Towards Stable and Storage-efficient Dataset Distillation: Matching Convexified Trajectory

Wenliang Zhong[1], Haoyu Tang[1,*], Qinghai Zheng[2], Mingzhu Xu[1], Yupeng Hu[1], Weili Guan[3]
[1]School of Software, Shandong University, [2]College of Computer and Data Science, Fuzhou University,
[3]School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen)
zhongwenliang@mail.sdu.edu.cn
{tanghao258,xumingzhu,huyupeng}@sdu.edu.cn
zhengqinghai@fzu.edu.cn
guanweili@hit.edu.cn

Abstract

The rapid evolution of deep learning and large language models has led to an exponential growth in the demand for training data, prompting the development of Dataset Distillation methods to address the challenges of managing large datasets. Among these, Matching Training Trajectories (MTT) has been a prominent approach, which replicates the training trajectory of an expert network on real data with a synthetic dataset. However, our investigation found that this method suffers from three significant limitations: 1. Instability of expert trajectory generated by Stochastic Gradient Descent (SGD); 2. Low convergence speed of the distillation process; 3. High storage consumption of the expert trajectory. To address these issues, we offer a new perspective on understanding the essence of Dataset Distillation and MTT through a simple transformation of the objective function, and introduce a novel method called Matching Convexified Trajectory (MCT), which aims to provide better guidance for the student trajectory. MCT creates convex combinations of expert trajectories by selecting a few expert models, guiding student networks to converge quickly and stably. This trajectory is not only easier to store, but also enables continuous sampling strategies during the distillation process, ensuring thorough learning and fitting of the entire expert trajectory. The comprehensive experiment of three public datasets verified that MCT is superior to the traditional MTT method.

1. Introduction

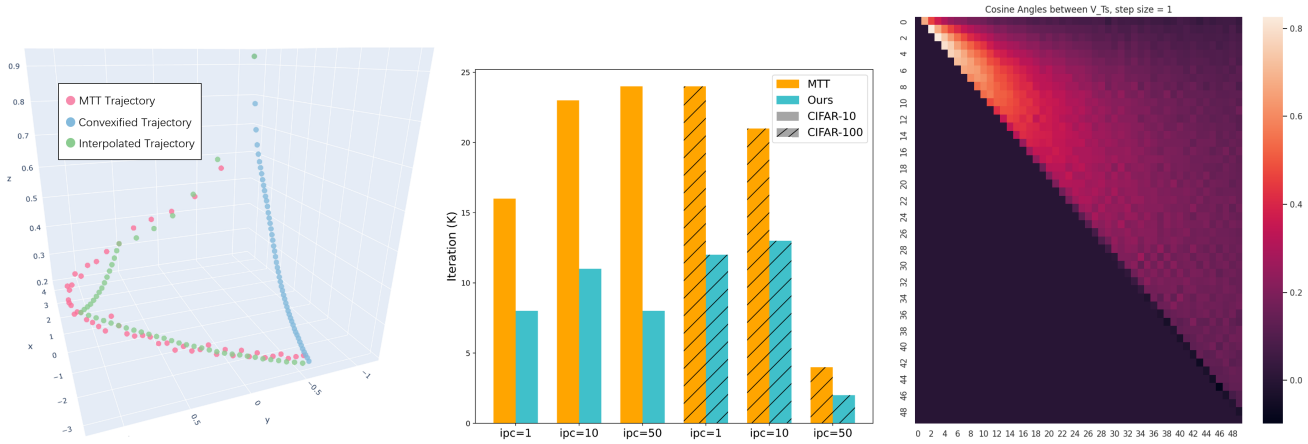
The advancement of deep learning has catalyzed an exponential surge in the requisite volume of training data [19]. With the emergence of Large Language Models (LLMs), there has been a corresponding rise in model complexity, further intensifying the demand for extensive datasets to facilitate the training of these intricate models. However, collecting and managing large datasets presents significant challenges, including storage requirements, computational load, privacy concerns, and the costs of data labeling. To mitigate these challenges, Dataset Distillation (DD) has emerged as a compelling strategy [20]. DD endeavors to distill the essence of a large, real-world dataset into a more compact, synthetic dataset that can train models with comparable efficacy.

In the landscape of DD methods, Matching Training Trajectories has emerged as a prominent approach. The MTT method aims to generate a synthetic dataset that guides the learning trajectory of the student network to approximate the expert trajectory of this network on real data. In other words, the expert trajectory should provide the guidance training dynamic during each distillation iteration, which is often defined as $\vec{V}_{\mathcal{T}} = \|\theta_{\mathcal{T}}^{(t+M)} - \theta_{\mathcal{T}}^{(t)}\|_2^2$, where $\theta_{\mathcal{T}}^{(t+M)}$ and $\theta_{\mathcal{T}}^{(t)}$ denotes the parameter of the $(t+M)$-th and t-th model on the expert trajectory. Despite the great progress [1, 6] in this field, we have identified several inherent limitations of the traditional MTT method due to poor training dynamics:

1. **Instability of Expert Trajectory**: As shown in Figure 1a, the validation accuracy of the expert network on the MTT trajectory exhibits oscillations. Matching the trajectory locally in each iteration will lead to similar oscillation in the trajectory of the synthetic data, thereby impeding robust distillation.

2. **Low Convergence Speed**: The learning process for the expert trajectory is often slow. As in Figure 1b, a considerable number of distillation iterations are required to generate a synthetic dataset capable of achieving satisfactory test accuracy, resulting in time-consuming procedures.

*Corresponding author.



(a) Visualization of the expert trajectory. (b) Illustration of Convergence Speed (c) Cosine Angle Between Training Dynamics

Figure 1. (a): PCA projection of all waypoints model in the expert trajectory, where z-axis represents the value of $(1-\text{validation accuracy})$; (b): The required number of iterations to achieve convergence for CIFAR-10 and CIFAR-100 under different ipc (images per class) for the MCT and MTT methods during distillation, respectively. The convergence is defined by the condition where the difference between the accuracy at any iteration and the maximum accuracy is less than $\epsilon = 2\%$; (c): The cosine angle between the guidance vectors on the expert trajectory, where the step size $M = 1$.

3. **High Storage Consumption**: During the distillation process, the conventional MTT approach necessitates the storage of model weights along all timesteps, which is particularly burdensome in terms of storage (about 50 models should be stored). This high storage consumption is a significant limitation for applying existing DD methods to large-scale models.

Another simple pre-experiment gave us more insights into the above issues. As the heatmap in Figure 1c, we saved all 50 midpoint models on the expert trajectory and calculated the cosine angle between all generated guidance training dynamics along this trajectory. As we can conclude, only the direction of the vectors near the beginning phase is relatively consistent. However, as training progresses, the adjacent guidance vectors often appear nearly perpendicular. These results once again confirm the **instability of expert trajectory**, which are also the reason for **low convergence speed**.

To avoid the zigzag of the guidance training dynamics, we reformulated the vanish loss function of MTT, and introduced a novel perspective to interpret the essence of DD and MTT: obtaining a synthetic dataset that offers accurate guidance regarding the magnitude and direction of the next update for any given point in the parameter space of the student model, with this guidance determined by the expert trajectory's update vector at that point. From this perspective, those three limitations can be easily addressed: to find an optimized expert trajectory that can guide the model to stably converge at each iteration, which is also easy to fit and simple to save.

How to find such a trajectory? We present a simple yet novel Matching Convexified Trajectory (MCT) method. The MCT method creates a convex combination expert trajectory based on the network's training process real data. This trajectory, which starts from a random initialization model, sequentially passes through a few intermediate points before pointing at the optimal point, facilitating stable and rapid convergence of the distillation. Moreover, recovering this trajectory only needs storing a few models and a set of constants. Distinct from the MTT method, the convexified trajectory also permits a "continuous sampling" strategy during the distillation, ensuring comprehensive learning and fitting of the expert trajectory.

The contributions of this paper are as follows: 1. We highlight the three limitations of traditional MTT methods, and offer a novel perspective for understanding the objective of DD through a simple reformulation of MTT's loss function; 2. We propose the MCT method, which creates an easy-to-store convexified expert trajectory with a continuous sampling strategy to enable rapid and stable distillation; 3. Comprehensive experiments on three datasets have verified the superiority of our MCT and the effectiveness of the continuous sampling strategy.

2. Preliminaries and Related Work

2.1. Preliminaries

We first formally define the dataset distillation task. A large scale real dataset $\mathcal{T} = \{(x_{\mathcal{T}}^{(i)}, y_{\mathcal{T}}^{(i)})\}_{i=1}^{|\mathcal{T}|}$ is first provided, where $x_{\mathcal{T}}^{(i)} \in \mathbb{R}^d$ and $y_{\mathcal{T}}^{(i)} \in \mathcal{Y} = \{1, 2, \ldots, C\}$ are the i-th instance and the corresponding label. C denotes the number of classes. The core idea of this task is to learn a

tiny synthetic dataset $\mathcal{S} = \{(x_{\mathcal{S}}^{(i)}, y_{\mathcal{S}}^{(i)})\}_{i=1}^{|\mathcal{S}|}$ from the original dataset $\mathcal{T}$, where $x_{\mathcal{S}}^{(i)} \in \mathbb{R}^d$ and $y_{\mathcal{S}}^{(i)} \in \mathcal{Y}$. Typically, ipc instances are crafted for each class, culminating in a total count for $\mathcal{S}$ of $|\mathcal{S}| = C \cdot \text{ipc}$. It is always expected that $|\mathcal{S}| \ll |\mathcal{T}|$, while $\mathcal{S}$ still preserves the majority of the pivotal information in $\mathcal{T}$. Consequently, a model trained on $\mathcal{S}$ should achieve performance comparable to the model trained with the original dataset $\mathcal{T}$ under the real data distribution $\mathcal{P}_D$. Formally, the optimization of DD task can be formulated as:

$$\arg\min_{\mathcal{S}} \mathcal{L}(\mathcal{S}, \mathcal{T}), \tag{1}$$

where $\mathcal{L}$ is the certain objective function, which may differ from different DD methods.

2.2. Dataset Distillation Methods.

The field of DD contains four principal approaches. *a.* ***Meta-model Matching methods*** [15, 16, 20, 24] involve a bi-level optimization algorithm where the inner loop updates the weights of a differentiable model using gradient descent on a synthetic dataset while caching recursive computation graphs, and the outer loop validates models trained in the inner loops on a real dataset, back-propagating the validation loss through the unrolled computation graph to the synthetic dataset. *b.* ***Distribution Matching methods*** [19, 22] align synthetic and real data by optimizing within a set of embedding spaces using maximum mean discrepancy. However, inaccurate estimation of the data distribution often results in suboptimal performance. *c.* ***Single-step Gradient Matching methods*** [21, 23] aim to align the gradient of the synthetic dataset with that of the real dataset during each training step. To enhance generalization with improved gradients, recent research efforts have focused on further optimizing the gradient matching objective by incorporating class-related information [7, 10]. *d.* ***Multi-step Trajectory Matching methods*** [1, 6] address the accumulated trajectory errors of single-step methods by matching the multi-step training trajectories of models separately trained on synthetic and real datasets.

Our research primarily focuses on multi-step trajectory matching methods. The first method in this branch is MTT [1]. Based on MTT, Du et al. [3] presented to incorporate the random noise to the initialized model weights to mitigate accumulated trajectory errors, and Cui et al. [2] proposed to decompose the objective function of MTT to improve computational efficiency and reduce GPU memory without performance degradation. Further research has explored the robustness of the synthesized dataset [4, 6, 13] and applied this technique to downstream tasks [11, 12].

Despite their successes, none of these approaches address the detriment of oscillations in the MTT expert trajectory on the stability and convergence speed of the distillation process. Furthermore, the necessity to retain all waypoint networks along the expert trajectory has yet to be addressed.

3. Motivation

3.1. Review of Multi-step Trajectory Matching

In this section, we first review the multi-step trajectory matching methods. The essence of them is to minimize the discrepancy of the student training trajectory of $\mathcal{S}$ and the expert training trajectory of $\mathcal{T}$. Here we take MTT [1] as an example. Firstly, an expert trajectory $\tau_{\text{mtt}} = \{\theta_{\mathcal{T}}^{(t)} | 0 \leq t \leq K\}$ is generated by training a randomly initialized model $\theta_{\mathcal{T}}^{(0)}$ on the real dataset $\mathcal{T}$ with K timesteps. Afterward, MTT matches the student trajectory with the expert τ_{mtt} through massive iterations. During each iteration, MTT samples a random timestep $\theta_{\mathcal{T}}^{(t)}$ and captures the target timestep $\theta_{\mathcal{T}}^{(t+M)}$ after M steps from τ_{mtt}. Meanwhile, $\theta_{\mathcal{T}}^{(t)}$ is also trained on synthetic dataset $\mathcal{S}$ for N steps to get the updated student parameters $\theta_{\mathcal{S}}^{(t+N)}$. Formally, the objective is to minimize the normalized squared L_2 error between the updated student parameters $\theta_{\mathcal{S}}^{(t+N)}$ and the future expert (target) parameters $\theta_{\mathcal{T}}^{(t+M)}$:

$$\mathcal{L}(\mathcal{S}, \mathcal{T}) = \frac{\|\theta_{\mathcal{S}}^{(t+N)} - \theta_{\mathcal{T}}^{(t+M)}\|_2^2}{\|\theta_{\mathcal{T}}^{(t)} - \theta_{\mathcal{T}}^{(t+M)}\|_2^2} \tag{2}$$

$$\theta_{\mathcal{S}}^{(t+1)} = \theta_{\mathcal{S}}^{(t)} - \alpha_{\mathcal{S}} \nabla \ell(\mathcal{S}; \theta_{\mathcal{S}}^{(t)}) \tag{3}$$

$$\theta_{\mathcal{T}}^{(t+1)} = \theta_{\mathcal{T}}^{(t)} - \alpha_{\mathcal{T}} \nabla \ell(\mathcal{T}; \theta_{\mathcal{T}}^{(t)}), \tag{4}$$

where $\theta_{\mathcal{T}}^{(t)} = \theta_{\mathcal{S}}^{(t)}$. ℓ is the loss function for model training, where the cross-entropy loss is often adopted, and $\alpha_{\mathcal{S}}$ and $\alpha_{\mathcal{T}}$ are the learning rates for training on the synthetic and real datasets, respectively. To ensure generalization, MTT usually performs the above trajectory matching process on a large number of expert trajectories from different $\theta_{\mathcal{T}}^{(0)}$. Although the subsequent methods have focused on optimizing model parameters [3, 4] and objective functions [2], the overall process remains roughly the same as MTT.

3.2. Motivation: A New Perspective to Optimize the Trajectory

Through a lot of preliminary experiments and visualizations, we found that the MTT method have three serious shortcomings: 1) Instability of the expert trajectory generated by mini-batch SGD; 2) Low convergence speed of the distillation process; 3) High storage consumption of the expert trajectory.

To better explain the internalization of these drawbacks, we propose a novel perspective to view the dataset distillation and explain the essence of the MTT approach: The objective of DD task can be regarded as obtaining a set of parameters (*i.e.*, the synthetic dataset $\mathcal{S}$) that enables accurate

prediction of how far (**magnitude**) and where (**direction**) to step next for any given network parameters θ (*i.e.*, provides appropriate guidance $\vec{V}_{\mathcal{S}}$ to update the current network parameters θ). From this perspective, each distillation iteration of the MTT method can be viewed as updating the synthetic dataset $\mathcal{S}$ to provide the network update guidance $\vec{V}_{\mathcal{S}} = \|\theta_{\mathcal{S}}^{(t+N)} - \theta_{\mathcal{T}}^{(t)}\|_2^2$ of N-step SGD training on $\mathcal{S}$, which aligns closer to the M-step SGD guidance $\vec{V}_{\mathcal{T}} = \|\theta_{\mathcal{T}}^{(t+M)} - \theta_{\mathcal{T}}^{(t)}\|_2^2$ obtained from the expert trajectory, given an arbitrary initialized point $\theta_{\mathcal{T}}^{(t)}$. A simple reformulation of Equ. 2 yields the same result:

$$\{(\theta_{\mathcal{T}}^{(t)}, \vec{V}_{\mathcal{T}}^{(t)}) | \theta_{\mathcal{T}}^{(t)} \in \tau_{\text{mtt}}, 0 \leq t \leq K\} \tag{5}$$

where $\vec{V}_{\mathcal{T}}^{(t)}$ denotes the "label" of $\theta_{\mathcal{T}}^{(t)}$.

From this perspective, the first two drawbacks can be easily explained: Given that the models on the expert trajectory τ_{mtt} are all obtained by SGD training, and considering the variations in sample distribution across mini-batches, the expert trajectory τ_{mtt} has huge oscillations. Therefore, the training dynamics $\vec{V}_{\mathcal{T}}^{(t)}$ obtained by sampling two arbitrary points with an interval of M steps from τ_{mtt} cannot guarantee to always provide a favorable direction for $\vec{V}_{\mathcal{S}}^{(t)}$ to learn. The final result is: 1) poor $\vec{V}_{\mathcal{T}}^{(t)}$ leads to instability; 2) considerable time is expended in identifying the optimal optimization direction to achieve convergence. This raises the question: Is there a superior trajectory $\hat{\tau}$ that consistently delivers more advantageous $\vec{V}_{\mathcal{T}}^{(t)}$ to optimize the synthetic dataset $\mathcal{S}$ through $\vec{V}_{\mathcal{S}}^{(t)}$?

We believe that an ideal expert trajectory should: 1) For any $\hat{\theta}_{\mathcal{T}}^{(t)}$ on $\hat{\tau}$, the obtained $\vec{V}_{\mathcal{T}}^{(t)}$ should always point to the direction that guides the target loss $\ell(\mathcal{T}; \theta_{\mathcal{T}}^{(t)})$ to decrease. 2) This trajectory is easier to fit for $\mathcal{S}$, because the size of $\mathcal{S}$ is much smaller than the original dataset $\mathcal{T}$. 3) The trajectory cannot be too simple, since it should capture various patterns of real dataset. 4) The trajectory is easy to save and restore.

First, since deep learning is essentially a non-convex problem, if we can make expert trajectories exhibit more convex properties, optimization becomes much less difficult. How to find convex trajectories? We believe that replacing the original trajectory with a convex combination trajectory would be much more effective. The starting and ending points of this trajectory are the same as τ_{mtt}, and all the waypoints are distributed along this line. This trajectory meets our needs very well: 1) The visualization in Figure 1a verifies that the validation accuracy of the model on this trajectory consistently increases. 2) The direction of any $\vec{V}_{\mathcal{T}}^{(t)}$ sampled from this trajectory is always from the starting point to the ending (optimal) point, which is easy to fit for distilled data. 3) Only the parameters of its starting and ending points need to be stored, and the trajectory can be reconstructed by linear interpolation. 4) This trajectory is continuous, rather than consisting of intermittently sampled points like the original path, which greatly enriches our training set.

4. Our proposed MCT Method

4.1. Matching Convexified Trajectory

The expert trajectory τ_{mtt} is pre-generated with the parameter of all waypoint models stored in memory, *i.e.*, $\tau_{\text{mtt}} = \{\theta_{\mathcal{T}}^{(0)}, \theta_{\mathcal{T}}^{(1)}, \ldots, \theta_{\mathcal{T}}^{(t)}, \ldots, \theta_{\mathcal{T}}^{(K)}\}$, where $\theta_{\mathcal{T}}^{(t)}$ is computed by multiple steps of mini-batch SGD [1].

However, the trajectory generated by vanilla mini-batch SGD exhibits strong non-convexity, which makes synthetic data challenging to converge to an optimal solution. To this end, we proposed MCT, which creates a convexified trajectory τ_{conv} and is defined as:

$$\begin{aligned} \tau_{\text{conv}} &= \{\hat{\theta}^{(t)} | 0 \leq t \leq K\}, \\ \forall \hat{\theta}^{(t)} &= (1 - \beta^{(t)})\theta_{\mathcal{T}}^{(0)} + \beta^{(t)}\theta_{\mathcal{T}}^{(K)}, \end{aligned} \tag{6}$$

where $\beta^{(t)} \in (0, 1)$ is a weight value that determines the distribution of all waypoints. The starting point $\hat{\theta}^{(0)}$ and ending point $\hat{\theta}^{(K)}$ are same as $\theta_{\mathcal{T}}^{(0)}$ and $\theta_{\mathcal{T}}^{(K)}$ in τ_{conv}. Particularly, the generated trajectory τ_{conv} directly points from $\theta_{\mathcal{T}}^{(0)}$ to $\theta_{\mathcal{T}}^{(K)}$, and $\beta^{(t)}$ is determined by the ratio of the difference between $\theta_{\mathcal{T}}^{(t-1)}$ and $\theta_{\mathcal{T}}^{(t)}$ in τ_{mtt} to the total length of τ_{mtt} as:

$$\begin{aligned} \beta^{(0)} &= 0, \\ \beta^{(t)} &= \frac{\sum_{l=0}^{t-1} \text{Norm}(\theta_{\mathcal{T}}^{(l+1)} - \theta_{\mathcal{T}}^{(l)})}{\sum_{l=0}^{K-1} \text{Norm}(\theta_{\mathcal{T}}^{(l+1)} - \theta_{\mathcal{T}}^{(l)})}, t = 1, 2, \ldots, K, \end{aligned} \tag{7}$$

where $\text{Norm}(\cdot)$ is L_2 normalization. To mitigate discrepancies among different network layers, we calculate the L_2 normalization for each layer individually, *i.e.*, $\beta^{(t)} = [\beta_1^{(t)}, \beta_2^{(t)}, \ldots, \beta_n^{(t)}]^{\mathsf{T}}$, where each element in $\beta^{(t)}$ represents the weight value of a network layer. Note that our trajectory is generated based on τ_{mtt}, and the calculation of $\beta^{(t)}$ does not require saving all the intermediate models $\theta_{\mathcal{T}}^{(t)}$. It only needs to save $\text{Norm}(\theta_{\mathcal{T}}^{(l+1)} - \theta_{\mathcal{T}}^{(l)})$ obtained in each step of the expert trajectory τ_{mtt}, allowing $\beta^{(t)}$ to be calculated at the end of expert training. Given this expert trajectory τ_{conv}, the distillation in Equ. 2 can be conducted. During distillation, our MCT method always provides a convexified guidance $\vec{V}_{\mathcal{T}}^{(t)}$ with the direction from $\theta_{\mathcal{T}}^{(0)}$ to $\theta_{\mathcal{T}}^{(K)}$, leading to the steady optimization of $\vec{V}_{\mathcal{S}}^{(t)}$, and thus, the convergence of $\mathcal{S}$ will be rapid.

4.2. Interpolated Points

To enhance the full exploration of the model parameter space by our linearly convexified trajectories (this explo-

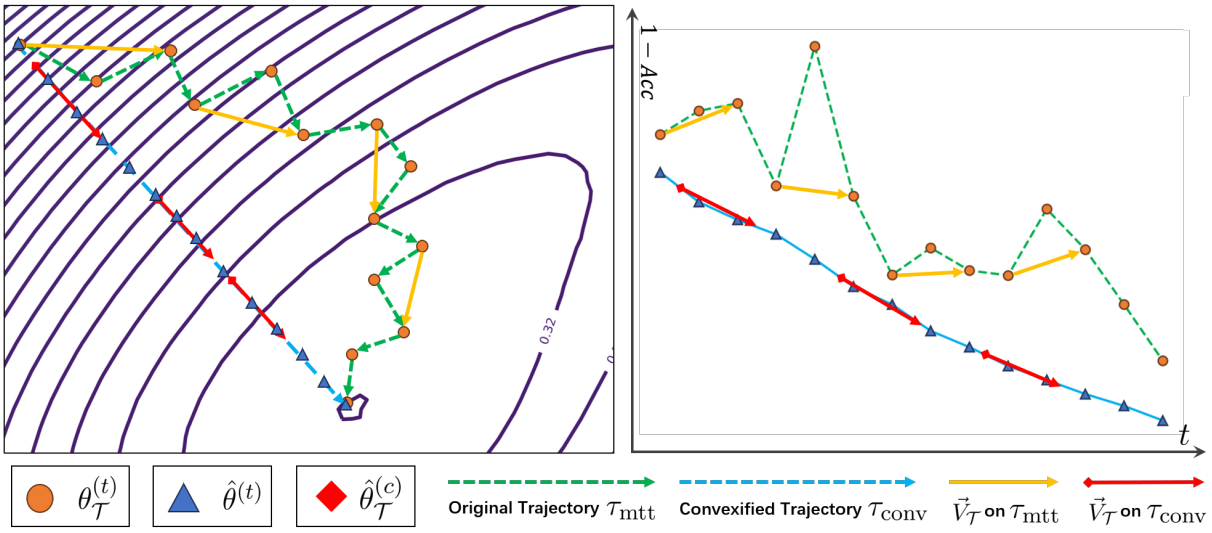


Figure 2. An illustration of the proposed MCT method. The left figure illustrates a schematic of the landscape in the model parameter space, while the right figure shows the validation accuracy of waypoint models extracted from expert trajectories of both the MTT method and our MCT method. In the left figure, the original trajectory τ_{mtt} exhibits constant oscillations, causing $\vec{V}_{\mathcal{T}}^{(t)}$ to continuously change, resulting in fluctuating accuracy of the expert model in the right figure. In contrast, the trajectory τ_{conv} of our MCT method is very stable, thereby ensuring a consistent guidance direction, which leads to a steady improvement of the expert model as shown in the right figure.

ration is also important for data distillation), we also sample a small number of interpolated points on τ_{mtt}. This process turns the linear trajectory described above into polyline segments, which enhances the trajectory's exploration of the model parameter space with only a small increase in storage requirements. Formally, in addition to the starting and ending points $\theta_{\mathcal{T}}^{(0)}$ and $\theta_{\mathcal{T}}^{(K)}$, we sample a few interpolated points $\{\theta_{\mathcal{T}}^{(p_i)}\}_{i=1}^{I}$ on the original expert trajectory τ_{mtt}, where $p_i \in (0, K)$ is the index of i-th interpolated point and I is the number of interpolated points. Thereafter, an interpolated trajectory τ_{int} is formulated as:

$$\begin{aligned}\tau_{\text{int}} =&\tau_{\text{conv}}[\theta_{\mathcal{T}}^{(0)} \rightarrow \theta_{\mathcal{T}}^{(p_1)}] + \tau_{\text{conv}}[\theta_{\mathcal{T}}^{(p_1)} \rightarrow \theta_{\mathcal{T}}^{(p_2)}] + \cdots + \\ &[\theta_{\mathcal{T}}^{(p_I-1)} \rightarrow \theta_{\mathcal{T}}^{(p_I)}] + [\theta_{\mathcal{T}}^{(p_I)} \rightarrow \theta_{\mathcal{T}}^{(K)}]\end{aligned} \tag{8}$$

where $\tau_{\text{conv}}[\theta_{\mathcal{T}}^{(p_{i-1})} \rightarrow \theta_{\mathcal{T}}^{(p_i)}]$ denotes the linear segment of τ_{int} between the interpolated points $\theta_{\mathcal{T}}^{(p_{i-1})}$ and $\theta_{\mathcal{T}}^{(p_i)}$. This linear segment has the similar definition as that of τ_{conv} as follows:

$$\begin{aligned}&\tau_{\text{conv}}[\theta_{\mathcal{T}}^{(p_{i-1})} \rightarrow \theta_{\mathcal{T}}^{(p_i)}] = \{\hat{\theta}^{(t)} | p_{i-1} \leq t \leq p_i\}, \\ &\forall \hat{\theta}^{(t)} = (1 - \beta_{p_i}^{(t)})\theta_{\mathcal{T}}^{(p_{i-1})} + \beta_{p_i}^{(t)}\theta_{\mathcal{T}}^{(p_i)},\end{aligned} \tag{9}$$

where $\beta_{p_i}^{(t)}$ determines the chosen waypoints on the linear segment $\tau_{\text{conv}}[\theta_{\mathcal{T}}^{(p_{i-1})} \rightarrow \theta_{\mathcal{T}}^{(p_i)}]$, which is calculated in a manner similar to Equ.7. In training, the proportion of sampled waypoints on this segment relative to the total K matches the proportion of the segment's projection on the convexified trajectory τ_{conv} relative to its total length. To maintain trajectory simplicity and storage efficiency, we uniformly sample only two interpolation points, *i.e.*, $I = 2$. We have also conducted some experimental explorations regarding the quantity for the selection of interpolation points. More details can be found in Appendix.

4.3. Continuous Sampling

Due to the continuity of our convexified trajectory, we can perform continuous sampling from the trajectory during distillation. This approach is completely different from the MTT method, enabling the selection of intermediate positions such as "the 1.5th point". Specifically, the MTT method only performs discrete sampling on the expert trajectory (*i.e.*, selecting $\theta_{\mathcal{T}}^{(t)}$ with an integer t). In contrast, for τ_{conv} with the starting point $\hat{\theta}^{(0)}$ and ending point $\hat{\theta}^{(K)}$, since all points are on a straight line, we can obtain any timestep $\hat{\theta}^{(c)}$ with a real-valued $c \in [0, K]$ on this line by interpolation:

$$\begin{aligned}\hat{\theta}^{(c)} &= (1 - \hat{\beta})\hat{\theta}^{(0)} + \hat{\beta}\hat{\theta}^{(K)}, \\ \hat{\beta} &= (1 - \eta)\beta^{(\lfloor c \rfloor)} + \eta\beta^{(\lceil c \rceil)}, \\ \eta &= c - \lfloor c \rfloor,\end{aligned} \tag{10}$$

After $\hat{\theta}^{(c)}$ and $\hat{\theta}^{(c+M)}$ are obtained, the distillation process can be conducted. This continuous sampling strategy ensures sufficient learning and fitting of the entire expert trajectory τ_{conv}, facilitating thorough learning of the synthetic dataset $\mathcal{S}$.

4.4. Memory-Efficient Storage

In conventional MTT, the learning of the expert trajectory requires storing the parameters of all timesteps in memory, which will incur significant storage overhead. Formally, let W denote the size of the network parameter $\theta_{\mathcal{T}}^{(t)}$ and C denote the size of other irrelevant parameters. Since there are K timesteps on τ_{mtt}, the entire required storage will be:

$$\text{S}_{\text{mtt}} = K \times W + \mathcal{C} = O(KW), \tag{11}$$

where S_{mtt} denotes the required storage of the MTT method. In contrast, our method only requires storing the starting point $\hat{\theta}^{(0)}$, the ending point $\hat{\theta}^{(K)}$, and point distribution $\{\beta^{(t)} | 0 \leq t \leq K\}$ along the trajectory. Therefore, the entire storage cost becomes:

$$\text{S}_{\text{conv}} = 2 \times W + I \times W + K \times (\beta^{(t)}) + \mathcal{C}, \tag{12}$$

where S_{conv} denotes the required storage of our method and $\mathcal{C}$ denotes a constant. Since $\beta^{(t)}$ is a floating-point number, the storage cost will be $\text{Storage}_{\text{conv}} = O(W)$. In practice, K is usually set to 50. Once the surrogate models in distillation become complex (*e.g.* LLMs), K and W will increase simultaneously, highlighting the significant storage advantages of our MCT method.

5. Experiment

5.1. Experiment Setup

Experiment Settings: We evaluated our method on three datasets: CIFAR-10 and CIFAR-100 [8], and Tiny-ImageNet [9]. We first generated the convexified trajectories with our MCT method. Similar to MTT, we applied Kornia [17] Zero component analysis (ZCA) whitening on CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets, and utilized Differentiable Siamese Augmentation (DSA) [21] technique during training and evaluation.

Evaluation and Baselines: Our MCT method is compared with several baselines from different branches, including Dataset Condensation (DC) [23], Distribution Matching (DM) [22], DSA [21], Condense Aligning FEatures (CAFE) [19], dataset distillation using Parameter Pruning (PP) [14], and MTT. Following the conventional settings, we conducted dataset distillation using 1/10/50 images per class (ipc) for evaluations, respectively. The images with the resolution of 32 × 32 and 64 × 64 were synthesized on the CIFAR and Tiny-ImageNet datasets, respectively. Subsequently, five randomly initialized networks were trained in 1000 iterations with the cross-entropy loss on the distilled dataset. These trained networks were then evaluated on the real validation set, and their average accuracy (Acc) was reported as the evaluation metric. To maintain consistency with MTT and DC, we use ConvNet [5] as the surrogate model. This model comprises 128 filters with a 3 × 3 kernel size. Following the filters, instance normalization [18] and ReLU activation are applied. Additionally, an average pooling layer with a kernel size of 2 × 2 and a stride of 2 is incorporated into the network.

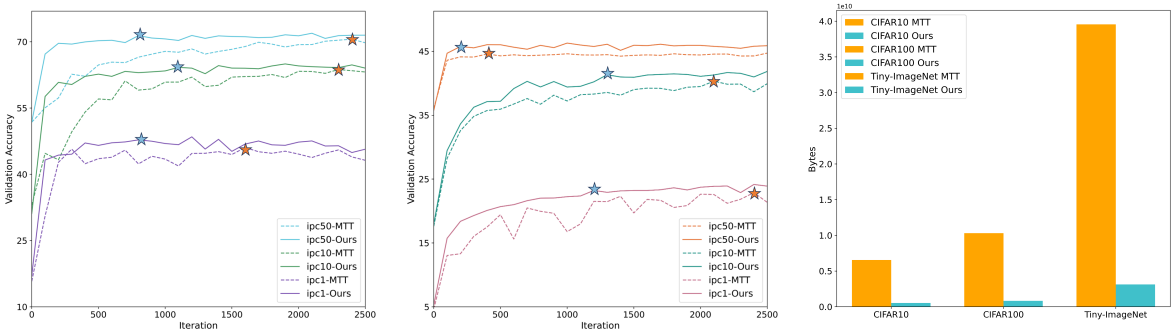
Implementation Details: We adopt the same settings of MTT in most cases. Specifically, 100 expert trajectories are generated, each spanning 50 epochs of training (*i.e.*, 51 timesteps). For large-scale synthetic data configurations (*e.g.*, CIFAR-100 with IPC $= 50$ and Tiny-ImageNet with IPC $= \{10, 50\}$), we set batch_syn $= 200$ to balance the memory cost. During the distillation process, 5,000 distillation iterations are conducted. For each iteration, $\hat{\theta}^{(c)}$ is generated from Equ. 10, where the decimal c is randomly sampled within [0, MaxStartEpoch]. We adopt the SGD optimizer, and a learnable learning rate is employed to distill the synthetic data. All experiments are run on four RTX3090 GPUs. The algorithm and more hyper-parameter settings are in the Appendix.

5.2. Experiment Result

Validation Accuracy Comparison. Table 1 presents a comparison of validation accuracy between our method and various baselines across three datasets, where the best results are highlighted and the second best ones are italicized. Although performance is not the main focus of our MCT method, it is evident that our method achieves the best performance on the three metrics of the CIFAR-10 dataset as well as the ipc $= 1$ metric of the Tiny ImageNet dataset. Notably, compared to the crucial MTT method, our MCT method demonstrates performance improvements in most metrics, indicating that our convexified trajectory and continuous sampling strategy can indeed provide enhanced guidance to the optimization of synthetic datasets.

Convergence of Distillation Process. Figure 3a and 3b illustrate the distillation processes utilizing the MCT and MTT methods for the CIFAR-10 and CIFAR-100 datasets. After every 100 distillation iterations, five networks with random initialization are trained on the current distillation dataset and their average accuracy on the validation set are recorded. The figures present the validation accuracy trends of both methods over the initial 2,500 iterations. As depicted, under all ipc settings, our MCT method achieves a substantial performance much sooner (200-1,200 iterations ahead), indicating a faster convergence speed; after nearing convergence, the performance of the MCT method remains consistently stable as iterations proceed, whereas the MTT method still experiences significant performance fluctuations. These two phenomena suggest that our method effectively enhances training stability and accelerates the convergence process.

Comparison of Storage Requirement. Figure 3c compares the required storage of the expert trajectory between MTT and our MCT method. As demonstrated in Sec. 4.4,



(a) Convergence on CIFAR-10 (b) Convergence on CIFAR-100 (c) Storage Comparison

Figure 3. (a) and (b): Convergence comparisons of distillation process on CIFAR-10 and CIFAR-100, where the symbol "star" denotes the convergence point. (c): Storage comparisons on three datasets.

Table 1. Performance of Various Algorithms on Different Datasets

Dataset	CIFAR-10			CIFAR-100			Tiny-ImageNet		
ipc	1	10	50	1	10	50	1	10	50
Random	15.4±0.3	31.0±0.5	50.6±0.3	4.2±0.3	14.6±0.5	33.4±0.4	1.4±0.1	5.0±0.2	15.0±0.4
DC [23]	28.3±0.5	44.9±0.5	53.9±0.5	12.8±0.3	25.2±0.3	-	-	-	-
DSA [21]	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 [19]	30.3±1.1	46.3±0.6	55.5±0.6	12.9±0.3	27.8±0.3	37.9±0.3	-	-	-
DM [22]	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	24.1±0.3
PP [14]	*46.4±0.6*	*65.5±0.3*	*71.9±0.2*	**24.6±0.1**	**43.1±0.3**	**48.4±0.3**	-	-	-
MTT [1]	46.3±0.8	65.3±0.7	71.6±0.2	24.3±0.3	40.1±0.4	*47.7±0.2*	*8.8±0.3*	**23.2±0.2**	**28.0±0.3**
Ours	**48.5±0.2**	**66.0±0.3**	**72.3±0.3**	*24.5±0.5*	*42.5±0.5*	46.8±0.2	**9.6±0.5**	*22.6±0.8*	*27.6±0.4*
Full dataset		84.8±0.1			56.2±0.3			37.6±0.4	

it is clear that our convex trajectories require significantly less memory (approximately 8%) compared to the expert trajectories needed by the MTT method. It is foreseeable that as model sizes and expert trajectories continue to grow, the space savings offered by our method will become even more substantial.

Visualization of Distilled Data. The visualization results of the synthetic data on CIFAR-10 with ipc $= 10$ and CIFAR-100 with ipc $= 1$ are presented in Figure 5a and 5b. As we can see, the synthetic set learned from our expert trajectories exhibits notable degrees of recognizability and authenticity, while it also tends to integrate various characteristic features of images within the same category.

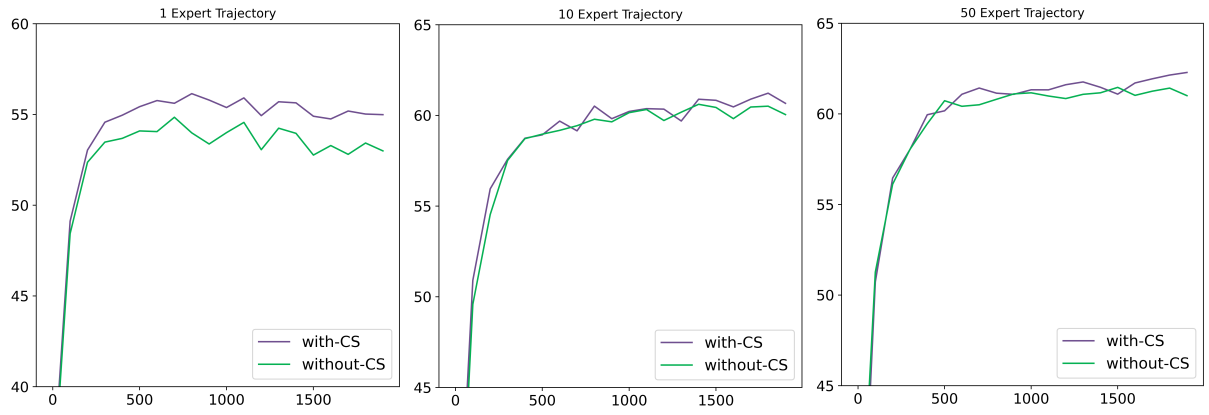
5.3. Ablation Studies

Effects of Continuous Sampling. To verify the effect of the continuous sampling, we set ipc $= 10$ and randomized the starting epoch parameter within the range $[0, 5]$ on the CIFAR-10 dataset. The validation accuracy over iterations and the optimal accuracy throughout the entire distillation process are reported in Figure 4c and Table. 2, respectively. Overall, the integration of continuous sampling can improve the validation performance under all conditions. Moreover, the fewer the number of expert trajectories, the more pronounced the performance improvement brought about by the continuous sampling strategy. Those results prove that our continuous sampling can effectively expand the sampling space, ultimately leading to the enhancement of the final distillation outcomes.

Effects of expert updating step M. Table 3 shows the effects of the updating step M of the expert trajectory τ_{conv} on the CIFAR-10 dataset. N is set to 50 for all results. As we can see, when ipc $= 1$, the optimal performance can be obtained at $M = 5$, while when ipc $= 10$ and ipc $= 50$, the optimal performance can be obtained at $M = 6$. Overall, our MCT method is robust to the selection of M and will not experience significant performance degradation with changes in M.

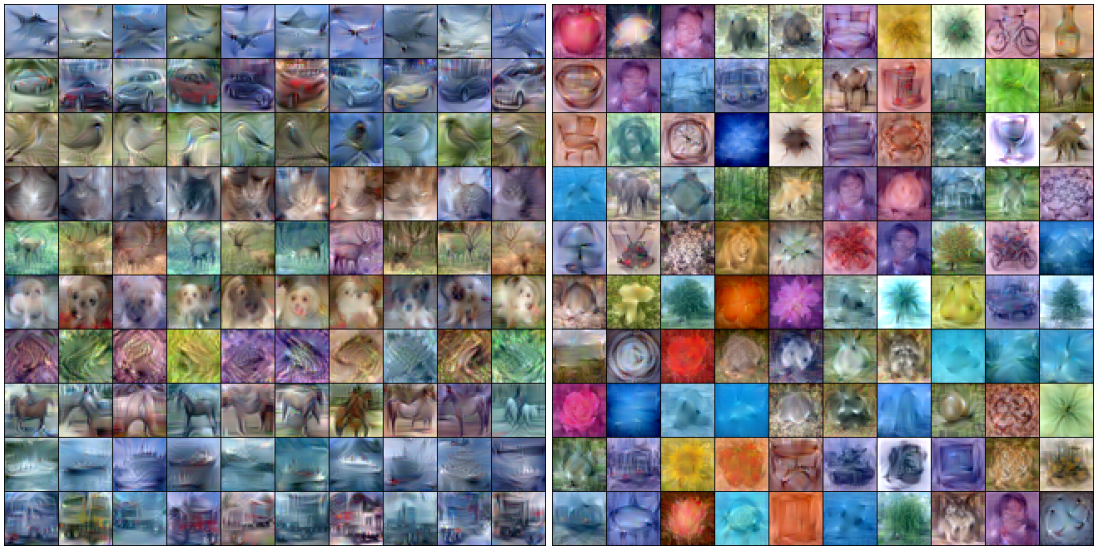
6. Conclusion

To address three major limitations of traditional MTT, this paper proposes a novel perspective to understand the essence of dataset distillation and MTT. A simple yet novel Matching Convexified Trajectory method is introduced to create a simplified, convexified expert trajectory that enhances the optimization process, leading to more stable and



(a) 1 Expert Trajectory (b) 10 Expert Trajectories (c) 50 Expert Trajectories

Figure 4. Effects of Continuous Sampling over iterations with different expert trajectory numbers.



(a) CIFAR-10, ipc=10 (b) CIFAR-100, ipc=1

Figure 5. Visualization of synthetic dataset.

Table 2. Effects of Continuous Sampling with different numbers of expert trajectories on CIFAR-10.

Number of expert trajectories	**1**	**5**	**10**	**20**	**50**
w/o. Continuous Sampling	54.8±0.2	60.6±0.2	61.5±0.3	62.3±0.3	62.1±0.4
w. Continuous Sampling	56.2±0.3	61.3±0.5	61.8±0.6	62.8±0.3	62.8±0.2

Table 3. Effects of different M with different ipc on CIFAR-10.

M	**3**	**4**	**5**	**6**	**7**
ipc $= 1$	46.7	47.1	**48.5**	48.0	45.6
ipc $= 10$	62.3	62.6	65.0	**66.0**	65.2
ipc $= 50$	70.0	71.4	71.8	**72.3**	71.8

rapid convergence and reduced memory consumption. The convexified trajectory allows for continuous sampling during distillation, enriching the learning process and ensuring thorough fitting of the expert trajectory. Our experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets demonstrate MCT's superiority over MTT and other baselines. MCT's ablation studies confirm the benefits of continuous sampling and the impact of the convexified trajectory on distillation performance.The results indicate that MCT is a promising solution for training complex models with reduced data needs, offering an efficient, stable, and memory-friendly approach to dataset distillation.

7. Acknowledgment

This work was supported in part by the National Natural Science Foundation (NSF) of China, No.62206156, No.62306074, No.62276155, No.72004127, and No.62206157; in part by the NSF of Shandong Province, No.ZR2024QF104, No.ZR2021MF040 and No.ZR2022QF047; in part by the Key R&D Program of Shandong Province, China (Major Scientific and Technological Innovation Projects), No.2022CXGC020107; in part by the Alibaba Group through Alibaba Innovative Research Program, No.21169774.

References

[1] 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, 4, 7

[2] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory. In *International Conference on Machine Learning*, pages 6565–6590. PMLR, 2023. 3

[3] Jiawei Du, Yidi Jiang, Vincent YF Tan, Joey Tianyi Zhou, and Haizhou Li. Minimizing the accumulated trajectory error to improve dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3758, 2023. 3

[4] Jiawei Du, Qin Shi, and Joey Tianyi Zhou. Sequential subset matching for dataset distillation. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[5] 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. 6

[6] Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *arXiv preprint arXiv:2310.05773*, 2023. 1, 3

[7] Zixuan Jiang, Jiaqi Gu, Mingjie Liu, and David Z Pan. Delving into effective gradient matching for dataset condensation. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2023. 3

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

[9] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. 6

[10] 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

[11] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Soft-label anonymous gastric x-ray image distillation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 305–309. IEEE, 2020. 3

[12] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Compressed gastric image generation based on soft-label dataset distillation for medical data sharing. *Computer Methods and Programs in Biomedicine*, 227:107189, 2022. 3

[13] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Dataset distillation for medical dataset sharing. *arXiv preprint arXiv:2209.14603*, 2022. 3

[14] Guang Li, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Dataset distillation using parameter pruning. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2023. 6, 7

[15] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. *Advances in Neural Information Processing Systems*, 35:13877–13891, 2022. 3

[16] 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

[17] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020. 6

[18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 6

[19] 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. 1, 3, 6, 7

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

[21] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021. 3, 6, 7

[22] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023. 3, 6, 7

[23] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2020. 3, 6, 7

[24] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *Advances in Neural Information Processing Systems*, 35:9813–9827, 2022. 3