Few-Shot Dataset Distillation via Translative Pre-Training

Songhua Liu Xinchao Wang*
National University of Singapore
songhua.liu@u.nus.edu, xinchao@nus.edu.sg

Abstract

*Dataset distillation aims at a small synthetic dataset to mimic the training performance on neural networks of a given large dataset. Existing approaches heavily rely on an iterative optimization to update synthetic data and multiple forward-backward passes over thousands of neural network spaces, which introduce significant overhead for computation and are inconvenient in scenarios requiring high efficiency. In this paper, we focus on **few-shot** dataset distillation, where a distilled dataset is synthesized with only a few or even a single network. To this end, we introduce the notion of **distillation space**, such that synthetic data optimized only in this specific space can achieve the effect of those optimized through numerous neural networks, with dramatically accelerated training and reduced computational cost. To learn such a distillation space, we first formulate the problem as a quad-level optimization framework and propose a bi-level algorithm. Nevertheless, the algorithm in its original form has a large memory footprint in practice due to the back-propagation through an unrolled computational graph. We then convert the problem of learning the distillation space to a first-order one based on image translation. Specifically, the synthetic images are optimized in an arbitrary but fixed neural space and then translated to those in the targeted distillation space. We pre-train the translator on some large datasets like ImageNet so that it requires only a limited number of adaptation steps on the target dataset. Extensive experiments demonstrate that the translator after pre-training and a limited number of adaptation steps achieves comparable distillation performance with state of the arts, with $\sim 15\times$ acceleration. It also exerts satisfactory generalization performance across different datasets, storage budgets, and numbers of classes.*

1. Introduction

Given an original dataset $\mathcal{T}$, Dataset distillation (DD) [41, 51, 35, 18, 47, 20, 22] aims at a much smaller synthetic dataset $\mathcal{S}$, such that the training effect of $\mathcal{S}$ on neural networks can match that of $\mathcal{T}$. Such techniques have received considerable attention to alleviate the heavy burden on storage, transmission, and training of deep learning models brought by large-scale datasets. Recent advances on DD have revealed that models trained with only a few synthetic samples, even 1 image per class in some cases, can retain most of the performance of those trained with real data [49, 50, 2, 28, 29, 40, 13, 52, 23, 21]. Thus, it becomes a research area receiving increasing attention from the aca-

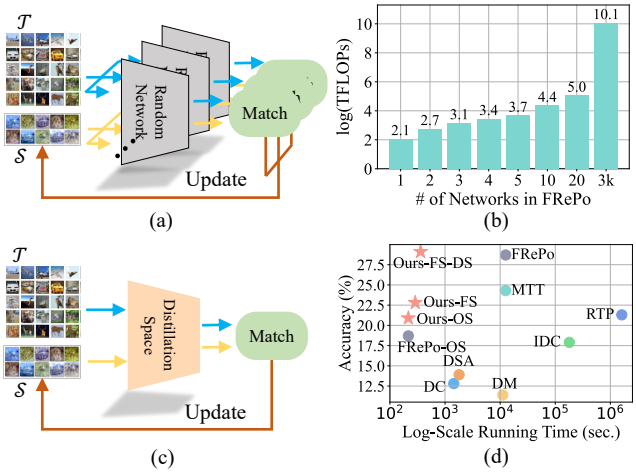


Figure 1. (a) Existing DD methods heavily rely on forward-backward processes through enormous neural networks. (b) More neural networks used in dataset distillation would result in more floating point operations (FLOPs) and higher latency. (c) We introduce the notation of *distillation space* for *few-shot dataset distillation* in this paper and aim to learn a neural network such that distilled data only optimized in this specific space can approximate the performance by multiple networks. (d) Given only *a few or even a single* network, our method achieves comparable performance with state-of-the-art baselines requiring enormous networks. Results in this figure are concerning models trained with 1 image per class distilled from CIFAR100 dataset. OS denotes one-shot DD using only one network, FS denotes few-shot DD with a limited number of adaptation steps, and DS denotes using down-sampled parameterization, *i.e.*, storing down-sampled distilled images with the same storage budget.

*Corresponding Author.

demic and industrial community recently as a potential solution to alleviate the concerns on the storage, transmission, pre-processing, and training effort for large datasets.

Nevertheless, as shown in Fig. 1(a), all existing methods on DD work iteratively: the synthetic dataset is optimized in an outer loop, and real and synthetic data are sent to a newly-initialized neural network in each iteration. Although they have achieved remarkable performance, such forward-backward passes in multiple neural networks can introduce extensive computational overhead to generate a distilled dataset, as illustrated in Fig. 1(b). As a result, the burdensome distillation procedure in existing methods brings significant latency. For instance, one of the state-of-the-art works by Deng *et al.* [5] requires ~ 18 days to distill only 1 image per class for CIFAR100 dataset [15] as shown in Fig. 1(d) and [47]. Even FRePo [52], with the most satisfactory efficiency, takes over 3 hours in the same setting requiring $500,000$ neural network steps. The long training process in current DD makes it inconvenient for scenarios requiring high efficiency, *e.g.*, processing streaming data.

Focusing on these drawbacks, in this paper, we propose *few-shot* dataset distillation and are interested in the performance of DD if only a few or even a single neural network is available for training. We first try a naive solution by merely adopting one neural network for the existing method FRePo [52] and visualize the accuracy and efficiency. As shown in Fig. 1(d), although the simple strategy improves the efficiency, the performance gap with the original version based on thousands of networks is significant, which suggests a trade-off between performance and the number of networks. A similar effect is also revealed by Cazenavette *et al.* [2], where the number of teacher training trajectories would affect the performance substantially. Thus, how to boost the performance by only a limited number of networks becomes an interesting question in few-shot DD.

Currently, the networks used in Fig. 1(d) are randomly and independently initialized, which makes us curious: whether there exists a dedicated neural network, such that the synthetic dataset distilled in it can achieve similar performance to those distilled in multiple ones. In this paper, we describe this concept through a quad-level definition termed as *distillation space*, as shown in Fig. 1(c), and propose a bi-level algorithm to find it, to optimize the error on real data, for networks trained with the synthetic dataset, distilled in this space.

However, the bi-level optimization requires back-propagating through an unrolled computational graph over multiple steps of gradient descent, which imposes an intensive GPU footprint. Due to the bottleneck on GPU memory, the number of steps for inner optimization is limited, which further limits the performance in practice. To reduce the complexity, we free the dependency of higher-order gradients via casting the problem as image translation, to make sure that the iterative optimization process is not nested in the computational graph: the synthetic dataset is distilled in a random but fixed neural network space and then translated to the desired space via a translator network.

Specifically, the translator is pre-trained on some large datasets like ImageNet [4]. In each iteration, a random subset is sampled. Distilled results after being translated from the original space are evaluated in some random networks, and the distillation loss is back-propagated to the translator to update its parameters. We find that the distillation space translator after pre-training can generalize well across different datasets, storage budgets, and numbers of classes. With only a few adaptation steps on the target dataset, the performance of distilled data after translation achieves can be on par with those distilled by multiple neural networks at $\sim 15\times$ faster, as shown in Fig. 1(d).

In summary, our contributions are listed as follows:

- Tailoring for scenarios requiring high efficiency, we are the first to study the problem of few-shot dataset distillation, where only a limited number of neural networks are available for DD, and introduce the quad-level definition of distillation space for it.
- An efficient translative modeling approach with a pre-training algorithm is proposed to learn an effective distillation space;
- Extensive experiments demonstrate that our method is capable of generating dataset distillation results with $\sim 15\times$ acceleration.

2. Related Works

Dataset distillation (DD) refers to distilling a small synthetic dataset $\mathcal{S}$ from a given large dataset $\mathcal{T}$, where $\mathcal{S}$ is optimized to have similar downstream training performance to $\mathcal{T}$. Unlike most existing techniques that focus on reducing the size of a model [6, 25, 45, 44, 7, 46, 12, 11, 10, 33], DD concentrates on learning compressed data to enhance learning efficiency. Wang *et al.* [41] firstly introduce the task and propose a meta-learning method to minimize the loss of meta-test on $\mathcal{T}$ for models meta-trained on $\mathcal{S}$. This scheme aligns with the evaluation protocol of DD: to pursue higher accuracy on the real test data for networks optimized by the synthetic ones. However, restricted by the GPU memory, the steps of meta-training in this framework are limited, which bottlenecks the performance [5]. The bi-level loop with a large number of inner steps also affects the efficiency negatively.

To address this issue, a series of works consider mapping real and synthetic samples to a feature space with some kernels, *e.g.*, NTK [28, 29], NNGP [23], and non-linear network [52, 24], and then perform kernel ridge regression

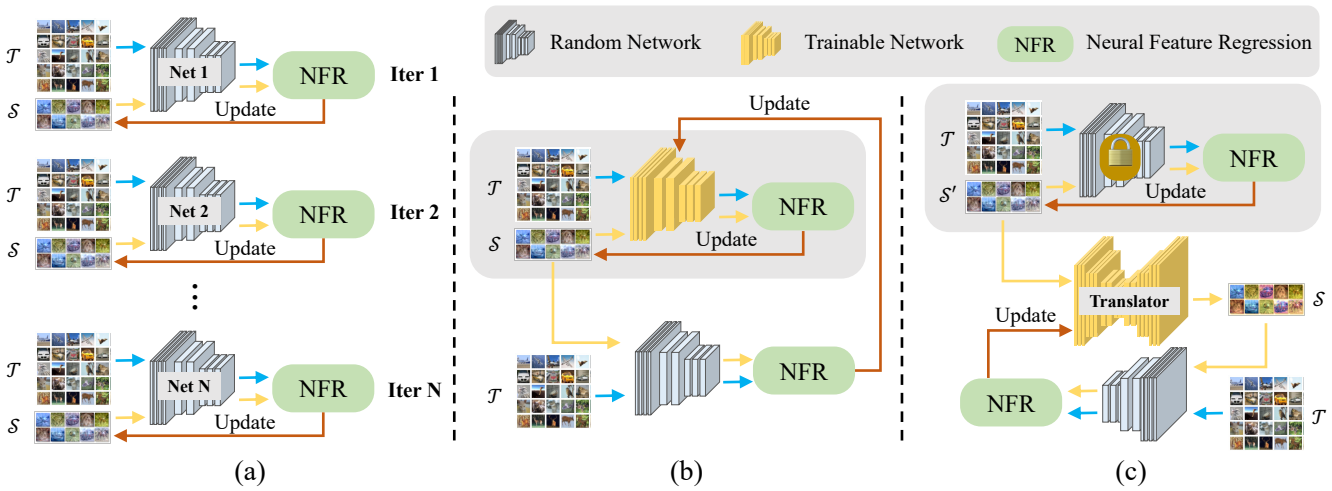


Figure 2. (a) Existing DD methods rely on an iterative training process, where a new network has to be initialized for forward and backward passes of the real dataset $\mathcal{T}$ and the synthetic dataset $\mathcal{S}$. (b) We desire to learn a specific distillation space, where $\mathcal{S}$ optimized only in such a space can minimize the error of neural feature regression in an arbitrary random network. The loop for optimizing $\mathcal{S}$ is nested in the computational graph for optimizing the distillation space. (c) We detach the loop for optimizing $\mathcal{S}$ for computational efficiency and convert the problem to a translative model, where $\mathcal{S}$ is distilled in an arbitrary but fixed neural space first and then translated to the desired space.

(KRR). Benefiting from the analytical solution of KRR, these methods can access the optimal parameters trained by synthetic datasets via direct computation without the necessity of higher-order gradients, which achieves favorable efficiency and accuracy at the same time. In this paper, the KRR-based formulation is also adopted as the main objective to reduce the complexity of meta-learning, thanks to its satisfactory efficiency.

In the literature of DD, matching-based methods are another major branch. For example, Zhao *et al.* [51] and a lot of following works [49, 13, 17, 9, 48] adopt gradient matching. Cazenavette *et al.* [2] propose MTT, which optimizes the distance between network parameters trained by $\mathcal{S}$ and $\mathcal{T}$ respectively from some cached initialization. Their insights lie in that gradients in networks or trained model parameters can represent the training effects of different datasets. Some works also consider matching the statistics of $\mathcal{S}$ and $\mathcal{T}$ in some neural networks [50, 40], which get rid of computing gradients of training networks via $\mathcal{S}$ and $\mathcal{T}$ and thus improves the efficiency.

No matter what objectives existing methods adopt, they all rely on an extensive outer optimization loop to update synthetic data, where $\mathcal{S}$ and $\mathcal{T}$ have to undergo forward-backward passes through thousands of networks. To make things worse, a series of works considering synthetic dataset parameterization, which tries to increase the number of images with the same storage budget, require even more such iterations [5, 19]. Motivated by these facts, we focus on the problem of few-shot dataset distillation in this paper where a synthetic dataset is distilled given only a few or even a single neural network. Given that naively adopting only a few random networks results in inferior performance as shown in Fig. 1(d), we introduce the concept of distillation space, such that the performance of synthetic data distilled in this space can match those in enormous networks. Dedicated to finding such a space to benefit dataset distillation to reduce the number of networks required for acceleration while retaining the performance as much as possible, a novel translative model with a pre-training approach is proposed to this end. To our best knowledge, this is the first work focusing on few-shot learning and pre-training in DD.

3. Methods

In this section, we delve into details about our methods for few-shot dataset distillation, where real and synthetic datasets can be processed by only a few neural networks. We first provide a brief introduction of current mainstream solutions for DD, based on which we propose the quad-level definition of *distillation space* in formal and a basic bi-level algorithm to learn the space. For computational efficiency, we then convert the original formulation of learning the distillation space to a translative problem and propose an alternative algorithm, backed up by theoretical analysis. Finally, we explain the paradigm of pre-training for few-shot learning to increase the transferability of the learned distillation space across various datasets, the number of classes, and the storage budgets of synthetic datasets.

3.1. Preliminary

Dataset distillation (DD) targets a small synthetic dataset $\mathcal{S}$ such that the performance on downstream training with $\mathcal{S}$ can match that with the original dataset $\mathcal{T}$. The seminal work by Wang *et al.* [41] proposes a meta-learning ap-

Algorithm 1 Bi-Level Optimization for Distillation Space.

Input: $\mathcal{T}$: Target Dataset; Θ: Distribution for Initializing Neural Networks; T: Number of Inner Update Steps; α: Learning Rate for Inner Optimization; η: Learning Rate for Outer Optimization.

Output: θ: Distillation Space after Optimization.

Initialize $\theta \sim \Theta$;
for $i = 1, 2, 3, \dots$ **do**
  Initialize $\mathcal{S}$ with random samples from $\mathcal{T}$;
  $X_{\mathcal{T}}^{\theta} \leftarrow f_\theta(X_{\mathcal{T}})$;
  for $1 \leq t \leq T$ **do**
    $w_{\mathcal{S},\theta}^{*} \leftarrow f_\theta(X_{\mathcal{S}})^\top (f_\theta(X_{\mathcal{S}}) f_\theta(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
    $X_{\mathcal{S}} \leftarrow X_{\mathcal{S}} - \alpha \nabla_{X_{\mathcal{S}}} \|X_{\mathcal{T}}^{\theta} w_{\mathcal{S},\theta}^{*} - Y_{\mathcal{T}}\|^2$;
    $Y_{\mathcal{S}} \leftarrow Y_{\mathcal{S}} - \alpha \nabla_{Y_{\mathcal{S}}} \|X_{\mathcal{T}}^{\theta} w_{\mathcal{S},\theta}^{*} - Y_{\mathcal{T}}\|^2$;
  end for
  Randomly sample $\theta' \sim \Theta$;
  $X_{\mathcal{T}}^{\theta'} \leftarrow f_{\theta'}(X_{\mathcal{T}})$;
  $w_{\mathcal{S},\theta'}^{*} \leftarrow f_{\theta'}(X_{\mathcal{S}})^\top (f_{\theta'}(X_{\mathcal{S}}) f_{\theta'}(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
  $\theta \leftarrow \theta - \eta \nabla_\theta \|X_{\mathcal{T}}^{\theta'} w_{\mathcal{S},\theta'}^{*} - Y_{\mathcal{T}}\|^2$;
end for

proach, which optimizes the loss on $\mathcal{T}$ for models trained with $\mathcal{S}$ and backpropagates the gradients through multiple training steps to update $\mathcal{S}$:

$$\begin{aligned} \mathcal{S}^* &= \arg\min_{\mathcal{S}} \mathbb{E}_{\theta^{(0)} \sim \Theta}[l(\mathcal{T}; \theta^{(T)})], \\ \theta^{(t)} &= \theta^{(t-1)} - \alpha \nabla l(\mathcal{S}; \theta^{(t-1)}), \end{aligned} \tag{1}$$

where Θ denotes a distribution for initialization, T is the number of training steps with $\mathcal{S}$ and $1 \leq t \leq T$, α is the learning rate, and l is the loss function, *e.g.*, cross-entropy loss for classification problems.

The bi-level optimization algorithm indicated in Eq. 1 relies on backpropagating through an unrolled computational graph of inner optimization steps, which makes it inefficient in both time and memory. Nguyen *et al.* [28, 29] thus consider the problem of kernel ridge regression (KRR), where the analytical optimal solution gets rid of the necessity to optimize the model with $\mathcal{S}$ for multiple nested steps. For more satisfactory accuracy and efficiency, the spirit is further extended by following works [52, 23], which map a batch of n samples X to a feature space with a random neural network and then perform KRR. Denote the function of neural network parameterized by θ as f_θ and the label matrix corresponding to $X \in \mathbb{R}^{n \times d}$ as $Y \in \mathcal{R}^{n \times c}$, where d and c are the dimensions of sample and label respectively, *i.e.*, $h \times w \times 3$ and the number of classes in the RGB image classification problem, the objective can be formulated as:

$$\begin{aligned} X_{\mathcal{S}}^*, Y_{\mathcal{S}}^* &= \arg\min_{X_{\mathcal{S}}, Y_{\mathcal{S}}} \mathbb{E}_{\theta \sim \Theta}[\|f_\theta(X_{\mathcal{T}}) w_{\mathcal{S},\theta}^{*} - Y_{\mathcal{T}}\|^2], \\ w_{\mathcal{S},\theta}^{*} &= f_\theta(X_{\mathcal{S}})^\top (f_\theta(X_{\mathcal{S}}) f_\theta(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}. \end{aligned} \tag{2}$$

This technique is known as neural feature regression (NFR) [52]. In each training iteration, a new random neural network is initialized for solving the problem in Eq. 2, as shown in Fig. 2(a).

3.2. Learning a Distillation Space

As illustrated in Fig. 1(a), the iterative optimization framework shown in Fig. 2(a) involves forward and backward processes through thousands of networks and introduces significant computational overhead. In this work, we thus focus on the problem of DD using only a few or even one network. Given the results in Fig. 1(b) that simply adopting a random network yields limited performance, we are interested in learning a specific network such that $\mathcal{S}$ distilled in it can achieve performance on par with that distilled in multiple networks, *i.e.*, minimize the error on $\mathcal{T}$ for an arbitrary model trained with $\mathcal{S}$. Formally, we term such a network as *distillation space* in the following definition.

Definition 1 (Distillation Space)**.** *A distillation space for a distribution of network Θ and a target dataset $\mathcal{T}$ is a neural network function f parameterized by θ^* which satisfies:*

$$\begin{aligned} \theta^* &= \arg\min_{\theta^{(0)}} \mathbb{E}_{\theta^{(0)'} \sim \Theta}[l(\mathcal{T}; \theta^{(T)'})], \\ \theta^{(t)'} &= \theta^{(t-1)'} - \alpha' \nabla l(\mathcal{S}^*; \theta^{(t-1)'}), \\ \mathcal{S}^* &= \arg\min_{\mathcal{S}} l(\mathcal{T}; \theta^{(T)}), \\ \theta^{(t)} &= \theta^{(t-1)} - \alpha \nabla l(\mathcal{T}; \theta^{(t-1)}). \end{aligned} \tag{3}$$

In practice, to find an effective distillation space, we first simplify the quad-level definition via NFR in Eq. 2, which derives a bi-level optimization problem:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathbb{E}_{\theta' \sim \Theta}[\|f_{\theta'}(X_{\mathcal{T}}) w_{\mathcal{S}^*,\theta'}^{*} - Y_{\mathcal{T}}\|^2], \\ w_{\mathcal{S}^*,\theta'}^{*} &= f_{\theta'}(X_{\mathcal{S}}^*)^\top (f_{\theta'}(X_{\mathcal{S}}^*) f_{\theta'}(X_{\mathcal{S}}^*)^\top)^{-1} Y_{\mathcal{S}}^*, \\ X_{\mathcal{S}}^*, Y_{\mathcal{S}}^* &= \arg\min_{X_{\mathcal{S}}, Y_{\mathcal{S}}} \|f_\theta(X_{\mathcal{T}}) w_{\mathcal{S},\theta}^{*} - Y_{\mathcal{T}}\|^2, \\ w_{\mathcal{S},\theta}^{*} &= f_\theta(X_{\mathcal{S}})^\top (f_\theta(X_{\mathcal{S}}) f_\theta(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}. \end{aligned} \tag{4}$$

Eq. 4 indicates a bi-level meta-learning algorithm as shown in Fig. 2(b) and Alg. 1. In each inner iteration, $\mathcal{S}$ is meta-trained via NFR in the current neural space f_θ. $\mathcal{S}$ after a whole inner loop is meta-tested in a random neural network $f_{\theta'}$ and the loss of NFR is back-propagated to update θ.

3.3. Learning a Translation to a Distillation Space

Although Alg. 1 can be an effective algorithm to learn a valid distillation space, the back-propagation process through a nested computational graph is inefficient in GPU memory. The memory complexity is proportionate to the number of inner optimization steps T to obtain the optimal synthetic dataset in the current space, which is crucial

for the final performance [5] but withheld by the limited GPU memory. For instance, in a 40G A100 GPU, T in Alg. 1 can be set as 16 at most, which is too small to get convergence for inner optimization. To alleviate this drawback, we consider an alternative approach to learn a distillation space. Specifically, we detach the inner optimization from the computational graph by converting the problem to a translative model, based on the following proposition.

Proposition 1. *For a target dataset $X_{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}} \times d}$ and $Y_{\mathcal{T}} \in \mathbb{R}^{n_{\mathcal{T}} \times c}$ with $n_{\mathcal{T}} \geq d$, assume that $X^*_{\mathcal{S},0}$ and $Y^*_{\mathcal{S},0}$ satisfy $\|f_\theta(X_{\mathcal{T}})w^*_{\mathcal{S},\theta} - Y_{\mathcal{T}}\|^2 = 0$, where $w^*_{\mathcal{S},\theta} = f_\theta(X^*_{\mathcal{S},0})^\top (f_\theta(X^*_{\mathcal{S},0}) f_\theta(X^*_{\mathcal{S},0})^\top)^{-1} Y^*_{\mathcal{S},0}$ and f_θ is a known linear transformation parameterized by a full-rank matrix $\theta \in \mathbb{R}^{d \times d'}$, $d \leq d'$. Similarly, let f_{θ^*} be a desired linear distillation space and $\theta^* \in \mathbb{R}^{d \times d''}$, $d \leq d'' \leq n_{\mathcal{T}}$. The distilled dataset $X^*_{\mathcal{S},*}$ and $Y^*_{\mathcal{S},*}$ in this space satisfy $\|f_{\theta^*}(X_{\mathcal{T}})w^*_{\mathcal{S},\theta^*} - Y_{\mathcal{T}}\|^2 = 0$, where $w^*_{\mathcal{S},\theta^*} = f_{\theta^*}(X^*_{\mathcal{S},*})^\top (f_{\theta^*}(X^*_{\mathcal{S},*}) f_{\theta^*}(X^*_{\mathcal{S},*})^\top)^{-1} Y^*_{\mathcal{S},*}$. There exists a translation function G for $X^*_{\mathcal{S},0}$ and $Y^*_{\mathcal{S},0}$ such that $X^*_{\mathcal{S},*}, Y^*_{\mathcal{S},*} = G(X^*_{\mathcal{S},0}, Y^*_{\mathcal{S},0})$, whose parameters are not dependent on the target dataset $X_{\mathcal{T}}$ and $Y_{\mathcal{T}}$.*

Proof. Due to the linearity of f_θ and f_{θ^*}, we can rewrite the given conditions as:

$$X_{\mathcal{T}}\theta w^*_{\mathcal{S},\theta} - Y_{\mathcal{T}} = \mathbf{0}, \tag{5}$$

$$X_{\mathcal{T}}\theta^* w^*_{\mathcal{S},\theta^*} - Y_{\mathcal{T}} = \mathbf{0}, \tag{6}$$

Since $n_{\mathcal{T}} \geq d$ for $X_{\mathcal{T}}$, pre-multiplying terms in Eq. 5 by the Moore–Penrose inverse $X^\dagger_{\mathcal{T}}$, we have

$$X^\dagger_{\mathcal{T}} Y_{\mathcal{T}} = \theta w^*_{\mathcal{S},\theta}. \tag{7}$$

Similarly, through Eq. 6, we have

$$w^*_{\mathcal{S},\theta^*} = (X_{\mathcal{T}}\theta^*)^\dagger Y_{\mathcal{T}} = \theta^{*\dagger} X^\dagger_{\mathcal{T}} Y_{\mathcal{T}} = \theta^{*\dagger}\theta w^*_{\mathcal{S},\theta}, \tag{8}$$

where the first, second, and third equalities are due to $n_{\mathcal{T}} \geq d''$, $d'' \geq d$, and Eq. 7 respectively. We can observe in Eq. 8 that $w^*_{\mathcal{S},\theta^*}$ in the desired distillation space is not dependent on $\mathcal{T}$. Thus, the optimal $X^*_{\mathcal{S},*}$ and $Y^*_{\mathcal{S},*}$ resulting the optimal KRR parameter can be obtained given only $X^*_{\mathcal{S},0}$ and $Y^*_{\mathcal{S},0}$ by some translation function G, which implicitly includes the knowledge of the distillation space. □

Prop. 1 indicates that it is possible to approach the distillation space by translating samples from some space rather than learning θ^* directly, motivated by which we propose to learn a translation from one another space to the distillation space. The main workflow is similar to Alg. 1: synthetic data are first distilled in a random but fixed network in an inner loop and then translated to final results, as shown in Fig. 2(c). The translator is optimized so that results after translation are equipped with the property of those produced in a distillation space: to minimize the NFR error in any random network. Since gradients are only required to back propagated to the translator rather than the whole inner loop, the memory complexity is irrelevant to the number of inner optimization steps. Thus, the translative model is a memory-efficient algorithm where only first-order gradients are necessary.

Algorithm 2 Pre-training a Distillation Space Translator.

Input: $\mathcal{Z}$: A Large Dataset; Θ: Distribution for Initializing Neural Networks; θ: An Arbitrary Random Neural Network; T: Number of Update Steps for Synthetic Data; α: Learning Rate for Synthetic Data; η: Learning Rate for Translator.

Output: ω: A Pre-trained Translator.

1: Initialize ω randomly;
2: **for** $i = 1, 2, 3, \ldots$ **do**
3: Randomly choose a subset of data $\mathcal{T}$ from $\mathcal{Z}$;
4: Initialize $\mathcal{S}'$ with random samples from $\mathcal{T}$;
5: $X^\theta_{\mathcal{T}} \leftarrow f_\theta(X_{\mathcal{T}})$;
6: **for** $1 \leq t \leq T$ **do**
7: $w^*_{\mathcal{S}',\theta} \leftarrow f_\theta(X_{\mathcal{S}'})^\top (f_\theta(X_{\mathcal{S}'}) f_\theta(X_{\mathcal{S}'})^\top)^{-1} Y_{\mathcal{S}'}$;
8: $X_{\mathcal{S}'} \leftarrow X_{\mathcal{S}'} - \alpha \nabla_{X_{\mathcal{S}'}} \|X^\theta_{\mathcal{T}} w^*_{\mathcal{S}',\theta} - Y_{\mathcal{T}}\|^2$;
9: $Y_{\mathcal{S}'} \leftarrow Y_{\mathcal{S}'} - \alpha \nabla_{Y_{\mathcal{S}'}} \|X^\theta_{\mathcal{T}} w^*_{\mathcal{S}',\theta} - Y_{\mathcal{T}}\|^2$;
10: **end for**
11: $X_{\mathcal{S}}, Y_{\mathcal{S}} = G_\omega(X_{\mathcal{S}'}, Y_{\mathcal{S}'})$;
12: Randomly sample $\theta' \sim \Theta$;
13: $X^{\theta'}_{\mathcal{T}} \leftarrow f_{\theta'}(X_{\mathcal{T}})$;
14: $w^*_{\mathcal{S},\theta'} \leftarrow f_{\theta'}(X_{\mathcal{S}})^\top (f_{\theta'}(X_{\mathcal{S}}) f_{\theta'}(X_{\mathcal{S}})^\top)^{-1} Y_{\mathcal{S}}$;
15: $\omega \leftarrow \omega - \eta \nabla_\omega \|X^{\theta'}_{\mathcal{T}} w^*_{\mathcal{S},\theta'} - Y_{\mathcal{T}}\|^2$;
16: **end for**

3.4. Pre-training a Distillation Space Translator

We expect that the distillation space optimized in a dataset is transferable across different datasets, storage budgets for synthetic data, and the number of classes. To tackle these challenges on transferability, in this paper, we consider pre-train a translator on some large dataset like ImageNet [4], so that it provides a satisfactory initialization and only requires a small number of adaptation steps for a target dataset, like the typical few-shot learning [42, 39, 38, 30]. The algorithm is shown in Alg. 2. In each training iteration i, we sample a subset from the large dataset and perform a gradient-decent step with the strategy mentioned in Sec. 3.3. Upon a translator is pre-trained, it can serve as a strong initialization for the adaptation process to a new dataset, which behaves identically to Line 11 to 15 of Alg. 2.

4. Experiments

In this section, we conduct experiments to validate the effectiveness of the proposed method for few-shot DD. We first introduce our experiment settings and implement de-

Hyper-Parameter	Denotation	Value
T	Number of Update Steps for Synthetic Data	3,000
α	Learning Rate for Synthetic Data	1e-3
η	Learning Rate for Translator in Pre-Training	1e-5
N	Number of Cached Subsets in Pre-Training	5,000
C_{max}	Maximal Number of Classes in a Subset in Pre-Training	100
C_{min}	Minimal Number of Classes in a Subset in Pre-Training	10
M	Maximal Number of Images in a Subset in Pre-Training	4,000
I_{max}	Maximal Number of Images in a Synthetic Dataset in Pre-Training	1,000
B	Batch Size in Pre-Training	4
S	Number of Pre-Training Steps	200,000
S'	Number of Adaptation Steps for Translator	1,000
η'	Learning Rate for Translator in Adaptation	1e-4

Table 1. List of hyper-parameters and their values.

tails of the method. Then, we compare our method with state-of-the-art solutions of DD in terms of both accuracy and efficiency. To illustrate how the proposed method works, we also provide an in-depth analysis of the method with a variety of baselines. Finally, we conduct experiments on continual learning, an important use case for DD, to further demonstrate the practical value of our method.

4.1. Settings and Implementing Details

In few-shot dataset distillation, a real dataset $\mathcal{T}$ can be processed by only a few neural networks to generate a synthetic dataset $\mathcal{S}$. In our method, $\mathcal{S}$ is firstly optimized in an arbitrary but fixed neural network by T steps, with the learning rate α. The results are then processed by a translator to generate the final synthetic dataset. The translator adopts an auto-encoder architecture with 3 `Conv-BatchNorm-ReLU` blocks, and the detailed configuration can be found in the supplement. Notably, due to the fully-convolutional structure, the pre-trained translator can also be adapted to datasets with different resolutions beyond the one used during pre-training. Please refer to the supplement for details. The translator is pre-trained on the ImageNet dataset by S steps with the learning rate η and then adapted on the target dataset by a small number of iterations S', with a learning rate η'. Some hyper-parameters are summarized in Tab. 1. Other hyper-parameters follow the default setting in the PyTorch [31] implementation of FRePo [52].

Pre-Training: The key design of the translative pre-training pipeline lies in the separation of two stages: dataset distillation in an arbitrary but fixed neural network and the translation to the desired space. In practice, instead of directly following the pipeline in Alg. 2 that nests the two steps in one loop, the two stages are conducted independently in the interest of better training efficiency. Results distilled in the pre-defined network are cached for multiple random subsets so that they can be loaded directly and used repeatedly in the pre-training stage, which improves the training efficiency. The detailed algorithms for the caching stage and the pre-training stage can be found in the supplement.

The caching stage aims to generate distilled results for multiple subsets. For sampling a subset, we first randomly choose the number of classes ranging from C_{min} to C_{max} and then randomly select the corresponding number of classes from the 1,000 ImageNet classes. For each select class, a certain number of random real images are sampled uniformly for each selected class such that the total number of images is M at most. The data augmentation technique DSA [49] is applied in the caching stage to mimic the cases of larger datasets. The maximal number of synthetic images is I_{max} and they are initialized with the selected real data. After distillation, indices of selected real samples and the distilled dataset are cached in the disk for future use in the pre-training stage. This process can be deployed to multiple GPUs in parallel, since the distillation for each subset is independent.

In the pre-training stage, cached results in the previous stage are loaded. The NFR loss is calculated in a randomly initialized networks over a distribution in each iteration. Like the caching stage, pre-cached subsets in a batch can also be distributed to multiple GPUs in parallel. In this work, the pre-training is conducted on a single 40G A100 GPU, which takes roughly 2 days for the 200,000 steps.

Adaptation: Given a pre-trained translator, it requires a small number of adaptation steps in general for the target dataset. The detailed adaptation algorithm can be found in the supplement. Intuitively, the procedure is consistent with the two-stage process in pre-training: to generate distilled results in a pre-defined network and then use the translator for the final results. The only difference with the pre-training stage is that real data come from the target dataset. By default, the set of IPC for adaptation only contains the target IPC. More IPCs can also be included in this set to

Dataset	IPC	Metric	DD	RTP	DC	DSA	IDC	MTT	DM	FRePo	Ours	Ours-DS
CIFAR 10	1	Acc. (%)	$40.5_{\pm 0.8}$	$\underline{49.1}_{\pm 0.6}$	$28.3_{\pm 0.5}$	$28.8_{\pm 0.7}$	$36.7_{\pm 0.4}$	$46.3_{\pm 0.8}$	$26.0_{\pm 0.8}$	$46.8_{\pm 0.7}$	$44.0_{\pm 0.2}$	$\mathbf{51.4}_{\pm 0.4}$
		Time (h)	363.9	363.9	$\underline{0.1}$	$\underline{0.1}$	10.6	0.8	0.3	2.4	$\mathbf{0.02}_{\times 120.0}$	$\underline{0.1}_{\times 24.0}$
	10	Acc. (%)	$50.0_{\pm 0.5}$	$62.4_{\pm 0.4}$	$44.9_{\pm 0.5}$	$52.1_{\pm 0.5}$	$58.3_{\pm 0.4}$	$65.3_{\pm 0.7}$	$48.9_{\pm 0.6}$	$\underline{65.5}_{\pm 0.6}$	$59.2_{\pm 0.3}$	$\mathbf{66.0}_{\pm 0.3}$
		Time (h)	447.2	447.2	1.0	1.3	11.1	2.4	0.3	3.1	$\mathbf{0.04}_{\times 77.5}$	$\underline{0.2}_{\times 15.5}$
	50	Acc. (%)	-	$70.5_{\pm 0.4}$	$53.9_{\pm 0.5}$	$60.6_{\pm 0.5}$	$69.5_{\pm 0.3}$	$71.6_{\pm 0.2}$	$63.0_{\pm 0.4}$	$\underline{71.7}_{\pm 0.2}$	$66.7_{\pm 0.2}$	$\mathbf{71.8}_{\pm 0.1}$
		Time (h)	-	1377.8	4.4	5.3	12.8	3.8	$\underline{0.4}$	8.1	$\mathbf{0.12}_{\times 67.5}$	$1.3_{\times 6.2}$
CIFAR 100	1	Acc. (%)	-	$21.3_{\pm 0.6}$	$12.8_{\pm 0.3}$	$13.9_{\pm 0.3}$	$17.9_{\pm 0.2}$	$24.3_{\pm 0.3}$	$11.4_{\pm 0.3}$	$\underline{28.7}_{\pm 0.1}$	$22.8_{\pm 0.4}$	$\mathbf{29.1}_{\pm 0.4}$
		Time (h)	-	447.2	0.4	0.5	50.0	3.5	3.1	3.1	$\mathbf{0.08}_{\times 38.8}$	$\underline{0.1}_{\times 31.0}$
	10	Acc. (%)	-	$34.7_{\pm 0.5}$	$25.2_{\pm 0.3}$	$32.3_{\pm 0.3}$	$36.1_{\pm 0.4}$	$\underline{40.1}_{\pm 0.4}$	$29.7_{\pm 0.3}$	$\mathbf{42.5}_{\pm 0.2}$	$35.1_{\pm 0.2}$	$38.4_{\pm 0.1}$
		Time (h)	-	2520.8	10.6	11.4	68.9	4.2	3.1	10.4	$\mathbf{0.32}_{\times 32.5}$	$\underline{1.0}_{\times 10.4}$

Table 2. Comparisons with state-of-the-art methods of DD on test accuracy and training efficiency. DS denotes using down-sampled parameterization, *i.e.*, storing down-sampled distilled images with the same storage budget. The black subscript indicates the standard deviation over multiple evaluations. The red subscript denotes the times of acceleration compared with the baseline method FRePo [52]. The best and second best results in each setting are marked by **bold** and underline respectively.

increase the generalization ability of the translator across other unseen IPCs during adaptation, as shown in Sec. 4.3 and the supplement.

In this paper, we consider the widely-adopted benchmarks of CIFAR10 and CIFAR100 [15] with a resolution of 32×32 for the main evaluation. Results on datasets with higher resolutions and larger sizes can be found in the supplement. Following previous works [52, 5], we evaluate the proposed method on storage budgets of 1, 10, and 50 images per class (IPC) for CIFAR10 and 1 and 10 IPC for CIFAR100. Synthetic datasets are trained with 3-layer convolutional networks and evaluated on the same architecture. The cross-architecture performance will also be studied in Sec. 4.3. We report the average result and the standard deviation over 3 replicate evaluations for each setting.

4.2. Comparisons with State of the Arts

Here, we compare the proposed method for few-shot dataset distillation with state-of-the-art methods, including meta-learning-based methods in the seminal *DD* [41] and *remember-the-past (RTP)* [5], gradient-matching based methods in *DC*, *DSA*, and *IDC*, trajectory-matching based *MTT*, distribution-matching based *DM*, and KRR-based *FRePo* [52]. The accuracy and training efficiency results are shown in Tab. 2. The accuracy results are from original papers while the training time is estimated based on the time-per-iteration results evaluated on the same hardware and the total number of iterations in the official codes. The shape/format of synthetic images in all methods except "Ours-DS" is consistent with that of real images in original datasets. For the running time of our method, we report the total time required for distilling using one network and the adaptation. Time for pre-training is not included since it is only conducted once, and once pre-trained, the translator can be adapted for an infinite number of downstream datasets.

Through the results, we find that in the default parameterization, the proposed method within 1k adaptation steps falls about $3 \sim 7$ points behind the FRePo baseline but with $30 \sim 120\times$ acceleration due to the few-shot setting. Even compared with the currently most efficient algorithm, the speed of our method is still considerable. And the performance is at least comparable with other methods in terms of accuracy, with even dramatic acceleration, *e.g.*, the meta-learning-based DD and RTP. The superior efficiency makes our method applicable in scenarios where data cannot be held for a long time, like streaming data.

Additionally, in "Ours-DS", down-sampled synthetic images are alternatively stored with the same number of stored parameters, which is a simple yet effective method to improve the performance especially when the storage budget is small. This parameterization typically requires more iterations for the adaptation stages. For small synthetic datasets like 1 and 10 IPC, the number of steps is 6k, and for 50 IPC, the number is 15k. Nevertheless, our method can still result in $6 \sim 31\times$ acceleration compared with the FRePo baseline with even better performance.

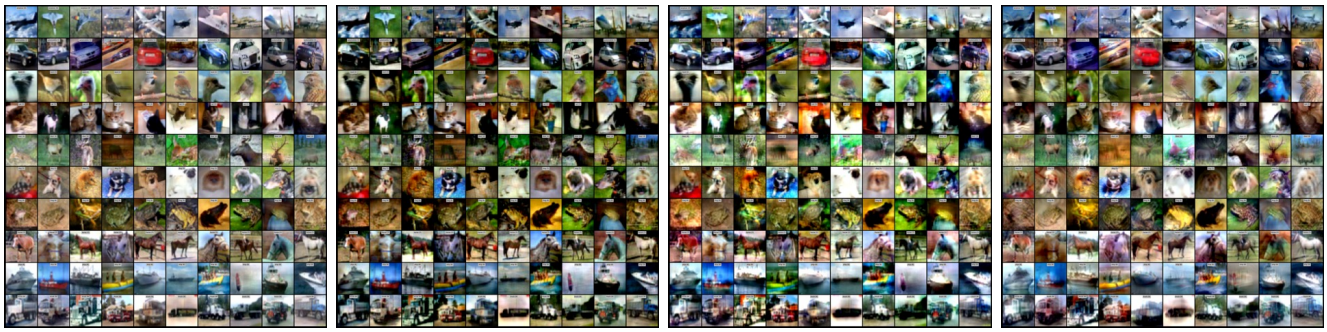
4.3. Ablation Studies

Baselines: To demonstrate the effectiveness of the translative pre-training and target adaptation strategies proposed in this paper, we consider a series of baselines for ablation studies, including:

- *Baseline*: a new neural network is sampled for each training iteration to process the real dataset and update synthetic data, which can be viewed as a benchmark given a sufficient number of networks;

Dataset	IPC	Baseline	1 Net	Bi-Level	w/o Pre-Train	w AE	w/o Ada	w Ada
CIFAR10	1	$41.2_{\pm 0.7}$	$35.3_{\pm 0.6}$	$40.8_{\pm 0.5}$	$36.1_{\pm 0.5}$	$40.6_{\pm 0.6}$	$39.5_{\pm 0.8}$	$44.0_{\pm 0.2}$
	10	$57.9_{\pm 0.7}$	$50.5_{\pm 0.8}$	$52.1_{\pm 0.5}$	$53.8_{\pm 0.1}$	$57.6_{\pm 0.3}$	$54.7_{\pm 0.8}$	$59.2_{\pm 0.3}$
	50	$65.2_{\pm 0.3}$	$65.1_{\pm 0.1}$	$60.7_{\pm 0.1}$	$59.7_{\pm 0.4}$	$63.5_{\pm 0.1}$	$65.8_{\pm 0.2}$	$66.7_{\pm 0.2}$
CIFAR100	1	$21.6_{\pm 0.6}$	$18.7_{\pm 0.1}$	$19.1_{\pm 0.1}$	$20.5_{\pm 0.2}$	$21.8_{\pm 0.3}$	$20.9_{\pm 0.7}$	$22.8_{\pm 0.4}$
	10	$33.8_{\pm 0.3}$	$31.7_{\pm 0.1}$	$31.0_{\pm 0.2}$	$30.7_{\pm 0.2}$	$27.0_{\pm 0.2}$	$33.0_{\pm 0.2}$	$35.1_{\pm 0.2}$

Table 3. Ablation studies of our method in different settings. The black subscript indicates the standard deviation over multiple evaluations.



An Arbitrary Net
Acc. 50.5%

Translated Results w/o Ada
Acc. 54.7%

Translated Results w Ada
Acc. 59.2%

FRePo
Acc. 65.5%

Figure 3. Qualitative analysis of results produced by our method for IPC 10 on CIFAR10. Zoom in for details.

- *1 Net*: synthetic data are distilled in a single network, which can be treated as a lower bound;
- *Bi-Level*: the bi-level algorithm described in Alg. 1;
- *w/o Pre-Train*: the target adaptation stage starts from a randomly initialized translator;
- *w AE*: the target adaptation stage starts from an auto-encoder trained with a reconstruction objective;
- *w/o Ada*: directly using the pre-trained translator without the adaptation stage;
- *w Ada*: the default method.

All the candidates are trained with 4k iterations. For those requiring adaptation, *i.e.*, w/o Pre-Train, w AE, and w Ada, we first distill a synthetic dataset using 1 network with 3k iterations and adapt the translator for another 1k steps. Note that distilling with a single network only requires one forward propagation pass of the real dataset. Thus, the running time of our method is less than that of the baseline given the same number of iterations. Results are shown in Tab. 3.

For one thing, it turns out that the default method in this paper consistently surpasses the benchmark baseline requiring enormous networks in most cases. The reason is that with the number of networks for NFR increasing, it would be more difficult for the training convergence of distilled datasets, especially in the few-shot case. Please refer to the supplement for details.

For another, the consistent improvement comparing results of *1 Net* and *w/o Ada* indicates that the pre-trained translator is capable of generating distilled results in a more favorable space. We visualize their results in the first two plots of Fig. 3, where the global tone is refined towards higher brightness contrast. Moreover, as shown in the 3rd plot, the adaptation stage injects more local details, which tries to improve the performance in a similar way performed by the state-of-the-art FRePo [52].

Further, as shown in both Tab. 3 and Fig. 4(a-b), the pre-trained model offers a more satisfactory initialization for the following adaptation stage, with lower initial loss, faster convergence, and higher accuracy.

At last, the bi-level algorithm often leads to inferior performance, which may be partially attributed to the insufficient number of inner steps restricted by GPU memory.

Generalization: We observe that one benefit of adapting the translator in a target dataset is that it is generalizable across different storage budgets for synthetic datasets. For instance, as shown in Fig. 4(c), the translator adapted on IPC 1 and 50 can also behave well on IPC 10.

Following previous works [51, 49, 50], we also evaluate the cross-architecture performance in Tab. 4 over AlexNet [16], VGG11 [37], and ResNet18 [8]. The conclusion is consistent with the above analysis.

4.4. Application: Continual Learning

Continual learning (CL) aims to learn a sequence of tasks where the training data of past tasks are unavailable when

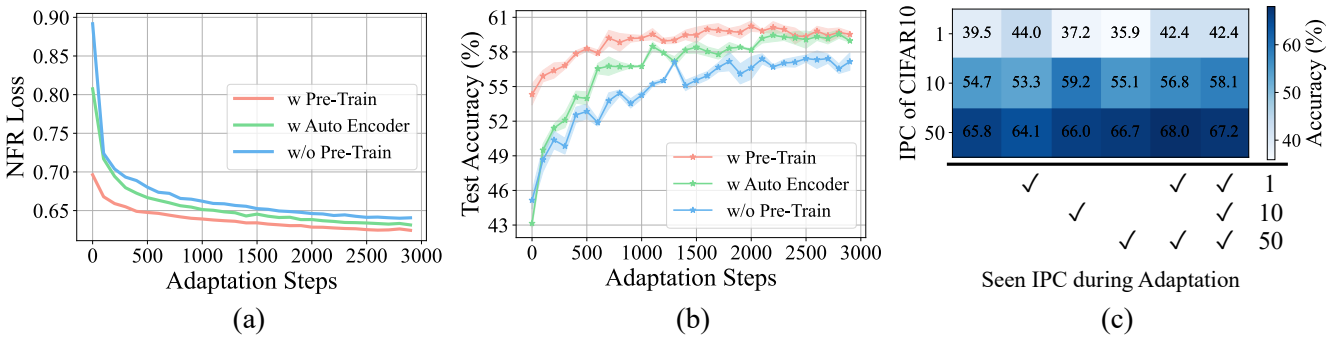


Figure 4. (a-b) NFR loss and test accuracy during adaption for CIFAR10 IPC 10 starting with the pre-trained translator by our algorithm, an auto-encoder, and a randomly initialized model. (c) Generalization performance of the translator on unseen IPC during adaptation.

Setting	ConvNet	AlexNet	VGG	ResNet
Baseline	58.8±0.2	**57.1**±0.5	41.1±0.3	38.1±0.9
1 Net	50.5±0.8	50.9±0.6	34.2±0.3	35.3±0.3
Bi-Level	52.1±0.5	50.0±0.4	37.4±0.5	34.7±0.5
w/o Pre-Train	53.8±0.1	56.2±0.3	39.3±0.7	38.2±0.7
w/o Ada	54.7±0.8	52.8±0.3	35.7±0.7	34.8±1.2
w Ada	**59.2**±0.3	56.8±0.2	**43.0**±0.7	**40.8**±0.5

Table 4. Cross-architecture performance of our method in different settings. Results on CIFAR10 with 10 IPC are shown here.

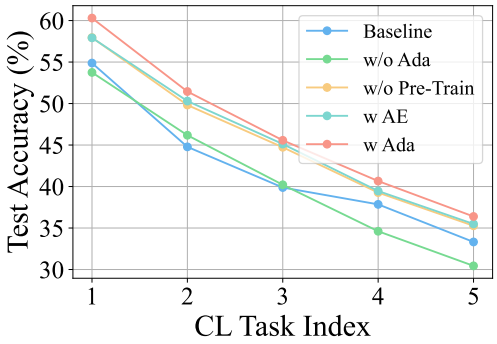


Figure 5. Performance of continual learning on CIFAR100 with 5 steps and 20 IPC for different settings.

learning the current one. To overcome the problem of catastrophic forgetting [14], many works use a small buffer to store some valuable data of past tasks for future use [32, 1]. Dataset distillation can thus benefit this area by synthesizing informative samples [27, 34, 36, 43, 3, 26] to prevent forgetting as much as possible.

In this paper, we evaluate the proposed method on CIFAR100-CL following the same 5-step protocol of [52, 50] with 20 classes for each task and 20 IPC. The results in Fig. 5 suggest a consistent conclusion with the ablation studies shown in Sec. 4.3 and indicate that the few-shot dataset distillation proposed in this paper can further facilitate the practical use when tasks come fast like a stream, which provides limited time for processing and learning.

5. Conclusion

In this paper, we focus on the setting of few-shot dataset distillation to improve its efficiency, where data can only be processed by a limited number of neural networks. We introduce the concept of distillation space to optimize the performance of synthetic data distilled by only one network. Regarding that the complexity of the quad-level definition and the bi-level algorithm for distillation space, we convert the problem to a translative model. The synthetic dataset is first distilled in a random but fixed neural network and then translated to the desired space. For transferability, we train the translator with a pre-training algorithm, so that a pre-trained translator can be adapted for a new dataset within a limited number of steps. Experiments demonstrate that our method achieves $\sim 15\times$ acceleration and comparable performance with the state-of-the-art baselines relying on thousands of networks. Even in some cases, the translator without adaptation yields competitive performance.

One limitation of our method is that in the adaptation stage, errors of translated synthetic datasets have to be further back-propagated through the translator, which introduces additional GPU memory consumption compared with the baseline method, *e.g.*, 3946 v.s. 3456 MB on CIFAR10 with 10 IPC. Nevertheless, as shown in Fig. 4(c), the translator adapted for small sizes can conduct inference for larger sizes, thereby breaking the limitation on GPU memory. Future works may focus on further improving the transferability of the distillation space translator by dedicated architectural designs or objective functions.

Acknowledgement

This work is supported by the Advanced Research and Technology Innovation Centre (ARTIC), the National University of Singapore under Grant (project number: A-0005947-21-00, project reference: ECT-RP2), and the Singapore Ministry of Education Academic Research Fund Tier 1 (WBS: A0009440-01-00).

References

[1] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2180–2187. IEEE, 2021. 9

[2] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. *arXiv preprint arXiv:2203.11932*, 2022. 1, 2, 3

[3] 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 transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 9

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 5

[5] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. *arXiv preprint arXiv:2206.02916*, 2022. 2, 3, 5, 7

[6] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023. 2

[7] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *arXiv preprint arXiv:2305.10924*, 2023. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8

[9] Zixuan Jiang, Jiaqi Gu, Mingjie Liu, and David Z. Pan. Delving into effective gradient matching for dataset condensation. *arXiv preprint arXiv:2208.00311*, 2022. 3

[10] Yongcheng Jing, Yining Mao, Yiding Yang, Yibing Zhan, Mingli Song, Xinchao Wang, and Dacheng Tao. Learning graph neural networks for image style transfer. In *ECCV*, 2022. 2

[11] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Amalgamating knowledge from heterogeneous graph neural networks. In *CVPR*, 2021. 2

[12] Yongcheng Jing, Chongbin Yuan, Li Ju, Yiding Yang, Xinchao Wang, and Dacheng Tao. Deep graph reprogramming. In *CVPR*, 2023. 2

[13] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. *arXiv preprint arXiv:2205.14959*, 2022. 1, 3

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 9

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 7

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 8

[17] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. *arXiv preprint arXiv:2202.02916*, 2022. 3

[18] Shiye Lei and Dacheng Tao. A comprehensive survey to dataset distillation. *arXiv preprint arXiv:2301.05603*, 2023. 1

[19] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[20] Songhua Liu, Jingwen Ye, Runpeng Yu, and Xinchao Wang. Slimmable dataset condensation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1

[21] Songhua Liu, Jingwen Ye, Runpeng Yu, and Xinchao Wang. Slimmable dataset condensation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[22] Yanqing Liu, Jianyang Gu, Kai Wang, Zheng Zhu, Wei Jiang, and Yang You. DREAM: Efficient dataset distillation by representative matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 1

[23] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 4

[24] Noel Loo, Ramin Hasani, Mathias Lechner, and Daniela Rus. Dataset distillation with convexified implicit gradients. *arXiv preprint arXiv:2302.06755*, 2023. 2

[25] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the Structural Pruning of Large Language Models. *arXiv preprint arXiv:2305.11627*, 2023. 2

[26] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 9

[27] Wojciech Masarczyk and Ivona Tautkute. Reducing catastrophic forgetting with learning on synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Workshop*, 2020. 9

[28] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020. 1, 2, 4

[29] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2, 4

[30] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 5

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[32] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 9

[33] Sucheng Ren, Fangyun Wei, Zheng Zhang, and Han Hu. Tinymim: An empirical study of distilling mim pre-trained models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3687–3697, 2023. 2

[34] Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. *arXiv preprint arXiv:2103.15851*, 2021. 9

[35] Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *arXiv preprint arXiv:2301.04272*, 2023. 1

[36] Mattia Sangermano, Antonio Carta, Andrea Cossu, and Davide Bacciu. Sample condensation in online continual learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. 9

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 8

[38] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 5

[39] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 5

[40] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. *arXiv preprint arXiv:2203.01531*, 2022. 1, 3

[41] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018. 1, 2, 3, 7

[42] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020. 5

[43] Felix Wiewel and Bin Yang. Condensed composite memory continual learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. 9

[44] Xingyi Yang, Jingwen Ye, and Xinchao Wang. Factorizing knowledge in neural networks. In *European Conference on Computer Vision*, 2022. 2

[45] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. In *Advances in neural information processing systems*, 2022. 2

[46] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 2

[47] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *arXiv preprint arXiv:2301.07014*, 2023. 1, 2

[48] Lei Zhang, Jie Zhang, Bowen Lei, Subhabrata Mukherjee, Xiang Pan, Bo Zhao, Caiwen Ding, Yao Li, and Xu Dongkuan. Accelerating dataset distillation via model augmentation. *arXiv preprint arXiv:2212.06152*, 2022. 3

[49] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021. 1, 3, 6, 8

[50] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *arXiv preprint arXiv:2110.04181*, 2021. 1, 3, 8, 9

[51] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020. 1, 3, 8

[52] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022. 1, 2, 4, 6, 7, 8, 9