32	93.73
Ours-2 (Generative Replay)	88.07	60.86	90.52	5.20	91.44	3.18	92.45	1.45	93.48	0.18	94.02
CIFAR100 [16] $\mathcal{T}_{Acc}=77.94\%,\ \mathcal{S}_{Acc}=77.10\%,\ (\mathcal{E}_{max})=400$											
DAFL [4]	52.48	437.88	61.65	82.96	65.88	32.23	69.23	11.93	72.23	1.29	73.78
DFAD [10]	59.62	192.64	65.32	28.84	67.71	2.72	68.69	0.33	69.07	0.24	69.73
DFQ [5]	68.20	55.91	70.45	9.78	71.77	5.88	73.19	2.22	74.48	0.47	75.39
DFKD* [3]	69.36	74.67	72.35	8.34	73.59	4.54	74.86	1.63	75.94	0.18	76.51
MB-DFKD [3]	66.05	207.29	71.16	14.67	72.97	5.61	74.40	1.56	75.48	0.18	76.14
PRE-DFKD [2]	70.23	86.63	73.39	6.77	74.59	2.88	75.60	1.03	76.46	0.12	76.93
Ours-1 (Memory Bank)	69.87	75.67	72.96	8.72	74.30	4.03	75.49	1.38	76.50	0.20	77.11
Ours-2 (Generative Replay)	71.49	60.17	74.16	4.61	75.12	2.26	76.01	0.74	76.75	0.09	77.21
Tiny-ImageNet [17] $\mathcal{T}_{Acc}=60.83\%,\ \mathcal{S}_{Acc}=57.88\%,\ (\mathcal{E}_{max})=500$											
DAFL [4]	21.04	106.04	24.95	52.56	28.10	28.22	31.19	11.44	34.11	1.33	35.46
DFAD [10]	14.60	22.32	16.48	7.48	17.76	1.43	18.44	0.26	18.84	0.10	19.60
DFKD* [3]	34.55	86.20	38.17	23.88	40.28	13.28	42.40	5.32	44.38	0.79	45.61
MB-DFKD [3]	34.16	122.92	38.77	32.42	41.18	18.40	43.63	8.48	46.17	1.73	47.96
PRE-DFKD [2]	38.89	80.12	42.27	21.66	44.25	12.61	46.31	5.47	48.33	1.39	49.94
Ours-1 (Memory Bank)	36.34	94.62	40.03	26.20	42.28	13.76	44.43	5.82	46.52	0.95	47.96
Ours-2 (Generative Replay)	39.09	74.95	42.30	21.23	44.19	13.55	46.30	6.21	48.48	1.12	49.88

to use JS divergence is based on empirical study by Binici *et al.* [3]. Hence, D is defined as:

$$D(a,b) = \mathcal{L}_{JS}(p(a), p(b)),$$

$$\mathcal{L}_{JS}(p(a), p(b)) = \frac{1}{2}(\mathcal{L}_{KL}(p(a), m) + \mathcal{L}_{KL}(p(b), m)),\ \text{and}$$

$$m = \frac{1}{2}(p(a) + p(b)). \quad (10)$$

Here $\mathcal{L}_{KL}$ stands for the Kullback–Leibler divergence and $p(a)$ and $p(b)$ denote the probability vectors obtained after the Softmax applied to the arguments a and b, respectively.

Moreover, the prior loss $\mathcal{L}_{\mathcal{P}}$ [3] is defined as the combination of three loss functions ($\mathcal{L}_{OH}$, $\mathcal{L}_A$, and $\mathcal{L}_{EM}$) that

capture different characteristics from the teacher model and impose them on the pseudo samples $\hat{x}$, and is defined as:

$$\mathcal{L}_{\mathcal{P}} = \mathcal{L}_{OH} + \gamma\mathcal{L}_A + \delta\mathcal{L}_{EM}, \quad (11)$$

where, γ and δ denote the weighing scalar coefficients.
• $\mathcal{L}_{OH}$ is the one-hot loss that forces the generated samples to have strong (one-hot vector like) predictions when input to the teacher. It is defined as the cross-entropy between the teacher's softmax output $p_{\mathcal{T}}(\hat{x}_n) = \text{Softmax}(\mathcal{T}_{\theta_{\mathcal{T}}}(\hat{x}_n))$, $\hat{x}_n \in \hat{x}$, and its one-hot vector version $e_c \in \{0,1\}^C$, where C denotes the total number of classes and e_c denotes the c-th canonical one-hot-vector, and $c = \arg\max_i(p_{\mathcal{T}}^i(\hat{x}_n))$, the superscript i denotes the

6

Figure 5. Student learning curves depicting the learning evolution of Wide-ResNet (WRN) [33]. The WRN-16-1 (*top-row*), and WRN-16-2 (*bottom-row*) networks are distilled by a WRN-40-2 teacher network pre-trained on CIFAR10 ($\mathcal{T}_{Acc} = 94.87\%$). Each column represent the learning curves with the Buffer-based (with different memory buffer sizes) and Generative replay schemes. The proposed method is in **Blue**.

i^{th} probability score of the predicted mass vector $p_{\mathcal{T}}(\hat{x}_n)$. Hence, $\mathcal{L}_{OH}$ is defined as:

$$\mathcal{L}_{OH} = -\frac{1}{N} \sum_{n=1}^{N} e_c^{\top} \log(p_{\mathcal{T}}(\hat{x}_n)). \qquad (12)$$

• $\mathcal{L}_A$ is the activation loss motivated by the notion that meaningful inputs result in higher-valued activation maps in a well-trained network [4] and is defined as:

$$\mathcal{L}_A = -\frac{1}{NL} \sum_{n=1}^{N} \sum_{l=1}^{L} \|\mathcal{A}_{\mathcal{T}}^l(\hat{x}_n)\|_1, \qquad (13)$$

where $\mathcal{A}_{\mathcal{T}}^l(\hat{x}_n)$ denotes the activation of the l-th layer in the Teacher network ($\mathcal{T}_{\theta_{\mathcal{T}}}$) for the input n^{th} input $\hat{x}_n \in \hat{x}$.

• $\mathcal{L}_{EM}$ loss is the entropy-maximization term imposed on the generator to output an equal number of pseudo-samples from each category [5]. In other words, the intra-batch entropy is maximized, resulting in a similar number of samples for each category *i.e.* if $\bar{p}_{\mathcal{T}} = \frac{1}{N} \sum_{n=1}^{N} p_{\mathcal{T}}(\hat{x}_n)$, then the loss $\mathcal{L}_{EM}$ is defined as:

$$\mathcal{L}_{EM} = \bar{p}_{\mathcal{T}}^{\top} \log(\bar{p}_{\mathcal{T}}). \qquad (14)$$

In sum, the Generator loss objective ($\mathcal{L}_{\mathcal{G}}$) is defined as:

$$\mathcal{L}_{\mathcal{G}}(\theta_{\mathcal{G}}) = -D(\mathcal{T}_{\theta_{\mathcal{T}}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z)), \mathcal{S}_{\theta_{\mathcal{S}}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z))) + \mathcal{L}_{\mathcal{P}}(\mathcal{G}_{\theta_{\mathcal{G}}}(z)). \qquad (15)$$

4. Experiments

4.1. Experimental Settings

Datasets: We evaluate the proposed method on SVHN [23], CIFAR10 [16], CIFAR100 [16] and Tiny-ImageNet [17] datasets.

Teacher model: For CIFAR10 and CIFAR100 datasets, we used the pre-trained teacher models made available by the authors of [11] and [3], respectively. For SVHN and Tiny-ImageNet we trained the teacher network from scratch. We provide the training details of the teacher models in the Supplemental Material.

Definition of an Epoch in DFKD: In Adversarial DFKD, the notion of an *epoch* is obscure. In a typical deep neural network-based classification training, an epoch is defined as one complete pass over the available training data. However, DFKD has no access to the training data; instead, the pseudo samples generated on-the-fly are used to distill knowledge to the student network. Therefore, prior works [3,5,10] defined an epoch in terms of a fixed number of training iterations ($\mathcal{I}$), where each iteration consists of a set number of generator update steps (g) and student update steps (s). Hence, to be consistent across the baselines and prior arts, we use the same number of training iterations, generator update steps, and student updates steps to define an epoch. For all the methods, we set $\mathcal{I} = 72$, $g = 1$, and $s = 10$ and use a batch size of 512 of the sampled noise (z) to generate the pseudo samples and optimize the parameters $\theta_{\mathcal{G}}$ and $\theta_{\mathcal{S}}$.

Training Details: Due to the page-limit constraint, the training details are provided in the Supplemental Material.

Evaluation: We evaluate the methods by comparing the mean and variance of the student network's test accuracy ($\mathcal{S}_{Acc}$), denoting them as $\mu[\mathcal{S}_{Acc}]$, and $\sigma^2[\mathcal{S}_{Acc}]$, respectively, across the epochs, motivated by Binci *et al.* [2]. Specifically, we compare the different sections of the student's learning evolution by partitioning them into different epoch percentiles. For example, computing the $\mu[\mathcal{S}_{Acc}]$ and $\sigma^2[\mathcal{S}_{Acc}]$ for epochs greater than the nth percentile conveys the mean and variance across all the epochs greater than the $\frac{n}{100}^{th}$ of the total number of training epochs.

4.2. Results and Observations

Baseline and State-of-the-art Comparisons In Table 1, we analyze our method on classification task and compare it with prior Adversarial DFKD methods [4,5,10] and closely related memory-based Adversarial DFKD methods [2,3]. For a fair comparison across the methods, we re-implemented all the methods and used the same generator architecture to generate the pseudo samples. For each of the methods we report the $\mu[\mathcal{S}_{Acc}]$ and $\sigma^2[\mathcal{S}_{Acc}]$ at different epoch percentiles and the maximum accuracy (Acc$_{max}$ (%)) attained by the student. We observe that, compared to MB-DFKD (Memory

	ZSKD[a] [22]	ADI[b] [31]	CMI[b] [11]	DeGAN[c] [1]	EATSKD[c] [21]	KEGNET[a] [32]	ZSKT[b] [20]	DDAD[a] [35]	DAFL[d] [4]	DFAD[d] [10]	DFQ[d] [5]	MB-DFKD[d] [3]	PRE-DFKD[d] [2]	*Ours-1*	*Ours-2*
$\mathcal{T}_{Acc}$ (%)	77.50	78.05	78.05	77.94	77.94	77.50	78.05	77.50	77.94	77.94	77.94	77.94	77.94	77.94	77.94
$\mathcal{S}_{Acc}$ (%)	70.21	61.32	77.04	65.25	67.18	73.91	67.74	75.04	73.79	69.73	75.39	76.14	76.93	**77.11**	**77.21**
$\Delta_{\mathcal{T}_{Acc}-\mathcal{S}_{Acc}}$	7.29	16.73	1.01	12.69	10.76	3.59	10.31	2.46	4.15	8.21	2.55	1.80	1.01	**0.83**	**0.73**

Table 2. Classification accuracy (in %) of the student trained using various DFKD methods on CIFAR100 with ResNet-34 [14] as the teacher and ResNet-18 [14] as the student. $\mathcal{T}_{Acc}$ and $\mathcal{S}_{Acc}$ denote the Teacher network's and the Student network's accuracy, respectively, and $\Delta_{\mathcal{T}_{Acc}-\mathcal{S}_{Acc}}$ denotes the difference between the Teacher and Student accuracies. Also, [a], [b], [c] and [d] denote results produced by [35], [11], [3], and our implementation, respectively.

Bank) [3], *Ours-1* (Memory Bank) demonstrates consistent improvement across all the datasets. Similarly, compared to PRE-DFKD (Generative Replay) [2], utilizing the same VAE decoder architecture as the generative replay, we observe a similar trend for *Ours-2* (Generative Replay).

Moreover, in Figure 4, we visualize the learning curves of the student networks trained on multiple datasets. We plot the test accuracy of the student at the end of each training epoch. The proposed method exhibits significant improvement in the learning evolution and the peak accuracies achieved, suggesting that the proposed approach can retain the knowledge from previously encountered samples as the learning progresses. However, on Tiny-ImageNet, with Generative replay, we did not observe substantial improvements; we conjecture that this may be due to the complexity of the dataset, and the inability to capture crucial samples as replay for the complex dataset, for a large number of epochs. Also, with Generative Replay we sometimes faced difficulty in training a VAE on a stream of synthetic samples (especially for complex dataset like Tiny-ImageNet) as it suffers due to the distribution drift of its own.

Additionally, we emphasize that we do not strive toward achieving state-of-the-art student classification accuracy (requiring extensive hyper-parameter tuning) in the DFKD setting, but verify the viability of our hypothesis of retaining the previously acquired knowledge while learning on new samples. Nonetheless, we observe that our method improves upon the student classification accuracy on CIFAR100 [16] compared to the contemporary works and the current state-of-the-art [2] with the ResNet-34 [14] ($\mathcal{T}$) and ResNet-18 [14] ($\mathcal{S}$) setup, as shown in Table 2. Additionally, since previous works use the same teacher network with different test accuracies, we also report the teacher accuracies of the respective methods used to distill the knowledge to the student. Nonetheless, we also compute the Teacher-Student accuracy difference ($\Delta_{\mathcal{T}_{Acc}-\mathcal{S}_{Acc}}$) to assess the distance of the student from its teacher in terms of classification accuracy.

Interpreting the Result: Because of the proposed student update strategy, we observe a global monotonicity in the student's learning evolution which the existing approaches with naive replay [2, 3] claim, but do not warrant (described in Section 3.3). The global monotonicity in the learning evolution encompasses crucial advantages. For example, when the validation data is unavailable, the developer cannot assess the student's accuracy and identify the best parameters for the student. In such a scenario, the final accuracy is dependent on the random termination epoch set by the developer. In other words, the ideal DFKD approach should sustain high accuracy via monotonically enhancing it during the course of distillation. Therefore, $\mu[\mathcal{S}_{Acc}]$ and $\sigma^2[\mathcal{S}_{Acc}]$ contribute as crucial metrics to asses the distillation method as opposed to the maximum accuracy (Acc_{max}), since the Acc_{max} value can be attained at any point of time prior to the termination, and can be misleading. The improvements in the monotonicity and the $\mu[S_{Acc}]$ and $\sigma^2[S_{Acc}]$ values of proposed method are evident from Table 1, Figure 4 and Figure 5.

Architecture Scalability: The proposed student update strategy is generic and is scalable across different neural network architecture families since the method is not constrained to a particular choice of architecture. From the ResNet-18 [14], WRN-16-1 [33] and WRN-16-2 [33] student learning curves (in Figure 4 and Figure 5), we observe our method's advantage on both the network architectures. Moreover, for large student network architectures (Deeper or Wider) that include complex layers, the proposed method efficiently handles the intricacies with regard to computing the Hessian and the Hessian-product, which becomes highly essential for cumbersome models.

Improvement across Replay Schemes: Furthermore, the proposed method is agnostic to the memory scheme employed for replay, as demonstrated by the experiments (in Table 1, Figure 4 and Figure 5) using a Memory Buffer and Generative-Replay, thus, rendering our method generalized to the choice of replay. In Table 1 and Figure 5, we can observe that the proposed method enhances the student's performance on both the replay schemes (Memory Bank and Generative Replay) used in the prior arts. Moreover, we experiment with different memory buffer sizes on WRN-16-1 [33] and WRN-16-2 [33] distillation (in Figure 5) and observe consistent and substantial improvements across different memory sizes. Here, the memory size is defined as the maximum number of pseudo-example batches that the bank can contain and each batch consists of randomly sampled 64 examples from $\hat{x}$.

GPU Memory Utilization: Moreover, our student update strategy brings in no practical memory overhead, compared to memory-based Adversarial DFKD methods. We observe only a minimal increase in the GPU memory usage of few MBs (≈ 40 MB) due to the higher order gradients computed as a part of the update on $\theta_{\mathcal{S}}$ through $\theta'_{\mathcal{S}}$. Moreover, we use a *single* gradient descent step to obtain $\theta'_{\mathcal{S}}$, which does not incur a large memory overhead. Thus, we do not opt for a first order approximation [24] of our method, which is much prevalent in the meta-learning literature.

5. Conclusion

Societal Impact: Similar to other DFKD methods, our method may be framed as an attack strategy to create clones of proprietary pre-trained models that are accessible online [30]. However, this work makes no such efforts and does not support such practices.

Summary: In this paper, we proposed a meta-learning inspired student update strategy for the Adversarial DFKD setting, that treats *Knowledge-Acquisition* and *Knowledge-Retention* as meta-train and meta-test, respectively. The proposed strategy substantially improves the learning evolution of the student network by implicitly aligning the *Knowledge-Retention* and the *Knowledge-Acquisition* tasks. The intended effect of having the gradient directions aligned is to obtain student parameters (θ_S) that have optimal performance on both $\mathcal{L}_{Acq}$ and $\mathcal{L}_{Ret}$. The conducted experiments on multiple datasets, network architectures, and replay schemes demonstrate the effectiveness, scalability and generalizability of the proposed strategy.

References

[1] Sravanti Addepalli, Gaurav Kumar Nayak, Anirban Chakraborty, and Venkatesh Babu Radhakrishnan. Degan: Data-enriching gan for retrieving representative samples from a trained classifier. In *AAAI*, 2020. 8

[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 *AAAI*, 2022. 2, 3, 4, 6, 7, 8, 11, 12, 13

[3] Kuluhan Binici, Nam Trung Pham, Tulika Mitra, and Karianto Leman. Preventing catastrophic forgetting and distribution mismatch in knowledge distillation via synthetic data. In *WACV*, 2022. 2, 3, 4, 6, 7, 8, 11

[4] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Dafl: Data-free learning of student networks. In *ICCV*, 2019. 1, 2, 5, 6, 7, 8, 13

[5] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *CVPR Workshops*, 2020. 2, 3, 5, 6, 7, 8, 13

[6] Yatin Dandi, Luis Barba, and Martin Jaggi. Implicit gradient alignment in distributed and federated learning. In *AAAI*, 2022. 2

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 44(7):3366–3385, 2021. 2

[8] Kien Do, 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. In *NeurIPS*, 2022. 3

[9] Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. Up to 100x faster data-free knowledge distillation. In *AAAI*, 2022. 3

[10] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. In *CVPR*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 13

[11] Gongfan Fang, Jie Song, Xinchao Wang, Chen Shen, Xingen Wang, and Mingli Song. Contrastive model inversion for data-free knowledge distillation. In *IJCAI*, 2021. 1, 7, 8, 13

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 5

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 8, 11, 13

[15] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 4, 12, 13

[16] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 4, 6, 7, 8, 12, 13

[17] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 4, 6, 7, 11, 12, 13

[18] Jingru Li, Sheng Zhou, Liangcheng Li, Xifeng Yan, Zhi Yu, and Jiajun Bu. How to teach: Learning data-free knowledge distillation from curriculum. *arXiv preprint arXiv:2208.13648*, 2022. 13

[19] Yuang Liu, Wei Zhang, and Jun Wang. Zero-shot adversarial quantization. In *CVPR*, 2021. 1

[20] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *NeurIPS*, 2019. 1, 2, 3, 8

[21] Gaurav Kumar Nayak, Konda Reddy Mopuri, and Anirban Chakraborty. Effectiveness of arbitrary transfer sets for data-free knowledge distillation. In *WACV*, 2021. 8

[22] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *ICML*, 2019. 1, 8

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 4, 6, 7, 11, 12

[24] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 8

[25] Mert Bulent Sariyildiz and Ramazan Gokberk Cinbis. Gradient matching generative networks for zero-shot learning. In *CVPR*, 2019. 2

[26] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017. 1, 3

[27] Yuge Shi, Jeffrey Seely, Philip Torr, Siddharth N, Awni Han- nun, Nicolas Usunier, and Gabriel Synnaeve. Gradient match- ing for domain generalization. In *ICLR*, 2022. 2

[28] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017. 2, 3, 4

[29] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *IJCNN*, 2020. 1

[30] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. In *CVPR*, 2021. 9

[31] 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 deepin- version. In *CVPR*, 2020. 1, 8, 13

[32] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *NeurIPS*, 2019. 8

[33] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 7, 8, 13

[34] Sergey Zagoruyko and Nikos Komodakis. Paying more atten- tion to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 3

[35] Haoran Zhao, Xin Sun, Junyu Dong, Milos Manic, Huiyu Zhou, and Hui Yu. Dual discriminator adversarial distillation for data-free model compression. *International Journal of Machine Learning and Cybernetics*, 2022. 8

A. Training Details:

Algorithm 1: Proposed DFKD method, with Memory-Buffer replay.

Input: $\mathcal{T}_{\theta_\mathcal{T}}, \mathcal{S}_{\theta_\mathcal{S}}, \mathcal{G}_{\theta_\mathcal{G}}, \mathcal{M}, \mathcal{E}_{max}, \mathcal{I}, g, \alpha_\mathcal{G}, s, \alpha, \alpha_\mathcal{S}, f$
Output: $\mathcal{S}_{\theta_\mathcal{S}}$
$\mathcal{E} = 1$
while $\mathcal{E} \leq \mathcal{E}_{max}$ **do**
    for $\mathcal{I}$ *iterations* **do**
        for g *iterations* **do**
            $z \sim \mathcal{N}(0, I)$
            $\mathcal{L}_\mathcal{G} \leftarrow -\mathcal{D}(\mathcal{T}(\mathcal{G}_{\theta_\mathcal{G}}(z)), \mathcal{S}(\mathcal{G}_{\theta_\mathcal{G}}(z))) + \mathcal{L}_\mathcal{P}(\mathcal{G}_{\theta_\mathcal{G}}(z))$
            $\theta_\mathcal{G} \leftarrow \theta_\mathcal{G} - \alpha_\mathcal{G} \nabla_{\theta_\mathcal{G}} \mathcal{L}_\mathcal{G}$
        end
        for s *iterations* **do**
            $z \sim \mathcal{N}(0, I)$
            $\hat{x} \leftarrow \mathcal{G}_{\theta_\mathcal{G}}(z)$
            Compute $\mathcal{L}_{Acq}(\theta_\mathcal{S})$ using $\hat{x}$
            $\mathcal{L}_\mathcal{S} \leftarrow \mathcal{L}_{Acq}(\theta_\mathcal{S})$
            if $\mathcal{M}$ *is not empty* **then**
                $\hat{x}_m \sim \mathcal{M}$
                $\theta'_\mathcal{S} \leftarrow \theta_\mathcal{S} - \alpha \nabla \mathcal{L}_{Acq}(\theta_\mathcal{S})$
                Compute $\mathcal{L}_{Ret}(\theta_\mathcal{S})$ and $\mathcal{L}_{Ret}(\theta'_\mathcal{S})$ using $\hat{x}_m$
                $\mathcal{L}_\mathcal{S} \leftarrow \mathcal{L}_\mathcal{S} + \mathcal{L}_{Ret}(\theta_\mathcal{S}) + \mathcal{L}_{Ret}(\theta'_\mathcal{S})$
            end
            $\theta_\mathcal{S} \leftarrow \theta_\mathcal{S} - \alpha_\mathcal{S} \nabla_{\theta_\mathcal{S}} \mathcal{L}_S$
        end
    end
    if $\mathcal{E} \mod f == 0$ **then**
        Update $\mathcal{M}$ with x_m^*, where, $x_m^* \subseteq \hat{x}$
    end
    $\mathcal{E} \leftarrow \mathcal{E} + 1$
end

A.1. Teacher Model Training Details

We train the ResNet-34 [14] teacher model for SVHN [23] and Tiny-ImageNet [17]. For SVHN we use the ResNet-34 model definition made available by Binci *et al.*[1] and for Tiny-ImageNet, we use the `torchvision` model definition from PyTorch[2]. To train the teacher models we use SGD optimizer with an initial learning rate of 0.1, momentum of 0.9 and a weight-decay of 5e-4, with a batch size of 128 for 400 epochs. Moreover, the learning rate is decayed at each iteration till 0, using cosine annealing.

A.2. Student Model Training Details

For fair comparisons, we use the same Generator ($\mathcal{G}$) network (shown in Table 3) for all the methods. Unless not explicitly specified, for MB-DFKD [3] and our method (w/ Memory Buffer), we maintain a memory buffer of size 10 and update the memory buffer at a frequency of $f = 5$, following previous work [3] (Algorithm 1). Also, for PRE-DFKD [2] and our method (w/ Generative Replay), we use the same VAE architecture (as in Table 3 (Decoder) and 4 (Encoder)), from [2], to transfer the pseudo samples as memory, and use the decoder part (same as the generator architecture in Table 3) to replay the learnt distribution, with the VAE update parameters of $f = 1$ and $s_{max}^{gp} = 4$ (Algorithm 2), following previous works [2]. For all the methods and datasets, we use SGD optimizer with a momentum of 0.9 and a variable learning rate ($\alpha_\mathcal{S}$) with cosine annealing starting from 1e-1 and annealing it at each epoch to 0 to optimize the student parameters ($\theta_\mathcal{S}$). For the one-step gradient descent, we use a learning rate (α) of 0.9. Furthermore, we use Adam optimizer with a learning rate ($\alpha_\mathcal{G}$) of 0.02 to optimize

---

[1] https://github.com/kuluhan/PRE-DFKD
[2] https://pytorch.org/

Algorithm 2: Proposed DFKD method, with Generative replay.

Input: $\mathcal{T}_{\theta_\mathcal{T}}, \mathcal{S}_{\theta_\mathcal{S}}, \mathcal{G}_{\theta_\mathcal{G}}, \mathcal{M}, \mathcal{E}_{max}, \mathcal{I}, g, \alpha_\mathcal{G}, s, \alpha, \alpha_\mathcal{S}, f, s^{gp}_{max}$

Output: $\mathcal{S}_{\theta_\mathcal{S}}$

$\mathcal{E} = 1$

while $\mathcal{E} \leq \mathcal{E}_{max}$ **do**

    for $\mathcal{I}$ *iterations* **do**

        for g *iterations* **do**

            $z \sim \mathcal{N}(0, I)$

            $\mathcal{L}_\mathcal{G} \leftarrow -\mathcal{D}(\mathcal{T}(\mathcal{G}_{\theta_\mathcal{G}}(z)), \mathcal{S}(\mathcal{G}_{\theta_\mathcal{G}}(z))) + \mathcal{L}_\mathcal{P}(\mathcal{G}_{\theta_\mathcal{G}}(z))$

            $\theta_\mathcal{G} \leftarrow \theta_\mathcal{G} - \alpha_\mathcal{G} \nabla_{\theta_\mathcal{G}} \mathcal{L}_\mathcal{G}$

        end

        for s *iterations* **do**

            $z \sim \mathcal{N}(0, I)$

            $\hat{x} \leftarrow \mathcal{G}_{\theta_\mathcal{G}}(z)$

            Compute $\mathcal{L}_{Acq}(\theta_\mathcal{S})$ using $\hat{x}$

            $\mathcal{L}_\mathcal{S} \leftarrow \mathcal{L}_{Acq}(\theta_\mathcal{S})$

            $\hat{x}_m \sim \mathcal{M}$

            $\theta'_\mathcal{S} \leftarrow \theta_\mathcal{S} - \alpha \nabla \mathcal{L}_{Acq}(\theta_\mathcal{S})$

            Compute $\mathcal{L}_{Ret}(\theta_\mathcal{S})$ and $\mathcal{L}_{Ret}(\theta'_\mathcal{S})$ using $\hat{x}_m$

            $\mathcal{L}_\mathcal{S} \leftarrow \mathcal{L}_\mathcal{S} + \mathcal{L}_{Ret}(\theta_\mathcal{S}) + \mathcal{L}_{Ret}(\theta'_\mathcal{S})$

            $\theta_\mathcal{S} \leftarrow \theta_\mathcal{S} - \alpha_\mathcal{S} \nabla_{\theta_\mathcal{S}} \mathcal{L}_S$

            $s^{gp} = 0$

            if $\mathcal{E} \mod f == 0$ *and* $s^{gp} \leq s^{gp}_{max}$ **then**

                Train $\mathcal{M}$ with $\hat{x}_m$ and x^*_m, where, $x^*_m \subseteq \hat{x}$

                $s^{gp} \leftarrow s^{gp} + 1$

            end

        end

    end

    $\mathcal{E} \leftarrow \mathcal{E} + 1$

end

---

the Generator ($\mathcal{G}$). We test all our methods primarily on SVHN [23], CIFAR10 [16], CIFAR100 [16], and Tiny-ImageNet [17] for 200, 200, 400, and 500 epochs ($\mathcal{E}_{max}$), respectively. Our experiments were run on a mixture of Nvidia RTX2080Ti (11GB) and RTX3090 (24GB) GPUs. However, all our experiments incured not more than 11.5 GB of VRAM.

Table 3. Generator Network ($\mathcal{G}$) and Generative Replay (VAE [15]) Decoder Architecture.

Output Size	Layers
1000	Noise ($z \sim \mathcal{N}(0, I)$)
$128 \times h/4 \times w/4$	*Linear, BatchNorm1D, Reshape*
$128 \times h/4 \times w/4$	*SpectralNorm (Conv (3×3)), BatchNorm2D, LeakyReLU*
$128 \times h/2 \times w/2$	*UpSample ($2\times$)*
$64 \times h/2 \times w/2$	*SpectralNorm (Conv (3×3)), BatchNorm2D, LeakyReLU*
$64 \times h \times w$	*UpSample ($2\times$)*
$3 \times h \times w$	*SpectralNorm (Conv (3×3)), TanH, BatchNorm2D*

B. Attribution of Existing Assets:

B.1. Code-Base:

The code-base used to experiment with proposed method is adapted from the GitHub[1] repository of Binci *et al.* [2].

Table 4. Generative Replay (VAE [15]) Encoder Architecture.

Output Size	Layers
$3 \times h \times w$	Input Example
$64 \times h \times w$	*SpectralNorm(Conv (3 × 3)), BatchNorm2D, LeakyReLU*
$128 \times h \times w$	*SpectralNorm(Conv (3 × 3)), BatchNorm2D, LeakyReLU*
$128 \times h/2 \times w/2$	*DownSample* $(0.5\times)$
$128 \times h/2 \times w/2$	*SpectralNorm(Conv (3 × 3)), BatchNorm2D*
$128 \times h/4 \times w/4$	*DownSample* $(0.5\times)$
$\{1000, 1000\}$	*Reshape, Linear*

B.2. Pre Trained Teacher Model

The CIFAR10 pretrained [16] Teacher models of ResNet-34 and WRN-40-2 [33] are used used from the GitHub[3] repository made available by Fang *et al.* [11]. For the ResNet-34 Teacher model, pretrained on CIFAR100 [16], we used the model made available by Binci *et al.*[1] [2].

C. Extended Results

In Figure 6, we visualize the Cumulative Mean Accuracies (%) across the training epochs with Buffer-based and Generative Replay. The plots in Figure 6 complement the ones shown in Figure 5 of the main manuscript.

Based on the similarity of the Tiny-ImageNet teacher accuracy ($\mathcal{T}_{Acc}$) of the methods proposed and reported by Li *et al.* [18], we compare our methods with the accuracies reported by them.

Method	Teacher Accuracy (%) ($\mathcal{T}_{Acc}$)	Student Accuracy (%) ($\mathcal{S}_{Acc}$)
ADI[a] [31]	61.47	6.00
CMI[a] [11]	61.47	1.85
DAFL[b] [4]	60.83	35.46
DFAD[b] [10]	60.83	19.60
DFQ[a] [5]	61.47	41.30
CuDFKD[a] [18]	61.47	43.42
Ours-1 (w/ Memory Bank)	60.83	47.96
Ours-2 (w/ Generative Replay)	60.83	49.88

Table 5. Classification accuracy (in %) of the student trained using various DFKD methods on Tiny-ImageNet [17] with ResNet-34 [14] as the teacher and ResNet-18 [14] as the student. [a] and [b] denote results obtained from [18] and our implementation, respectively.

Figure 6. Cumulative Mean Accuracy (%) evolution of Wide-ResNet (WRN) [33]. The WRN-16-1 (*top-row*), and WRN-16-2 (*bottom-row*) networks are distilled by a WRN-40-2 teacher network pre-trained on CIFAR10 ($\mathcal{T}_{Acc} = 94.87\%$). Each column represent the learning curves with the Buffer-based (with different memory buffer sizes) and Generative replay schemes. The proposed method is in Blue.

---

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

Lemma 1. *If $\mathcal{L}_{Ret}$ has Lipschitz Hessian, i.e., $\|\nabla^2\mathcal{L}_{Ret}(\theta_1) - \nabla^2\mathcal{L}_{Ret}(\theta_2)\| \leq \rho\|\theta_1 - \theta_2\|$ for some $\rho > 0$, then:*

$$\nabla\mathcal{L}_{Ret}(\theta + \phi_\theta) = \nabla\mathcal{L}_{Ret}(\theta) + \nabla^2\mathcal{L}_{Ret}(\theta)\phi_\theta + \mathcal{O}(\|\phi_\theta\|^2).$$

For instance, when $\phi_\theta = -\alpha\nabla\mathcal{L}_{Acq}(\theta)$, we have,

$$\nabla\mathcal{L}_{Ret}(\theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)) = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}(\theta) + \mathcal{O}(\alpha^2).$$

Proof. Applying the fundamental theorem of calculus to each component of $\mathcal{L}_{Ret}$, we have:

$$\nabla\mathcal{L}_{Ret}(\theta + \phi_\theta) = \nabla\mathcal{L}_{Ret}(\theta) + \nabla^2\mathcal{L}_{Ret}(\theta)\phi_\theta + \int_{k=0}^{1}(\nabla^2\mathcal{L}_{Ret}(\theta + k\phi_\theta) - \nabla^2\mathcal{L}_{Ret}(\theta))\phi_\theta dk. \tag{16}$$

Omitting the subscript Ret for brevity,

$$\implies \|\nabla\mathcal{L}(\theta + \phi_\theta) - (\nabla\mathcal{L}(\theta) + \nabla^2\mathcal{L}(\theta)\phi_\theta)\| = \|\int_{k=0}^{1}(\nabla^2\mathcal{L}(\theta + k\phi_\theta) - \nabla^2\mathcal{L}(\theta))\phi_\theta dk\| \tag{17}$$

$$\implies \|\nabla\mathcal{L}(\theta + \phi_\theta) - (\nabla\mathcal{L}(\theta) + \nabla^2\mathcal{L}(\theta)\phi_\theta)\| \leq \int_{k=0}^{1}\|(\nabla^2\mathcal{L}(\theta + k\phi_\theta) - \nabla^2\mathcal{L}(\theta))\phi_\theta\|dk \tag{18}$$

$$\implies \|\nabla\mathcal{L}(\theta + \phi_\theta) - (\nabla\mathcal{L}(\theta) + \nabla^2\mathcal{L}(\theta)\phi_\theta)\| \leq \int_{k=0}^{1}\rho\|k\phi_\theta\|.\|\phi_\theta\|dk \quad \text{from } \rho\text{-Lipschitzness} \tag{19}$$

$$\implies \|\nabla\mathcal{L}(\theta + \phi_\theta) - (\nabla\mathcal{L}(\theta) + \nabla^2\mathcal{L}(\theta)\phi_\theta)\| \leq \frac{\rho}{2}\|\phi_\theta\|^2. \tag{20}$$

$\square$

Theorem 1. *If $\theta' = \theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)$, denotes the one step gradient descent on θ with the objective $\mathcal{L}_{Acq}(\theta)$, where α is a scalar, and $\nabla\mathcal{L}_{Acq}(\theta)$ denotes the gradients of $\mathcal{L}_{Acq}$ at θ, then:*

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\nabla^2\mathcal{L}_{Ret}(\theta).\nabla\mathcal{L}_{Acq}(\theta) - \alpha\nabla^2\mathcal{L}_{Acq}(\theta).\nabla\mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2).$$

Proof. We have

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').\frac{\partial\theta'}{\partial\theta} \tag{21}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').\frac{\partial(\theta - \alpha\nabla\mathcal{L}_{Acq}(\theta))}{\partial\theta} \tag{22}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta').(I - \alpha\nabla^2\mathcal{L}_{Acq}(\theta)) \tag{23}$$

Using Lemma 1, we substitute the value of $\nabla\mathcal{L}_{Ret}(\theta')$, where $\theta' = \theta - \alpha\nabla\mathcal{L}_{Acq}(\theta)$ in (23), and obtain:

$$\frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \overbrace{(\nabla\mathcal{L}_{Ret}(\theta) + \nabla^2\mathcal{L}_{Ret}(\theta).\underbrace{(\theta' - \theta)}_{=-\alpha\nabla\mathcal{L}_{Acq}(\theta)} + \underbrace{\mathcal{O}(\|\theta' - \theta\|^2)}_{=\mathcal{O}(\alpha^2)})}^{=\nabla\mathcal{L}_{Ret}(\theta')}.(I - \alpha\nabla^2\mathcal{L}_{Acq}(\theta)) \tag{24}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) + \nabla^2\mathcal{L}_{Ret}(\theta).\underbrace{(\theta' - \theta)}_{=-\alpha\nabla\mathcal{L}_{Acq}(\theta)} - \alpha\nabla^2\mathcal{L}_{Acq}(\theta)\nabla\mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2) \tag{25}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}(\theta) - \alpha\nabla^2\mathcal{L}_{Acq}(\theta)\nabla\mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2) \tag{26}$$

$$\implies \frac{\partial\mathcal{L}_{Ret}(\theta')}{\partial\theta} = \nabla\mathcal{L}_{Ret}(\theta) - \alpha\underbrace{(\overbrace{\nabla^2\mathcal{L}_{Ret}(\theta)\nabla\mathcal{L}_{Acq}}^{Hessian\ Product-1} - \overbrace{\nabla^2\mathcal{L}_{Acq}\nabla\mathcal{L}_{Ret}}^{Hessian\ Product-2})}_{Gradient\ Matching} + \mathcal{O}.(\alpha^2) \tag{27}$$

14

Note that, Lemma 1 provides an efficient way to obtain $Hessian\ Product - 1$ (highlighted in (27)) by computing the gradient of $\mathcal{L}_{Ret}$ at θ', thus, eradicating the time and memory overhead of explicitly computing $Hessian\ Product - 1$. Hence, we have:

$$\frac{\partial \mathcal{L}_{Ret}(\theta')}{\partial \theta} = \nabla \mathcal{L}_{Ret}(\theta) - \alpha \nabla^2 \mathcal{L}_{Ret}(\theta) \nabla \mathcal{L}_{Acq}(\theta) - \alpha \nabla^2 \mathcal{L}_{Acq} \nabla \mathcal{L}_{Ret}(\theta) + \mathcal{O}(\alpha^2). \tag{28}$$

$\square$