# Distilling Dataset via Matching Smooth and High-Quality Expert Trajectory

Jiyuan Shen
Nanyang Technological University, Singapore
jiyuan001@e.ntu.edu.sg

## Abstract

*Training a large and state-of-the-art machine learning model typically necessitates the use of large-scale datasets, which, in turn, makes the training and parameter-tuning process expensive and time-consuming. Some researchers opt to distil information from real-world datasets into tiny and compact synthetic datasets while maintaining their ability to train a well-performing model, hence proposing a data-efficient method known as Dataset Distillation (DD). Despite recent progress in this field, existing methods still underperform and cannot effectively replace large datasets. In this paper, unlike previous methods that focus solely on improving the efficacy of student distillation, we are the first to recognize the important interplay between expert and student. We argue the significant impact of expert smoothness when employing more potent expert trajectories in subsequent dataset distillation. Based on this, we introduce the integration of clipping loss and gradient penalty to regulate the rate of parameter changes in expert trajectories. Furthermore, in response to the sensitivity exhibited towards randomly initialized variables during distillation, we propose representative initialization for synthetic dataset and balanced inner-loop loss. Finally, we present two enhancement strategies, namely intermediate matching loss and weight perturbation, to mitigate the potential occurrence of cumulative errors. We conduct extensive experiments on datasets of different scales, sizes, and resolutions. The results demonstrate that the proposed method significantly outperforms prior methods.*

## 1. Introduction

Nowadays, large machine learning models have demonstrated unprecedented results in many fields such as computer vision, natural language processing and speech recognition. Behind the usage of promising prospects of large models, large-scale dataset has gradually become a norm. At such scales, storing and preprocessing datasets becomes burdensome, and the need for specialized equipment and substantial financial investment exacerbates the difficulties of training large machine learning models on large-scale datasets. As a result, some researchers try to explore data-efficient methods, including coreset selection [1, 2, 38], dataset pruning [10, 15, 23] and instance selection [35]. They try to summarize the entire dataset by a small subset or by only selecting the 'valuable' data for model training. However, these approaches have resorted to greedy algorithms with heuristics [2, 8, 38, 42], resulting in significant performance drops, and there is no assurance of producing representative samples.

Recently, a new data-efficient method called Dataset Distillation (DD) has emerged as a competitive alternative with promising results [9, 11, 37, 46]. The goal of DD is to distil a large training set into a small synthetic set, upon which machine learning models are trained from scratch, and similar testing performance on the validation set is expected to be preserved. DD has aroused significant interest from the machine learning community with various downstream applications, such as federated learning [26, 36, 53], neural architecture search [30, 40], and privacy-preserving tasks [7, 28, 29], etc.

As one of the state-of-the-art frameworks for dataset distillation, Matching Training Trajectories (MTT) [5] distinguishes itself by long-range matching characteristic. MTT encompasses two primary phases: expert trajectory generation (buffer) and student parameter matching (distillation). Nevertheless, despite recent advancements, the research works [5, 12, 14] continue to employ the simplest optimizer for expert trajectory generation, accompanied by sole weight matching loss in the second phase.

Prior works lack consideration regarding the suitability of experts for students, thus ignoring potential relationship that could lead to consistent improvement of both students and experts. We observe that the smoothness of expert trajectories plays an important role. For instance, when substituting the initial non-momentum Stochastic Gradient Descent (SGD) with a superior optimizer, it does enhance the accuracy of expert model. However, this replacement markedly intensifies the rate of parameter changes in expert model, rendering it less smooth and moderate, ultimately resulting in difficulty of student parameter matching and a

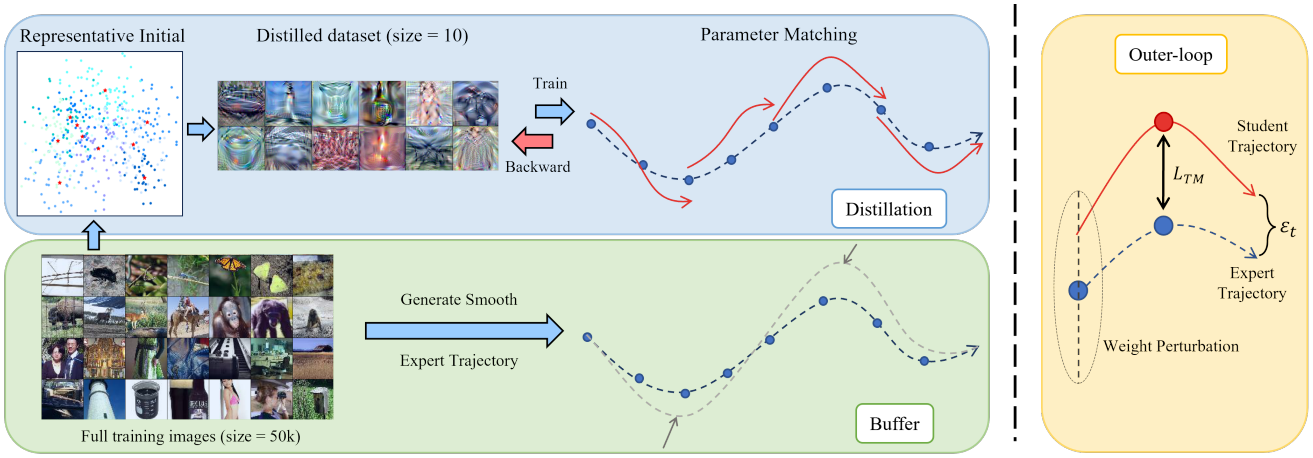



Figure 1. Illustration of improved trajectory matching method: (a) Buffer: We generate the smooth and high-quality expert trajectory (blue dash curve) on the real dataset $\mathcal{D}_{real}$. The grey dash curve is a much steeper expert trajectory without applying clipping loss and gradient penalty. (b) Distillation: We first select representative initialization samples from the original training images. Then, the synthetic dataset $\mathcal{D}_{syn}$ is optimized to match the segments of the expert trajectory through parameter matching. We apply a balanced inner-loop loss $\hat{\ell}_{BIL}$ to mitigate the influence of random initialized expert starting epoch $t$. The red solid curve denotes student trajectory. (c) Outer-loop in distillation: We show how student trajectories are matched with expert trajectories within a single iteration. We introduce weight perturbation $\mathbf{d}_{l,j}$ to the well-trained expert model parameters. Subsequently, we calculate the intermediate matching loss $\mathcal{L}_{TM}$ in the middle of the loop. These two methods can help mitigate the accumulated errors $\varepsilon_t$.

significant decline in the distillation outcomes.

Besides, within the trajectory matching paradigm, the student parameter matching in the second stage exhibits a heightened susceptibility to two randomly initialized variables, namely the initial samples and the initial starting epochs of expert trajectories. The above two factors together result in optimization instability of the distillation process. Particularly when distilling datasets of larger scale and higher resolution, it may lead to the generation of images akin to random noise and 'black holes'.

Finally, we argue that not only discrepancy between distillation and evaluation will lead to cumulative errors [14]. The long interval between inner and outer loop under the bi-level optimization structure will also generate errors in the second stage, which degrade the performance of distilled dataset.

Building upon this, we have instituted improvements in both the expert trajectory generation and student parameter matching phases, as shown in Fig. 1. We propose the amalgamation of clipping loss and gradient penalty with a more proficient optimizer to regulate the rate of parameter update and concurrently sustain high expert performance throughout the training of expert models in the initial phase. Additionally, we implement a strategy involving the use of representative sample initialization and inner loop loss balancing to mitigate the sensitivity to randomly initialized variables during dataset distillation. By leveraging smoother expert trajectories and the prescribed parameter matching strategy, we improve matching stability in the second stage and facilitate the production of superior distilled datasets. Moreover, we propose two enhancement strategies, namely intermediate matching loss and weight perturbation of expert model, to ameliorate the propensity for cumulative errors. Our proposed methods have been extensively evaluated on datasets varying in scale, size, and resolution, and the results substantiate their efficacy, exhibiting significant superiority over prior approaches.

In summation, our main contributions can be summarized as follows:

- We propose the need for integration of clipping loss and gradient penalty under a more potent optimizer to keep the expert trajectory smoothness.
- We propose representative sample initialization and inner loop loss balancing strategies to alleviate the impact of stochasticity during the second-stage distillation.
- We propose two enhancement strategies, namely intermediate matching loss and the perturbation of expert model weights, to mitigate the accumulation of errors.

## 2. Related Works

### 2.1. Dataset Distillation

The existing DD frameworks can be divided into four parts [37], namely Meta-model Learning, Gradient Matching, Distribution Matching and Trajectory Matching.

Wang *et al.* [44] first introduces the dataset distillation problem and uses bi-level optimization similar to Model-Agnostic Meta-Learning (MAML) [16]. Following the Meta-Learning frameworks, recent works include adding soft labels [4, 41], developing closed-form approximation KIP [33, 34] and FRePo [52], and applying prior towards the synthetic dataest [6, 27]. Gradient Matching conducts a single-step distance matching between the network trained on the original dataset and the identical network trained on synthetic data, methods including DC [51], DSA [48], DCC [25] and IDC [21]. To alleviate complex bi-level optimization and second-order derivative computation, Distribution Matching directly matches the distribution of original data and synthetic data with a single-level optimization, methods include DM [50], CAFE [43], IT-GAN [49]. However, the above-mentioned works are all short-range matching methods and may easily cause overfitting problem [46].

Trajectory Matching leverages the well-trained expert training trajectory as reliable references and aims to minimize the distance between expert and student trajectories in the parameter matching process of the second stage, which improves the performance of distilled dataset. Recent works include MTT [5], TESLA [12] that reparameterizes the computation of matching loss and FTD [14] that minimizes accumulated errors between distillation and evaluation. Nevertheless, Trajectory Matching still has its own issues, such as ignoring the interplay between smoothness and accuracy of expert trajectories, susceptibility to random initial variables, and cumulative errors.

### 2.2. Relationship between Teacher and Student

Currently, there exist some methods in the field of Knowledge Distillation (KD) that explore the factors of what makes teachers conducive to students learning or how to transfer more valuable knowledge from highly proficient teachers. Huang *et al.* [20] has empirically pointed out that simply enhancing a teacher's capabilities or employing a stronger teacher can, paradoxically, result in a decrease in student performance. In some cases, this decline can even be more pronounced than if students were to begin training from scratch with vanilla KD. Consequently, they have proposed a loose matching approach to replace the traditional Kullback-Leibler (KL) divergence.

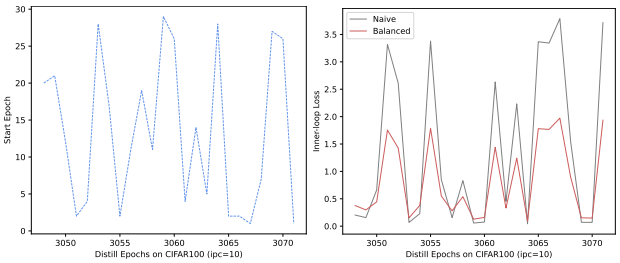
Similarly, Yuan *et al.* [47] argues that simplifying the teacher model's outputs can offer greater advantages to the student's learning process. Meanwhile, Shao *et al.* [39] proposes the principle that 'teachers should teach what you should teach' and introduces a data-based knowledge distillation method. All of the methods mentioned above underscore the intimate relationship between teachers and students, necessitating a thorough exploration of how to enhance the abilities of both teachers and students consistently. Nonetheless, there remains a vacancy in this relationship under the context of Dataset Distillation.

| Optimizer (Exp) | Acc. (Exp) | Acc. (Distill) |
|---|---|---|
| Naïve SGD | 48.6 | 39.7 |
| SGD w/ Mom | 57.1 | 18.8 |
| ADAM | 52.7 | 18.1 |

Table 1. Expert model accuracy and distilled results on various expert trajectories using various optimizers on CIFAR-100 under ipc=10.

| Initial Samples | Starting Epoch | Acc. (Distill) |
|---|---|---|
| Random | Random | 39.7 |
| Representative | Random | 40.3 |
| Random | Balanced Strategy | 40.1 |

Table 2. After using the stochasticity balancing strategy during parameter matching, there has been a consistent improvement in the performance of the distilled dataset.




(a) Randomly Start Epoch (b) Inner-loop Loss

Figure 2. Because of the random variable of starting epoch for each iteration (as depicted in the blue dashed plot in (a)), it leads to substantial fluctuations in the loss scale within the inner loop, as indicated by the solid grey plot in (b). After employing a balanced inner-loop loss, the extent of loss variation is constrained to a lesser degree during the early starting epochs, while allocating more weight to the later starting epochs, as illustrated by the red plot in (b).

## 3. Method

### 3.1. Generate Smooth Expert Trajectories

#### 3.1.1 Catastrophic Discrepancy with Stronger Expert Trajectory

As depicted in Sec. 1, the influence of a teacher on DD has not been adequately investigated, especially as the performance of the expert model becomes stronger, for instance, through the use of better optimizers. With this regard, as Tab. 1, we demonstrate both results of the expert model and the distilled dataset achieved with various optimizers.

We have observed that when employing SGD with momentum or ADAM as optimizers, although they generate stronger expert trajectories, theoretically elevating the upper limit of distillation dataset performance, this increment does not directly connect with improved results of dis-

tilled dataset. On the contrary, the accuracy of the distilled dataset on the validation set experiences a substantial decline, even falling behind the vanilla MTT method. This indicates that the knowledge encapsulated within the more potent expert trajectories is not effectively assimilated during following distillation process. The underlying reason is that the utilization of more potent optimizers often incorporates the previously accumulated gradient means into the optimization process, which facilitates faster model convergence [18]. However, this causes the expert model's parameters to update too rapidly, surpassing the limitation of what can be distilled during the second stage. It is detrimental to the parameter matching process. For specific experimental details, please refer to Sec. 4.4.1.

Consequently, our proposed approach aims to enhance the performance of the expert model while simultaneously moderating the rate of parameter updates in the training process of expert models. Our strategy allows us to obtain a series of smooth and high-quality expert trajectories.

#### 3.1.2 Better Expert Trajectory under Smoothness Constraint

To constrain the changing speed of expert parameters, the straightforward method involves the utilization of value clipping on the cross-entropy loss, with a specific focus on the initial epochs. Gradually, the clipping coefficient is incremented until it reaches a value of 1, which is then held constant. Our findings indicate that although loss clipping may impact the final expert performance to some extent, its key benefit lies in mitigating the rapid changes in model parameters, particularly during the initial few epochs.

Besides, to maintain the overall smoothness of the expert trajectory, we introduce the concept of gradient penalty, building upon the insights from the Wasserstein GAN (WGAN) [3, 19]. The goal of employing gradient penalty is to force the model to satisfy Lipschitz continuity, preventing abrupt and erratic changes in the model's parameters. We incorporate gradient penalty by adding a regularization term to the loss function. This term is formulated as the squared norm of the gradient $\nabla_x \mathcal{W}(x)$ of the model's output with respect to its input. By penalizing large gradients, we incentivize the model to produce outputs that change gradually as inputs vary slightly. This has the effect of maintaining a smoother transition between data points and preventing sudden shifts that could lead to instability. We formulate the final loss equation as below:

$$\mathcal{L}_{SC} = \underbrace{\lambda \log \frac{\exp\left(x_{n,y_n}\right)}{\sum_{c=1}^{C} \exp\left(x_{n,c}\right)}}_{\text{Clipped CELoss}} + \underbrace{\mu \mathop{\mathbb{E}}_{x \sim \mathbb{P}_x} \left[ \left( \|\nabla_x \mathcal{W}(x)\|_2 - \mathcal{K} \right)^2 \right]}_{\text{Gradient Penalty}} \tag{1}$$

where $\|\nabla_x \mathcal{W}(x)\|_2$ represents a dual-sided penalty that aims to constrain the gradient norm to values below a predetermined threshold denoted as $\mathcal{K}$. Typically, $\mathcal{K}$ is set at 1. The coefficient $\lambda$ operates within a range of 0.5 to 1 during the initial few epochs, while the coefficient $\mu$ is responsible for scaling the gradient penalty effect.

### 3.2. Balance Stochasticity during Parameter Matching

#### 3.2.1 Representative Initialization for Synthetic Dataset

Many existing dataset distillation algorithms initialize $\mathcal{D}_{syn}$ either by employing random initialization with Gaussian noise or by randomly selecting a subset of images from the real dataset. However, this approach can inadvertently introduce outliers, lead to the inclusion of samples with similar features, or overlook certain aspects of the feature space, which may introduce huge biases and stochasticity. Moreover, as distilled datasets often end up with a limited number of samples per class, it becomes crucial to efficiently encapsulate the inherent information of the original dataset within these limited samples. As a solution, we propose an alternative approach that facilitates more representative initialization.

Leveraging the benefits of the MTT framework can simplify the process of selecting representative initialization samples for $\mathcal{D}_{syn}$. In the first stage, we have already obtained the training trajectory of the expert model. By loading the parameters of expert trajectory into the model, we can acquire a well-trained model. Next, we input all real data of the same class into the model, obtaining feature vectors before entering the fully connected layer. Subsequently, we perform clustering on these feature maps, utilizing the K-Means algorithm to partition them into multiple sub-clusters. The value of K is chosen based on the desired number of distilled samples. These sub-cluster centroids are then selected as exemplary initialization samples.

The primary objective of this approach is to enhance the initialization process by strategically selecting representative samples that are better suited for distillation.

#### 3.2.2 Balanced Inner-loop Loss

During the second stage, student parameter matching exhibits sensitivity to the randomly initialized expert starting epochs $t$ in each iteration. This sensitivity is specifically manifested in significant fluctuations of the inner-loop loss $\ell_{IL}$. As shown in Fig. 2b, the largest loss is nearly 60 times larger than the smallest one. The fluctuations in the inner-loop loss introduce a notable level of instability into the distillation procedure, which impedes the model convergence and hinders the consistent acquisition of accurate distilled data.

Intuitively, our initial thought is to discard the practice of entirely random selection for the starting epoch and instead

opt for a method that references the preceding starting point and selects the subsequent starting point within a reasonable range. However, experiments (details in Sec. 4.4.2) have demonstrated that this dynamic selection approach fails to enhance the final performance. We speculate that employing a range selection may introduce potential bias into the distillation process, which in turn causes the model to learn incorrect inductive bias. Hence, we opt to directly balance the loss within the inner loop.

When the starting epoch $t$ is much smaller, which means the network is under convergence, it will typically generate a much larger loss $\ell_{IL}$ leading the gradient descent towards this direction. Thus, we add a penalty coefficient $\nu$ to narrow the loss, clipping the loss to a small extent. Conversely, when the starting epoch $t$ is large (close to $\mathcal{T}_+$), the optimized steps become much smaller due to the decreased loss. We also add a coefficient $\nu$ to encourage a much bigger gradient. Under this circumstance, the balanced inner-loop loss $\hat{\ell}_{BIL}$ can be formulated as:

$$\nu = \begin{cases} log(|start - middle| + \vartheta), & start \geq middle \\ 1/log(|middle - start| + \vartheta), & start < middle \end{cases}$$
$$\hat{\ell}_{BIL} = \nu \times \ell_{IL} \tag{2}$$

where $start$ represents the starting epoch, $middle$ refers to half of the maximum starting epoch and $\vartheta$ is included to balance the log scale and prevent negative values. As shown in Fig. 2, using balanced loss $\hat{\ell}_{BIL}$ can reduce the loss fluctuation in the inner loop. For $\ell_{IL}$, it is similar to cross-entropy loss:

$$\ell_{IL} = \frac{1}{|\mathcal{D}_{syn}|} \sum_{(s,y)\in\mathcal{D}_{syn}} \Big[ y \log(\hat{\theta}_t(\mathcal{A}(s))) \Big] \tag{3}$$

where $\mathcal{A}$ represents the differentiable siamese augmentation [48], while $\hat{\theta}_t$ denotes the student network parameterized using the expert's $t$-th epoch.network, $t$ is smaller than the defined maximum start epoch $\mathcal{T}_+$ and $\mathcal{D}_{syn}$ is the distilled samples.

## 3.3. Alleviate the Propensity for Accumulated Error

### 3.3.1 Intermediate Matching Loss

Trajectory matching is a long-range framework, typically involving a larger number of steps denoted as $\mathcal{N}$ to match a smaller number of epochs denoted as $\mathcal{M}$. Here, $\mathcal{M}$ signifies the interval between starting and target point of the expert trajectory, while $\mathcal{N}$ represents the training steps of the student model on the distilled dataset. $\mathcal{N}$ is also equivalent to the number of iterations in the inner loop. After every $\mathcal{N}$ steps, it will execute a matching interaction with the expert trajectory.

However, extensive intervals of interaction can lead to the inner loop deviating from the correct direction prematurely. It may result in the accumulation of errors without sufficient supervision guidance. Therefore, we propose the intermediate matching loss, that can optionally perform an additional parameter matching step within the inner loop.

Specifically, we establish a set of intermediate matching points denoted as $\xi$, which consists of two parameters, $\mathcal{M}$ and $\mathcal{N}$. Here, $\mathcal{M}$ signifies the interval between starting and target point of the expert trajectory, while $\mathcal{N}$ represents the training steps of the student model on the distilled dataset. Thus, it can be denoted as $\{\xi\} = \{\lfloor \mathcal{N}/\mathcal{M} \rfloor, \lfloor 2\times\mathcal{N}/\mathcal{M} \rfloor, ..., \lfloor (\mathcal{M}-1)\times\mathcal{N}/\mathcal{M} \rfloor\}$. When the inner loop $n$ $(from\ 1\ to\ \mathcal{N})$ precisely aligns with $\xi_i \in \{\xi\}$, the intermediate matching loss function is computed as:

$$\mathcal{L}_{TM} = \frac{\left\| \hat{\theta}_{t+n} - \theta^*_{t+\xi_i} \right\|_2^2}{\left\| \theta^*_t - \theta^*_{t+\xi_i} \right\|_2^2} \tag{4}$$

in which $\theta^*_t$ is the expert training trajectory at randomly starting epoch $t$, also known as the initial expert weights. Starting from $\theta^*_t$, $\theta^*_{t+\xi_i}$ denotes $\{\xi\}$ steps trained by the expert network on real data and $\hat{\theta}_{t+n}$ stands for $n$ steps $(n < \mathcal{N})$ by student network trained on synthetic data correspondingly. The goal is to minimize the divergence between $\hat{\theta}_{t+n}$ and $\theta^*_{t+\xi_i}$, and $\left\| \theta^*_t - \theta^*_{t+\xi_i} \right\|_2^2$ is used to self-calibrate the magnitude across different iterations. $\mathcal{L}_{TM}$ is then utilized to update the student parameters.

### 3.3.2 Weight Perturbation on Initial Expert Model

Another potential source of cumulative error stems from the disparity between the distillation and evaluation processes, as pointed out in [14]. Due to the discrete nature of the distillation process versus the continuous nature of the evaluation process, cumulative errors can thus arise. [14] introduce a regularization term during expert trajectory generation, however, we adopt a more direct and simple approach, named weight perturbation.

Weight perturbation is an orthogonal method to explore data diversity to boost the performance of the network, different from the data operation like transformation [48] or perturbation [31, 32]. We adopt a weight-perturbed initial expert model to formulate our dataset distillation, expressed as follows:

$$\mathbf{d}_{l,j} = \frac{\mathbf{d}_{l,j}}{\|\mathbf{d}_{l,j}\|_F} \|\theta_{l,j}\|_F$$
$$\theta^*_t = \theta_t + \alpha * \mathbf{d}_{l,j} \tag{5}$$

where $\mathbf{d}_{l,j}$ is sampled from a Gaussian distribution $\mathcal{N}(0,\ 1)$ with dimensions same as $\theta_{l,j}$. $\mathbf{d}_{l,j}$ is the $j$-th filter at the $l$-th layer of $\mathbf{d}$ and $\|\cdot\|_F$ refers to the Frobenius norm. Finally, a coefficient $\alpha$ is added to obtain the final $\theta^*_t$.

Although it may increase the inner-loop loss, the perturbed weight initialization during the distilled process simulates the slight noise of weight parameters generated in the

evaluation process. As the weight of the next epoch is obtained from the previous training epoch instead of getting from the totally correct expert network, weight perturbation on initial expert model during distillation process can effectively reduce the accumulated error.

### 3.4. Training Algorithm

We depict our proposed method in Algorithm 1.

## 4. Experiments

### 4.1. Experiments Setup

The majority of our experimental procedures closely follow the previous works [5, 12, 14], which ensure a fair and equitable basis for comparison. Each of our experiments comprises three essential phases: buffer (generating expert trajectories), distillation (parameter matching), and evaluation phase. First, we generate 50 distinct expert training trajectories, with each trajectory encompassing 50 training epochs. Second, we synthesize a small synthetic set (e.g., 10 images per class) from a given large real training set. Finally, we employ this learned synthetic dataset to train randomly initialized neural networks and assess the performance of these trained networks on the real test dataset.

**Datasets.** We verify the effectiveness of our method on both low- and high- resolution datasets distillation benchmarks, including CIFAR-10 & CIFAR-100 [22], Tiny ImageNet [24] and ImageNet [13]. Top-1 accuracy is reported to show the performance.

**Baselines.** We compare our method with various baselines including Dataset Condensation (DC) [51], Differentiable Siamese Augmentation (DSA) [48], Distribution Matching (DM) [50], Aligning Features (CAFE) [43], Feature Regression with Pooling (FRePo) [52], trajectory matching method (MTT) [5], memory-efficient trajectory matching method (TESLA) [12], and Flat Trajectory Distillation (FTD) [14].

**Implementation Details.** In the buffer phase, we utilize SGD with momentum as the optimizer, setting $\lambda$ as an array ranging from 0.5 to 1. This range is applied individually for the first 5 training epochs, while $\mu$ is set to 1. We train each expert model for 50 epochs, with the learning rate reduced by half in the 25th epoch. During the distillation process, the value of K in the K-Means algorithm is determined based on the ipc (images per class) value. However, when ipc equals 50, we opt to cluster only 10 sub-clusters, each selecting extra 4 points approximating the sub-cluster centroid. For the balanced loss, we set $\theta$ to 8. Concerning the intermediate matching loss, we introduce a hyperparameter $\beta$ to control the scale of several losses, encompassing two strategies: equal scale $\beta$=1 or varied scale $\beta$ based on the value of $\{\xi\}$. In terms of weight perturbation, we set $\alpha$ to 0.1, and we also conduct experiments to evaluate the performance with dropout added. Throughout the evaluation phase, the number of training iterations is set to 1000, with a learning rate reduction by half at the 500-th iteration. Furthermore, we employ the same Differentiable Siamese Augmentation (DSA) during both the distillation and evaluation processes.

### 4.2. Comparison with State-of-the-Art Methods

**Competitors**: We categorize the current works into three main parts, shown in Tab. 3. The first part focuses on coreset selection and includes non-learning methods such as random selection, herding methods [8], and example forgetting [42]. These techniques aim to select 'valuable' instances from a large dataset. The second part comprises methods like DC, DM, DSA, CAFE, and FrePo, which primarily address short-range matching and truncated gradient backpropagation. They employ gradient matching or distribution matching as the optimization objective. The third part including MTT, TESLA and FTD applies long-range matching characteristics, based on the framework of MTT, which are most relevant to our method.

**Results with Coreset & Short-range**: Our proposed method outperforms the coreset selection baselines significantly. The non-learning methods achieve inferior performance compared to our methods. When comparing with the second part works, our method demonstrates the same superiority over all other methods. Moreover, we improve one of the strong baseline DSA to nearly 15% accuracy on both CIFAR-10 and CIFAR-100 under all ipc settings.

**Results with Long-range**: Regarding the comparison of experimental results in the third part, we observe consistently positive outcomes with significant improvements in all settings. We specifically compare our method with the original MTT. For example, in CIFAR-10, at a compression rate of fifty images, we achieve a score of 74.6%, which is 3% higher than the accuracy score of MTT. Compared to the results obtained from the full dataset, we are only around 10% behind. Similar improvements are observed in CIFAR-100, where we achieve a 4% increase over MTT and our results are only 4.5% lower than the results obtained from the full dataset. We visualize parts of the synthetic images in Fig. 3 and discover that our distilled samples Fig. 3c can focus more on the classified object itself, while diluting the background information. Compared to the original images, the distilled dataset is more informative and abstract.

In the more intricate Tiny ImageNet dataset, our approach demonstrates consistent and significant improvements, achieving 13.7% and 25.7% in ipc values of one and ten, respectively. These empirical findings substantiate the efficacy of our proposed method.

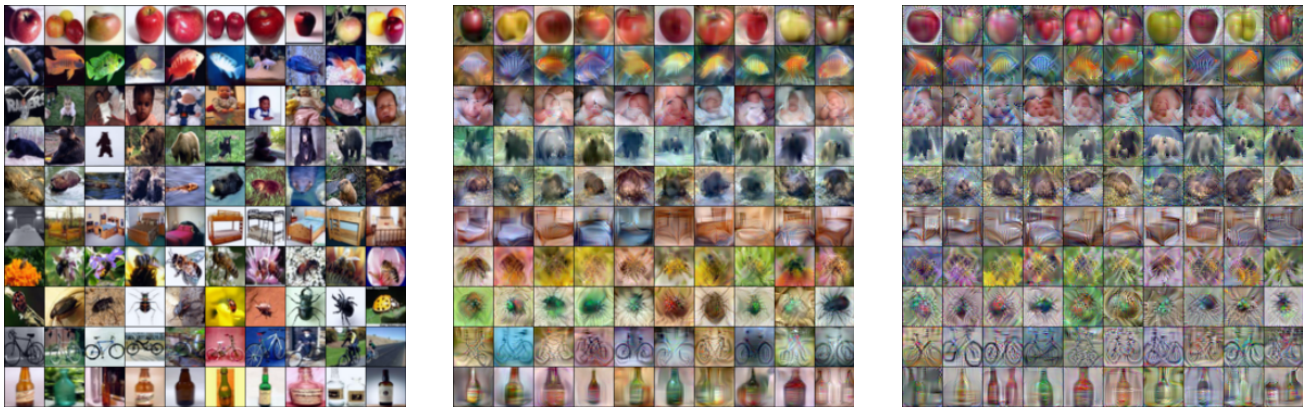
**Cross-Architecture Generalization**: We evaluate generalization capacity across various architectures. Initially, we distilled the synthetic dataset using ConvNet. Subse-

**Algorithm 1:** Dataset Distillation in Parameter Matching

**Input:** $\{\tau_i\}$: set of smoothed expert parameter trajectories trained on real dataset $\mathcal{D}_{real}$.
**Input:** $\mathcal{M}$: interval between starting and target expert trajectory.
**Input:** $\mathcal{N}$: distillation steps of student network.
**Input:** $\mathcal{A}$: differentiable siamese augmentation.
**Input:** $\mathcal{T}_+$: maximum start epoch.
**Input:** $\{\xi\} = \{\lfloor \mathcal{N}/\mathcal{M} \rfloor, ..., \lfloor (\mathcal{M}-1)\times\mathcal{N}/\mathcal{M} \rfloor\}$: set of intermediate matching epoch.

Select representative initialization samples $\mathcal{D}_{syn} \sim \mathcal{D}_{real}$ ⇒ Method 3.2.1;
Initialize trainable learning rate $\alpha := \alpha_0$;
**for** ***each*** *distillation step* **do**
  Sample smooth expert trajectory $\tau \sim \{\tau_i\}$ with $\tau := \{\theta_t\}$. ⇒ Method 3.1.2;
  Choose random start epoch, $t \leqslant \mathcal{T}_+$;
  Perturb weight on initial expert model with $\tau^* := \{\theta_t^*\}$ ⇒ Method 3.3.2;
  Initialize student network with expert parameters $\hat{\theta}_t := \theta_t^*$;
  **for** *n = 1 to* $\mathcal{N}$ **do**
    Sample a mini-batch of distilled images: $b_{t+n} \sim \mathcal{D}_{syn}$;
    Compute the cross-entropy loss based on Eq. (3). ;
    Get $\nu$ based on Eq. (2) and balance the inner-loop loss: $\hat{\ell}_{BIL} = \nu \times \ell_{IL}$ ⇒ Method 3.2.2;
    Update student model: $\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla \hat{\ell}_{BIL}$;
    **if** *n* ***in*** $\{\xi\}$ **then**
      Calculate intermediate matching loss $\mathcal{L}_i = \left\| \hat{\theta}_{t+n} - \theta^*_{t+\xi_i} \right\|_2^2 / \left\| \theta^*_t - \theta^*_{t+\xi_i} \right\|_2^2$ ⇒ Method 3.3.1 ;
    **end**
  **end**
  Get the final loss $\hat{\mathcal{L}} = \sum_{\xi_i} \beta_i \times \mathcal{L}_i$ ;
  Update $\mathcal{D}_{syn}$ and $\alpha$ with respect to $\hat{\mathcal{L}}$;
**end**

**Output:** Distilled dataset $\mathcal{D}_{syn}$ and learning rate $\alpha$



(a) Original CIFAR100 images. (b) The synthetic images of MTT. (c) The synthetic images of ous.

Figure 3. Visualizations of original images, and synthetic images generated by MTT and our proposed methods.

quently, we trained several architectures, namely AlexNet, VGG11, and ResNet18, on the distilled dataset. Tab. 4 presents the results of our evaluations. Our method outperforms MTT significantly in generalization performance.

### 4.3. Results on ImageNet Subsets (128×128)

To further evaluate our method, we present results on a larger and more challenging dataset in Tab. 5. The ImageNet subsets pose a greater difficulty compared to CIFAR-

| IPC | CIFAR-10 | | | CIFAR-100 | | | Tiny ImageNet | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 50 | 1 | 10 | 50 | 1 | 10 |
| Full Dataset | | 84.8±0.1 | | | 56.2±0.3 | | 39.5±0.4 | |
| Random | 14.4±2.0 | 26.0±1.2 | 43.4±1.0 | 4.2±0.3 | 14.6±0.5 | 30.0±0.4 | 1.4±0.1 | 5.0±0.2 |
| Herding [8] | 21.5±1.2 | 31.6±0.7 | 40.4±0.6 | 8.4±0.3 | 17.3±0.3 | 33.7±0.5 | 2.8±0.2 | 6.3±0.2 |
| Forgetting [42] | 13.5±1.2 | 23.3±1.0 | 23.3±1.1 | 4.5±0.2 | 15.1±0.3 | 30.5±0.3 | 1.6±0.1 | 5.1±0.2 |
| DC [51] | 28.3±0.5 | 44.9±0.5 | 53.9±0.5 | 12.8±0.3 | 25.2±0.3 | - | - | - |
| DM [50] | 26.0±0.8 | 48.9±0.6 | 63.0±0.4 | 11.4±0.3 | 29.7±0.3 | 43.6±0.4 | 3.9±0.2 | 12.9±0.4 |
| DSA [48] | 28.8±0.7 | 52.1±0.5 | 60.6±0.5 | 13.9±0.3 | 32.3±0.3 | 42.8±0.4 | - | - |
| CAFE [43] | 30.3±1.1 | 46.3±0.6 | 55.5±0.6 | 12.9±0.3 | 27.8±0.3 | 37.9±0.3 | - | - |
| FRePo [52] | 45.1±0.5 | 59.1±0.3 | 69.6±0.4 | 25.9±0.1† | 40.9±0.1 | - | 13.5±0.1† | 20.4±0.1 |
| MTT [5] | 46.2±0.8 | 65.4±0.7 | 71.6±0.2 | 24.3±0.3 | 39.7±0.4 | 47.7±0.2 | 8.8±0.3 | 23.2±0.2 |
| TESLA [12] | 48.5±0.8† | 66.4±0.8 | 72.6±0.7 | 24.8±0.4 | 41.7±0.3 | 47.9±0.3 | 9.8±0.4 | 24.4±0.6 |
| FTD [14] | 46.8±0.3 | 66.6±0.3† | 73.8±0.2† | 25.2±0.2 | 43.4±0.3† | 50.7±0.3† | 10.4±0.3 | 24.5±0.2† |
| **Ours** | **48.8±0.9** | **67.1±0.4** | **74.6±0.5** | **26.6±0.4** | **44.4±0.6** | **51.7±0.7** | **13.7±1.4** | **25.7±1.1** |

Table 3. Performance comparision trained with 128 width-ConvNet [17] to other state-of-the-art methods on the CIFAR and Tiny ImageNet. We cite the reported results from Sachdeva *et al.* [37] and Du *et al.* [14]. IPC: Images Per class. Bold digits represent the best results and † refers to the second-best results among all the methods..

| | | Evaluation Model | | | |
|---|---|---|---|---|---|
| | | ConvNet | ResNet | VGG | AlexNet |
| Method | DC | 53.9±0.5 | 20.8±1.0 | 38.8±1.1 | 28.7±0.7 |
| | CAFE | 55.5±0.4 | 25.3±0.9 | 40.5±0.8 | 34.0±0.6 |
| | MTT | 71.6±0.2 | 61.9±0.7 | 55.4±0.8 | 48.2±1.0 |
| | FTD | 73.8±0.2 | 65.7±0.3 | 58.4±1.6 | 53.8±0.9 |
| | **Ours** | **74.6±0.5** | **67.3±0.4** | **60.3±0.5** | **56.7±0.3** |

Table 4. Generalization testing of different architectures on CIFAR-10 dataset with IPC 50.

| IPC | ImageNette | | ImageWoof | | ImageFruit | | ImageMeow | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 |
| Full dataset | 87.4±1.0 | | 67.0±1.3 | | 63.9±2.0 | | 66.7±1.1 | |
| MTT [5] | 47.7±0.9 | 63.0±1.3 | 28.6±0.8 | 35.8±1.8 | 26.6±0.8 | 40.3±1.3 | 30.7±1.6 | 40.4±2.2 |
| FTD [14] | 52.2±1.0 | 67.7±0.7 | 30.1±1.0 | 38.8±1.4 | 29.1±0.9 | 44.9±1.5 | 33.8±1.5 | 43.3±0.6 |
| **Ours** | **53.1±0.8** | **68.0±1.2** | **30.9±0.6** | **39.1±1.5** | **30.0±1.2** | **45.1±1.5** | **34.2±1.5** | **43.9±1.7** |

Table 5. Applying our methods to 128×128 resolution ImageNet subsets. Bold digits represent the best results.

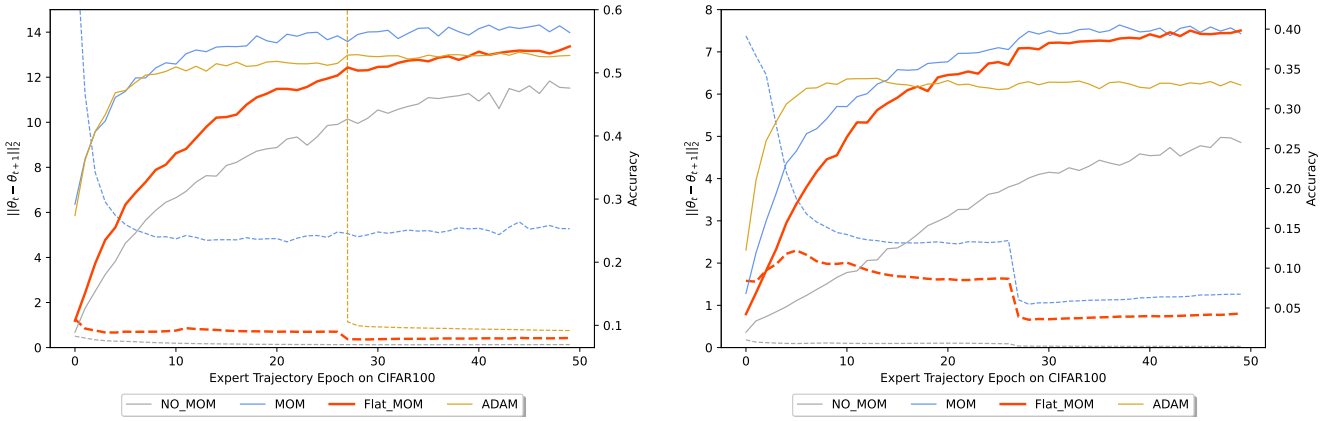
10/100 and Tiny ImageNet, primarily due to their higher resolutions. The higher resolution makes it challenging for the distillation procedure to converge. The ImageNet subsets consist of 10 categories selected from ImageNet-1k, following the setting of MTT. These subsets include ImageNette (assorted objects), ImageWoof (dog breeds), ImageFruits (fruits), and ImageMeow (cats). As shown in Tab. 5, our method significantly improves MTT in every subset. For instance, we achieve a significant performance boost on the ImageNette subset with ipc = 1 and 10, surpassing MTT by more than 5.0%. We also record the FTD results for fair comparison and our method achieves better results.

### 4.4. Analysis

#### 4.4.1 The Impact of Expert Trajectory Smoothness

Fig. 4 and Tab. 6 elucidate why previous works have invariably opted for naive SGD as the optimizer. This choice of SGD strikes an unavoidable trade-off between the performance of expert model and outcome of distilled dataset. For instance, as shown in the Tab. 6a, the expert model alone achieves a modest score of 48.6%, whereas distillation yields a respectable 39.7%. However, despite adopting SGD with momentum or Adam enhancing expert performance (shown in blue and yellow solid lines), it leads to the variation of parameters changing so fast (shown in blue and yellow dash lines) that surpasses the limitation for distillation. Finally, it will cause significant declines in distillation results, even gradient explosions and training collapses, especially for Adam.

The essence of our proposed method for generating smooth expert trajectories lies in constraining the speed of parameter variation. The ideal expert trajectory exhibits slow parameter variations with consistent performance improvements along the iterations. To quantify smoothness, we employ a metric called $Avg\_Var$ to measure the average weight variation magnitude between two iterations along the whole training process:




(a) CIFAR-100 (b) Tiny ImageNet

Figure 4. $\|\theta_t - \theta_{t+1}\|_2^2$ refers to the change of the model weights on two consecutive iterations, shown by dash curve. Correspondingly, the solid curve refers to the metric of evaluation accuracy. NO_MOM refers to SGD without momentum. MOM means using SGD with momentum alone.Flat_MOM denotes applying gradient penalty and clipped loss under the usage of SGD with momentum. ADAM means using ADAM as optimizer alone. We view the red curve as a much smoother and higher-quality expert trajectory. The $\|\theta_t - \theta_{t+1}\|_2^2$ of Adam is so huge that it cannot fully appear in both CIFAR-100 and Tiny ImageNet.

| Momentum | Gradient Penalty | Clipped Loss | Avg_Var ↓ | Acc. (Expert) ↑ | Acc. (Distill) ↑ |
|---|---|---|---|---|---|
| × | × | × | 0.1726 | 48.6 | 39.7 |
| ✓ | × | × | 7.9611 (×46) | 57.1 (+8.5) | 18.8 (−20.9) |
| ✓ | ✓ | × | 0.9331 (×5.4) | 54.1 (+5.5) | 41.7 (+2.0) |
| ✓ | ✓ | ✓ | **0.5899** (×3.4) | **54.4** (+5.8) | **42.0** (+2.3) |

(a) CIFAR-100

| Momentum | Gradient Penalty | Clipped Loss | Avg_Var ↓ | Acc. (Expert) ↑ | Acc. (Distill) ↑ |
|---|---|---|---|---|---|
| × | × | × | 0.0705 | 25.8 | 8.8 |
| ✓ | × | × | 2.2950 (×33) | 39.4 (+13.6) | 1.8 (−7.0) |
| ✓ | ✓ | × | 1.7358 (×24) | 39.0 (+13.2) | 10.0 (+1.2) |
| ✓ | ✓ | ✓ | **1.3066** (×18) | **39.5** (+13.7) | **10.8** (+2.0) |

(b) Tiny ImageNet

Table 6. Comparison between different buffer generation methods using SGD as base optimizer.

$$Avg\_Var = \mathbb{E}_t \left[ \|\theta_t - \theta_{t+1}\|_2^2 \right]$$

Fig. 4 shows $\|\theta_t - \theta_{t+1}\|_2^2$ of each epoch and Tab. 6 demonstrates the average result. Through employing gradient penalty and loss clipping, we achieve significant improvements in both expert performance (an increase of 5.8% and 13.7%) and distilled dataset performance (an increase of 2.3% and 2.0%) while only increasing $Avg\_Var$ by a factor of 3.4 and 18 (compared to 46× and 33× for direct momentum addition, which is just a small increment).

#### 4.4.2 The Impact of Balance Strategy

As demonstrated in Tab. 7, we conduct ablation experiments on each proposed module based on the usage of smooth expert trajectory. In the 'Balance Stochasticity' part, the act of selecting representative samples for initialization resulted in an incremental gain of 0.5%. In order to visualize the effectiveness of the selection strategy, We randomly choose five classes and apply PCA [45] to reduce the high-dimensional features to two dimensions. As can be seen from Fig. 6, compared to random initialization, our proposed method avoids introducing huge bias coming from outliers. Additionally, the distribution of distilled samples is more uniform, and there is no over-concentration in a specific area.

Moreover, the application of a balanced inner-loop loss leads to a further enhancement, yielding an improvement of 0.4%. As mentioned in Sec. 3.2.2, we compared another method: random initialization of $t$ within a certain range. We set the range to be within 5 when selecting the next expert starting point. From Fig. 5, the experiments indicate that initialization within a range not only lacks a positive effect but, conversely, introduces a potential erroneous inductive bias that results in a decline in distillation outcomes.

#### 4.4.3 The Impact of Accumulated Error

To explore the contributions of the two error reduction methods, we conduct ablation experiments as depicted in Tab. 7. In the 'Alleviate Errors' part, the experiments indicate that using weight perturbation leads to the most improvement (+0.8%) in distilled dataset performance. However, when applying $Dropout$ after weight perturbation shown in Fig. 5, which introduces another type of drop noise, we have not observed any improvement in the results.

Besides, intermediate matching loss contributes to an in-

| Methods | Acc. Distill (Gain) |
|---|---|
| Base | 42.0 |
| Balance Stochasticity: | |
| + Representative Initialization | 42.5 (+0.5) |
| + Balanced Inner-loop Loss | 42.9 (+0.4) |
| Alleviate Errors: | |
| + Intermediate Matching Loss | 43.6 (+0.7) |
| + Weight Perturbation | **44.4 (+0.8)** |

Table 7. We use the distilled results obtained by applying smooth expert trajectory as the 'Base'. Following that, we conduct two parts ablation studies (stochasticity and errors) on the CIFAR-100 dataset under IPC=10.

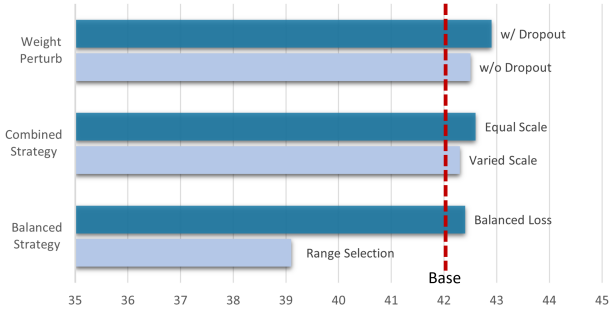



Figure 5. Results of ablation study on weight perturbation, intermediate matching loss and balanced strategy.

crease of 0.7% to the final outcome, which elucidates that extended-range interactions indeed may lead to a divergence in the optimization direction within the inner loop. Reducing this part of the accumulated error is equally significant. We also experiment with two different strategies for combining multiple intermediate matching losses. As illustrated in Fig. 5, we observe that a straightforward approach, where the same scale of $\beta$ coefficient is used, yielded superior results.

## 5. Conclusion

In this paper, we propose a novel dataset distillation strategy to address the problems of interplay between expert and student, sensitivity to stochastic variables, and accumulated errors. Building upon this, we introduce clipping loss and gradient penalty to smooth the expert trajectories under a more potent optimizer. To balance the impact of two random variables, we propose using representative initialization for $\mathcal{D}_{syn}$ and balanced inner-loop loss. Besides, we propose intermediate matching loss and weight perturbation to mitigate the errors arising from long-range inner steps and the discrepancies between distillation and evaluation. Furthermore, our methods are designed to be easily implemented and plugged in. Extensive experiments have confirmed the effectiveness of our approach. We hope our method can pave the way for future works on dataset distillation

## References


[1] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004. 1

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 4

[4] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020. 3

[5] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022. 1, 3, 6, 8

[6] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3739–3748, 2023. 3

[7] Dingfan Chen, Raouf Kerkouche, and Mario Fritz. Private set generation with discriminative information. *Advances in Neural Information Processing Systems*, 35:14678–14690, 2022. 1

[8] Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012. 1, 6, 8

[9] Zongxiong Chen, Jiahui Geng, Herbert Woisetschlaeger, Sonja Schimmler, Ruben Mayer, and Chunming Rong. A comprehensive study on dataset distillation: Performance, privacy, robustness and fairness. *arXiv preprint arXiv:2305.03355*, 2023. 1

[10] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 1

[11] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. *Advances in Neural Information Processing Systems*, 35:810–822, 2022. 1

[12] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory. *arXiv preprint arXiv:2211.10586*, 2022. 1, 3, 6, 8

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 6

[14] Jiawei Du, Yidi Jiang, Vincent TF Tan, Joey Tianyi Zhou, and Haizhou Li. Minimizing the accumulated trajec-

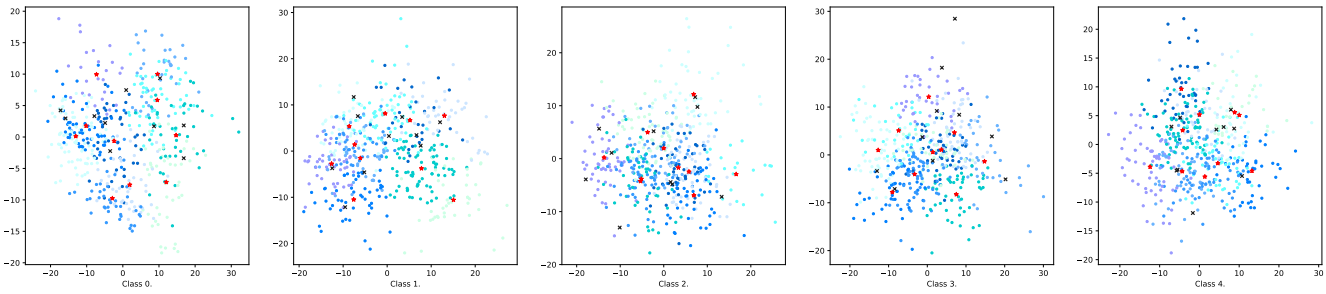



Figure 6. Example clustering and sub-cluster centre results. ⋆ denotes the representative initialization samples while × means the random initialization samples.

tory error to improve dataset distillation. *arXiv preprint arXiv:2211.11004*, 2022. 1, 2, 3, 5, 6, 8

[15] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 1

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 3

[17] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018. 8

[18] Gabriel Goh. Why momentum really works. *Distill*, 2017. 4

[19] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, 30, 2017. 4

[20] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. *Advances in Neural Information Processing Systems*, 35:33716–33727, 2022. 3

[21] 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. In *International Conference on Machine Learning*, pages 11102–11118. PMLR, 2022. 3

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[23] Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013. 1

[24] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 6

[25] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*, pages 12352–12364. PMLR, 2022. 3

[26] Ping Liu, Xin Yu, and Joey Tianyi Zhou. Meta knowledge condensation for federated learning. In *ICLR*, 2023. 1

[27] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. *Advances in Neural Information Processing Systems*, 2022. 3

[28] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Backdoor attacks against dataset distillation. In *Proceedings of the Network and Distributed System Security Symposium*, 2023. 1

[29] Noel Loo, Ramin Hasani, Mathias Lechner, and Daniela Rus. Dataset distillation fixes dataset reconstruction attacks. *arXiv preprint arXiv:2302.01428*, 2023. 1

[30] Dmitry Medvedev and Alexander D'yakonov. Learning to generate synthetic training data using gradient matching and implicit differentiation. In *Proceedings of the International Conference on Analysis of Images, Social Networks and Texts*, pages 138–150, 2021. 1

[31] Giung Nam, Hyungi Lee, Byeongho Heo, and Juho Lee. Improving ensemble distillation with weight averaging and diversifying perturbation. *arXiv preprint arXiv:2206.15047*, 2022. 5

[32] Giung Nam, Jongmin Yoon, Yoonho Lee, and Juho Lee. Diversity matters when learning from ensembles. *Advances in Neural Information Processing Systems*, 34:8367–8377, 2021. 5

[33] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *ICLR*, 2021. 3

[34] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021. 3

[35] J Arturo Olvera-López, J Ariel Carrasco-Ochoa, J Martínez-Trinidad, and Josef Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, 2010. 1

[36] Renjie Pi, Weizhong Zhang, Yueqi Xie, Jiahui Gao, Xiaoyu Wang, Sunghun Kim, and Qifeng Chen. DYNAFED: Tackling client data heterogeneity with global dynamics. *arXiv preprint arXiv:2211.10878*, 2022. 1

[37] Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *arXiv preprint arXiv:2301.04272*, 2023. 1, 2, 8

[38] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 1

[39] Shitong Shao, Huanran Chen, Zhen Huang, Linrui Gong, Shuai Wang, and Xinxiao Wu. Teaching what you should teach: A data-based distillation method. *International Joint Conference on Artificial Intelligence*, 2023. 3

[40] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pages 9206–9216. PMLR, 2020. 1

[41] Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2021. 3

[42] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019. 1, 6, 8

[43] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022. 3, 6, 8

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

[45] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 9

[46] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *arXiv preprint arXiv:2301.07014*, 2023. 1, 3

[47] Mengyang Yuan, Bo Lang, and Fengnan Quan. Student-friendly knowledge distillation. *arXiv preprint arXiv:2305.10893*, 2023. 3

[48] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021. 3, 5, 6, 8

[49] Bo Zhao and Hakan Bilen. Synthesizing informative training samples with gan. *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research*, 2022. 3

[50] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023. 3, 6, 8

[51] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021. 3, 6, 8

[52] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022. 3, 6, 8

[53] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020. 1

## 6. Limitations and Future Work

Although we have made significant improvements based on the currently best-performing MTT framework, focusing on enhancing the generation of expert trajectories and parameters matching, there are still two limitations to address.

Firstly, the expert trajectories allocate a substantial amount of storage space. Storing just five out of the 50 trajectories, each comprising 50 weight iterations, already requires 800MB. Hence, future research should explore methods to achieve similar results with fewer expert trajectories, perhaps even one or two.

Secondly, there is still room for improvement in the performance of distilled images across various evaluation models. Currently, ConvNet shows the best results. Ensuring that distilled datasets consistently exhibit competitive performance on different evaluation models is critical for development of dataset distillation. Future work might involve leveraging prior knowledge to enhance the distilled dataset generation.

## 7. Models

We employ the ConvNet architecture in the distillation process, in line with other methods mentioned in Sec. 4.1, except for FrePo, which utilizes a different model that doubles the number of filters when the feature map size is halved. Our architecture comprises 128 filters in the convolutional layer with a $3 \times 3$ kernel, followed by instance normalization, ReLU activation, and an average pooling layer with a $2 \times 2$ kernel and stride 2. For CIFAR-10 and CIFAR-100, we adopt 3-layer convolutional networks (ConvNet-3). In the case of the Tiny ImageNet dataset with a resolution of $64 \times 64$, we employ a depth-4 ConvNet. For the ImageNet subsets with a resolution of $128 \times 128$, a depth-5 ConvNet is used.

## 8. Training Burden

While we have introduced additional plug-in modules into the distillation process, the training time for each image in our approach remains comparable to that of the original MTT. It only requires a slight increase in time for each iteration, as illustrated in Tab. 8.

| Dataset | Image per class | 1 Iter. (sec) |
|---|---|---|
| CIFAR-10 | 1 | 0.5 |
| | 10 | 0.6 |
| | 50 | 0.9 |
| CIFAR-100 | 1 | 0.6 |
| | 10 | 0.85 |
| | 50 | 1.97 |
| Tiny ImageNet | 1 | 1.15 |
| | 10 | 2.42 |

Table 8. Distillation time for each dataset and support ipc.

However, notably, our method exhibits faster convergence and superior results. As depicted in Fig. 7, our approach on CIFAR100 essentially attains the final performance of the original MTT after just 500 iterations, while the original MTT requires 5000 iterations to achieve the same level of performance. Moreover, our method consistently demonstrates improvement. For example on TinyImageNet, our approach's performance continues to ascend, in contrast to MTT, which essentially plateaus after 3000 iterations.

## 9. Examples of Training Instability

The sources of training instability can be attributed to two main factors. Firstly, the original MTT itself is prone to encountering sudden spikes in gradients, which can lead to training collapse. However, by incorporating the proposed methods, our training process becomes significantly more stable. As illustrated in Fig. 8, our final loss is substantially lower than that of the original MTT, indicating a more effective transfer of expert network knowledge into the target compressed dataset.

The second source of instability stems from the parameter variations in the expert model trajectories. We utilize simple momentum-based SGD as the optimizer for training the expert model. Subsequently, we compare a crucial output during each iteration: the learning rate. From Fig. 9, it becomes evident that as the expert trajectories become less smooth, the learnable quantity experiences huge fluctuations. At around 1800 iterations, it even reaches Nan values due to sudden gradient explosions. This emphasizes the importance of generating smooth, slowly changing, and high-quality expert trajectories.

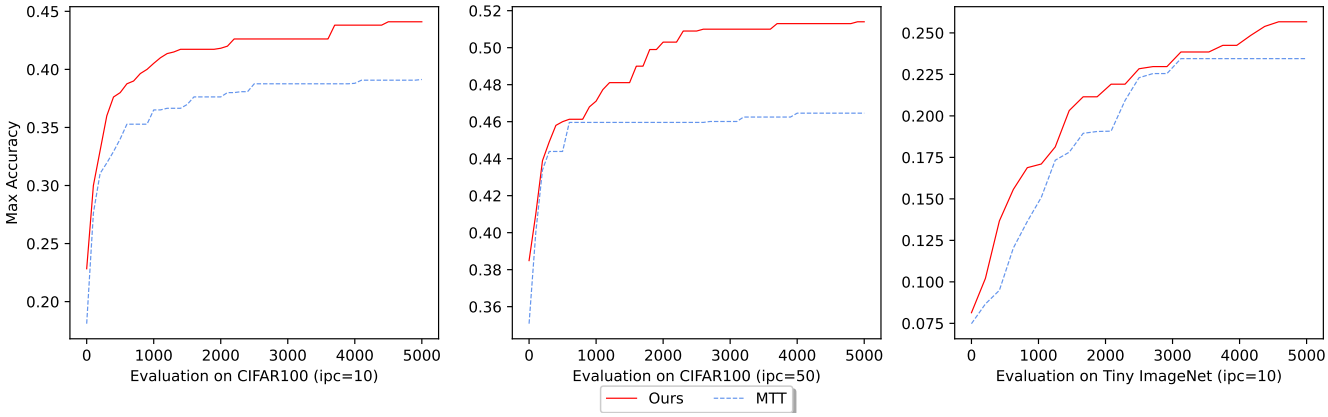



Figure 7. Applying our proposed methods brings stable performance and efficiency improvements.

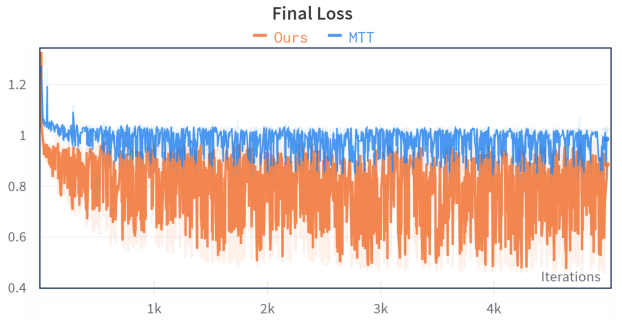
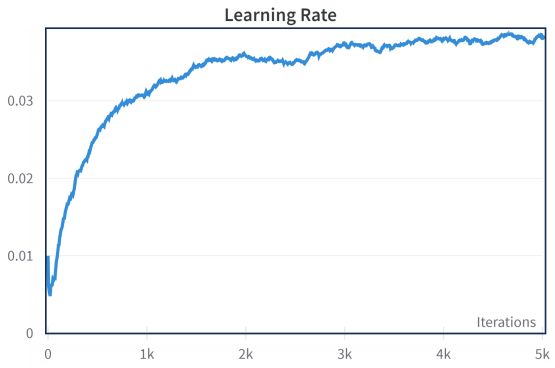
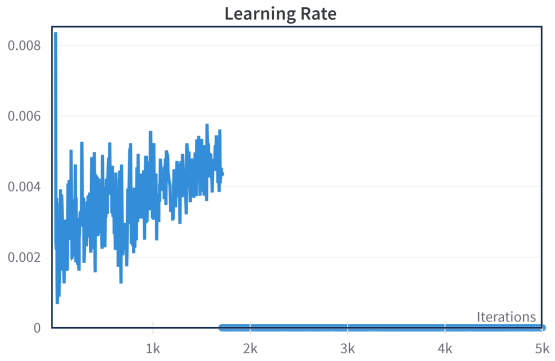



Figure 8. Comparison between proposed methods and original MTT. Our methods can generate a much lower final loss.




(a) Flat expert trajectory




(b) Non-flat expert trajectory

Figure 9. Learning curve for student model learning rate. When applying a non-flat expert trajectory, the output of the learning rate may encounter Nan which will lead to the collapse of training process.

## 10. Distilled Dataset Visualization

We visualize parts of the distilled dataset of Tiny ImageNet in Fig. 10.

Figure 10. Visualizations of synthetic images in Tiny ImageNet.