Optimizing Trajectory Matching Distillation via Parameter Difference-Driven Pruning

Xinyu Cao 2231987@s.hlju.edu.cn
He Liu 2231978@s.hlju.edu.cn
Liyuan Zhang 2231976@s.hlju.edu.cn
Zhongliang Kan* 1992007@hlju.edu.cn
School of Computer and Big Data, Heilongjiang University

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Dataset distillation aims to give models trained on synthetic datasets the same performance as models trained with complete real datasets. Trajectory matching distillation, as an efficient dataset distillation method, achieves this goal gradually by accurately matching the dynamic trajectories of the target dataset and the synthetic dataset during the training process. Where the training trajectory is composed of the time series parameters of the agent model, and each time series contains the network parameters of all the layers in the agent model, i.e., trajectory matching distillation achieves its goal by matching the network parameters between the target dataset and the synthetic dataset. However, the variability of the training datasets used by the teacher and student networks can lead to the problem of difficult alignment of network parameters during the distillation process, so this paper proposes Difference-Driven Pruning Distillation (DPD), an innovative approach to pruning the difficult-to-align parameters according to the magnitude of the difference in parameter comparisons to alleviate the above problem. Comparative experimental results show that DPD achieves a significant performance improvement, with a greatly reduced memory footprint and superior performance in several benchmarks.

Keywords: Dataset Distillation,Data Compression,Parameter Pruning.

1. Introduction

With the rapid development of deep learning, researchers have found that for deep neural network models to achieve ideal performance, they generally require large-scale labeled datasets and powerful computing power. However, with the continuous growth of data volume, efficiently storing and processing this data has become a major challenge. This is particularly true for applications that depend on iterative training with datasets, where achieving satisfactory results often requires significant time investment, such as in hyperparameter optimization Maclaurin et al. (2015); Lorraine et al. (2019) and neural architecture search Ren et al. (2022); Zhao and Bilen (2021). To solve this problem, dataset distillation has been developed as a data compression technique.

Before the advent of dataset distillation (DD) Wang et al. (2018), coreset selection Mirzasoleiman et al. (2020); Wang et al. (2024) played a key role in dataset compression. This method creates a "coreset" by identifying a concise collection of illustrative model samples from the primary training dataset. During the training process, the model is trained

* Corresponding author.

only on this small coreset, thus reducing the amount of training data and computational cost, while avoiding significant performance degradation as much as possible.

However, because the data elements in the coreset are drawn unedited from the source data and are neither modified nor reworked,the expressive power of this approach is limited, notably if the coreset has a restricted size, and often struggles to proficiently embody the variety and intricacy of the source data.

Dataset distillation effectively addresses the shortcomings of coreset selection methods. Inspired by the classic model distillation approach Hinton et al. (2015), which transfers knowledge from a complex teacher model (typically a high-performance large model) to a smaller student model, dataset distillation extends this concept to the data level. It generates a small-scale yet sufficiently representative "distilled dataset" that can approximate the entire training dataset (often containing thousands to tens of thousands of examples). Dataset distillation extends this concept to the data level, generating a small yet sufficiently representative "distilled dataset" that encapsulates nearly all knowledge from the original training dataset (typically containing thousands to millions of images). For example, Figure 1 illustrates the encapsulation effect achieved on the CIFAR-10 dataset.

Figure 1: Visualization of synthetic images extracted from the 32×32 CIFAR-10 dataset with IPC $= 1$. IPC refers to how many images per class (images-per-class).

The process of parameter matching, as an essential optimization direction for dataset distillation, can be summarized into two stages: (1) Training trajectory: record the network parameters, i.e., the expert trajectory, when the teacher model trains the target dataset. (2) Trajectory matching: minimizing the error between the teacher information in the expert trajectory and the student information obtained from training. In the trajectory training phase, the teacher network is trained with an extensive and heterogeneous pristine dataset. In contrast, in the trajectory matching phase, the training of the student network is predicated on a condensed, distilled dataset. It is precisely the variations in the training datasets employed by the teacher network and the student network that give rise to data heterogeneity. This can result in difficulties aligning the parameters of the teacher network and the student network during the distillation process, potentially preventing the student network from fully capturing the essence of the teacher network, particularly in terms of data distribution and feature representation. This information gap leads to challenges in parameter matching, which in turn impacts the distillation effectiveness and the performance of the student model. To address this issue and achieve parameter alignment between the teacher network and the student network, it is necessary to process parameters with significant differences between the two networks.

Given that parametric pruning, as a model pruning method Sucholutsky and Schonlau (2019); Deng and Russakovsky (2022), is frequently utilized for model size reduction and speeding up model training, this paper introduces parametric pruning into trajectory matching distillation, attempting to solve the problem of difficult alignment of parameters

between the teacher network and the student network by dynamically pruning the parameters with significant differences between teachers and students, thereby enabling the student network to better adapt to the complexity and magnitude of synthetic data. This processing method eliminates secondary or redundant parameters, thereby achieving a more streamlined presentation of the student network. At the same time, it also adjusts the matching relationship between the model and the data, which improves the training effect, generates a more robust distillation dataset, and ultimately improves the distillation performance and cross-architecture generalization ability. In addition, from Figure 2, we can visualize that the memory share of Difference-Driven Pruning Distillation (DPD) during the training process is always than that of the previous DATM Guo et al. (2023) method, and at the maximum difference between the two, DPD uses only 2/3 of the memory used by DATM. This indicates that the DPD method is more efficient in memory usage, and is able to reduce memory consumption during the same training process.

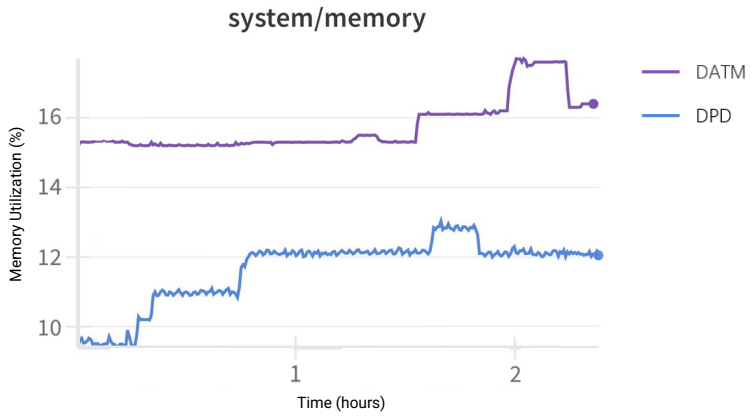

Figure 2: Memory usage of DPD and DATM methods throughout the training process when distilling the CIFAR-10 dataset with IPC=1 setting.

In this study, we introduce a difference-driven dynamic pruning-based approach to trajectory matching distillation, aiming to optimize the trajectory matching distillation process in terms of improving teacher-student network alignment. Based on the experimental results conducted, it is indicated that the approach detailed herein can largely achieve parameter alignment of the teacher network and the student network, and outperforms other SOTA dataset distillation methods on various datasets, including CIFAR-10, CIFAR-100 and Tiny ImageNet. Our main contributions are as follows:

- In this paper, the redundant parameters in the student network are effectively reduced by the pruning technique. This allows the distillation process to reduce the consumption of computational resources while maintaining the key features, thus improving the training efficiency.

- The structure of the student network was optimized through pruning to better align it with the teacher network. This enabled the student network to focus on learning

the key trajectories output by the teacher network, thereby optimizing the effect of the distillation process.

- The results of the cross-architecture experiment, Section 4.4, and the neural architecture search experiment, Section 4.5, performed on datasets of different resolutions, also demonstrate the effectiveness and generalizability of the method.

2. Preliminaries and Related Work

In the past few years, dataset distillation has also yielded a series of achievements, with various studies extending its application to multiple research domains, including continuous learning Gu et al. (2024); Yang et al. (2023) and federated learning Xiong et al. (2023); Song et al. (2023), privacy preservation Dong et al. (2022), and so on. It was initially conceived and explored by Wang et al. (2018), and at this stage, based on different implementation mechanisms, dataset distillation methods are largely structured around two technical architectures: meta-learning oriented optimization frameworks and data matching driven simulation frameworks. These two types of frameworks are fundamentally different in technical routes and can be categorized into finer-grained technologies based on specific implementation methods Yu et al. (2023).

In meta-learning frameworks, researchers consider distilled datasets as optimizable hyperparameter sets and achieve data refinement by constructing a two-layer optimization structure, e.g., DD Wang et al. (2018), LD Bohdal et al. (2020), SLDD Sucholutsky and Schonlau (2019), RTP Deng and Russakovsky (2022). The data matching framework, on the other hand, skips the explicit meta-optimization process and instead achieves data distillation through direct alignment of the parameter space or feature space. The core idea is to enable synthetic data to produce similar effects as the original training data when influencing model parameter updates by constructing appropriate matching objectives (e.g., gradient matching, feature distribution matching, or training dynamics matching).

In addition to directly optimizing the synthetic data instances themselves, in recent years researchers have proposed more scalable parametric generation paradigms to construct parametric representations of distillation data by introducing latent representation spaces and neural generators. For example, Such et al. (2021) proposed a deep generative distillation framework, which employs a noise-to-data generative network architecture and achieves a compressed representation of key features of the data distribution by jointly optimizing the generative network parameters in a meta-learning framework.

Among the above optimizations, parameter matching, as a key optimization strategy, has attracted much attention in recent years, and previous studies have laid the foundation for our work. Therefore, we will focus on the latest research progress in the direction of parameter matching.

Trajectory Match The Matching Trajectory Method (MTT) Cazenavette et al. (2022) was first proposed as a parameter matching method, introducing "expert trajectories" to direct the distillation of synthetic datasets. Its core idea is to train the synthetic data D_{syn}, beginning from a common parameter set, such that the results of N training steps on the synthetic data D_{syn} match the results of M steps on the real data D_{real}, where $\mathrm{N} \ll \mathrm{M}$. Where the expert trajectory $\mathcal{T}^*$ refers to the parameter sequence composed of different

periods of the model obtained by training a large number of models and the formula is expressed as $\mathcal{T}^* = \{\theta_i^*\}_0^n$, where θ_i^* denotes that in the real data D_{real} network parameters at the i th training step of training on it.

During every iteration of the distillation procedure , a trajectory $\mathcal{T}^*$ is randomly selected from a set of expert trajectories $\{\mathcal{T}^*\}$. Subsequently, two parameter points, θ_i^* and θ_{i+M}^*, are randomly chosen from this trajectory to serve as the starting point and target point for matching. Finally, the synthetic dataset D_{syn} is boosted to ensure that the student model's output at the parameter point $\widehat{\theta}_{i+N}$ approximates the expert output at the target parameter point θ_{i+M}^* as closely as possible. The optimization objective is the normalized squared L2 error, expressed as:

$$\mathcal{L} = \frac{\| \widehat{\theta}_{i+N} - \theta_{i+M}^* \|_2^2}{\| \theta_i^* - \theta_{i+M}^* \|_2^2}, \tag{1}$$

Where N and M are preset hyperparameters and the denominator is normalized by the distance moved by the expert to the L_2 error, $\widehat{\theta}_{i+N}$ is obtained from optimization of adaptive learning rate α and cross-entropy (CE) loss ℓ in the inner loop:

$$\widehat{\theta}_{i+N+1} = \widehat{\theta}_{i+N} - \alpha \nabla \ell(\widehat{\theta}_{i+N}, D_{\text{syn}}). \tag{2}$$

Memory optimization TESLA Cui et al. (2023), on the other hand, found that MTT could not be extended to massive datasets like ImageNet-1K, due to the high memory consumption when optimizing by unfolding the SGD steps. To solve this problem, TESLA proposed a process that precisely calculates the unfolding gradient at a constant memory complexity, making the memory complexity of calculating the MTT loss independent of step i. A novel soft label allocation (SLA) method is proposed, which initializes soft labels using logits $L_i = f_{\theta^*}(x_i)$. These soft labels are output by the pre-trained model f_{θ^*}, which is trained on data randomly selected from expert trajectories, significantly improving convergence.

Trajectory optimization FTD Du et al. (2023) conducted further analysis on the flat trajectory, revealing that although the synthetic dataset S obtained through gradient matching possesses a certain degree of generalization capability and can adapt to various initial weights, its resilience to error perturbations remains limited. The existence of cumulative trajectory error was then demonstrated to be the convergence weight error caused by the difference in starting points between the training and testing phases. The solution process was changed from finding the minimization solution of L_2 norm of the approximate initialization error in the buffer phase to finding the maximum eigenvalue equivalent to minimizing the Hessian matrix and using GSAM Zhuang et al. (2022) to help solve the expression to find the flattest possible trajectory of the teacher.

Trajectory Aligned Building upon the foundation of the former work, DATM Guo et al. (2023), as an SOTA method for dataset distillation, demonstrated that the size of the synthetic dataset (measured by IPC) impacts a model's ability to learn patterns of varying difficulty. Consequently, DATM further proposed a difficulty-aligned trajectory matching

approach to scale the size of the refined dataset to match the difficulty of the synthesis pattern. The core of this approach lies in regulating the complexity of generated patterns by constraining the scope of trajectory matching. Additionally, DATM observes that labels initialized using methods like TESLA may be incorrect (i.e., the target class is not the argmax of the logit scores). To address this issue and avoid mislabeled data, DATM selects samples correctly classified by the model f_{θ^*} to perform real-data filtering, using them to construct the subset D_{sub}. Finally, samples are randomly selected from D_{sub} to initialize $\mathcal{D}_{syn} = \{(x_i, \hat{y}_i = softmax(L_i))\}$.

From previous related work, even DATM, which has shown some advantages in dataset distillation, has neglected the treatment of irrelevant feature activation and redundant computation and failed to select and optimize model parameters effectively in distillation. If the model uses too many redundant parameters in the training process, it may learn some irrelevant patterns or noise, which affects the learning effect of the synthetic dataset, which not only increases the training time but also leads to the difficulty of model generalization. Therefore, this paper introduces the difference-driven dynamic pruning method based on the dataset distillation to try to solve the above problems.

3. Methodology

In Section 3.1 we analyze the feasibility of the pruning method applied to trajectory matching distillation through theory. Next, in Section 3.2 we describe the concrete implementation of the trajectory matching distillation method based on difference-driven dynamic pruning and show the concrete implementation process in the form of pseudo-code.

3.1. Theoretical analysis

The pruning technique projects the trajectory of the teacher model $\mathcal{T}_t$ into a sparse subspace $\mathcal{T}_t^{prune}$ with much lower dimensions than the original space by removing redundant or low contributing parameters. That is, the pruning can be viewed as a mapping:

$$\mathcal{T}_t^{\text{prune}} = \Pi(\mathcal{T}_t), \quad \Pi{:}\mathbb{R}^d \to \mathbb{R}^k (k \ll d),$$

which need to be satisfied

$$|\mathcal{T}_t - \mathcal{T}_t^{\text{prune}}|_2 \leq \epsilon |\mathcal{T}_t|_2, \quad \epsilon \ll 1.$$

After pruning, the student model only needs to match the sparsified teacher trajectory $\mathcal{T}_t^{\text{prune}}$ instead of the original high-dimensional $\mathcal{T}_t$. This allows the model's training process to reduce computational complexity while maintaining the learning of key information, especially in the environment of multiple datasets and finite sample learning. Because of this, the pruned trajectory matching distillation method has a significant advantage in optimization efficiency.

In Table 1 we also show the difference in computational complexity and efficiency between traditional dataset distillation and pruned trajectory-matched distillation by comparing the key metrics of parameter space dimensionality, dynamic system complexity, and convergence speed.

Table 1: Pruning efficiency gain analysis.

	Traditional	Pruned
Parameter Space	$\mathbb{R}^d$	$\mathbb{R}^k$
Dynamic Complexity	Match all modes[a]	Match dominant modes
Convergence Rate	$O(1/\sqrt{d})$	$O(1/\sqrt{k})$

[1] In neural network training, large eigenvalue directions (dominant modes) correspond to parameter variations that significantly affect the loss function, and small eigenvalue directions (secondary modes) may be noisy or redundant parameters.

3.2. Realization

We still use the most commonly used two-layer optimization for dataset distillation to obtain the optimal synthetic dataset. The goal of internal optimization is to make the student model fit the training data as well as possible on the synthetic dataset, and the purpose of external optimization is to construct the optimal synthetic dataset.

In the external for loop, at each iteration, a parameter, $T^- \leq i \leq T$, is randomly selected within the restricted matching range, and θ_i^* and θ_{i+M}^* are sampled from the expert trajectory as the start parameter and the target parameter for the current match. Considering that the expert parameter trajectory may contain redundant information in its later stages, we limit the step size to not exceed the upper bound I, thereby truncating the trajectory and utilizing only the information-rich portion. In the internal loop, a network that best fits the synthetic dataset, i.e., $\hat{\theta}_{i+N}$, is obtained with the help of the formula 2. After the inner loop, the parameters are pruned and the pruning methodology process has the following three steps:

1. Flatten the parameters.
   Flatten the teacher network parameters θ_{i+M}^* and student network parameters $\hat{\theta}_{i+N}$ into the following one-dimensional tensors (where a represents the total number of parameters) to enable element-wise comparison.

$$\theta_{i+M}^* = [t_{i+M,1}, t_{i+M,2}, ..., t_{i+M,a}] \tag{3}$$

$$\widehat{\theta}_{i+\mathrm{N}} = [s_{i+N,1}, s_{i+N,2}, ..., s_{i+N,a}] \tag{4}$$

2. Difference measures.
   D (i.e., differences between student and teacher parameters) was calculated for each pair of corresponding parameters using the L_2 paradigm (or some other method). We also compared the different metrics methods and the effects are shown in Section 4.3.

$$D = \| \theta_{i+M}^* - \widehat{\theta}_{i+N} \|_2 = \sqrt{\sum_{j=1}^{a} (t_{i+M,j} - S_{i+N,j})^2} \tag{5}$$

3. Pruning.
   According to the calculated difference D, the parameters are sorted in descending order according to the index. According to the *pruning_rate*, the indexes of the parameters that need to be pruned are filtered out, so that they can be pruned according to the index position. The number of parameters to be pruned is:

$$\mathrm{b} = \lfloor \mathrm{a} \times \mathrm{pruning\_rate} \rfloor, \tag{6}$$

   where *pruning_rate* is a hyperparameter used to represent the pruning rate. Finally, the result of the remaining parameters after pruning is:

$$\theta_{i+M}^{*\prime} = \left[t_{i+M,1}{}', t_{i+M,2}{}', ..., t_{i+M,c}{}' \right] \tag{7}$$

$$\hat{\theta}'_{i+N} = \left[s'_{i+N,1}, s'_{i+N,2}, ..., s'_{i+N,c} \right] \tag{8}$$

$$\theta_i^{*\prime} = \left[t'_{i,1}, t'_{i,2}, ..., t'_{i,c} \right], \tag{9}$$

   where $c = a - b$, denotes the number of remaining parameters. Our proposed pruning method is summarized visually in Figure 3.

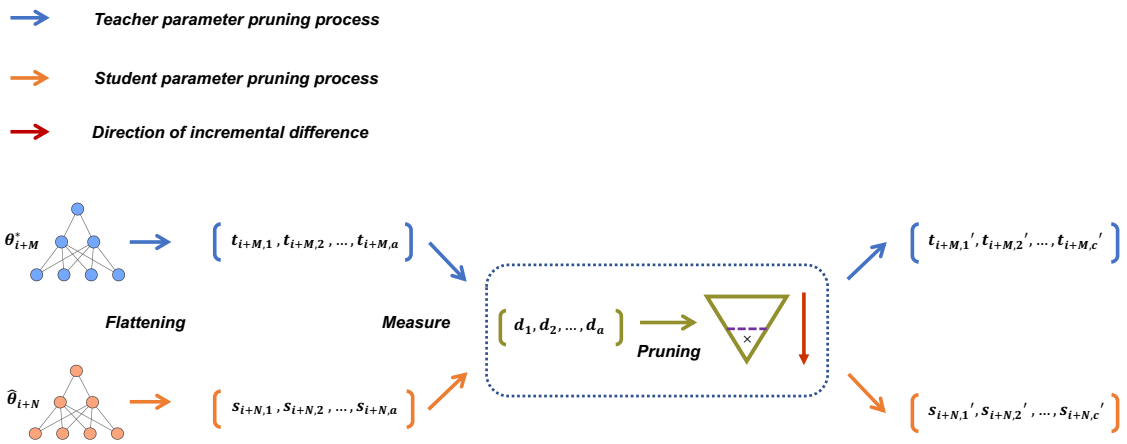

Figure 3: Alignment Process for Pruning Teacher and Student Parameters Based on Difference Metrics.

After pruning, the matching loss is computed using formula 1. Backpropagation is performed to calculate gradients, and gradient descent is employed to progressively adjust the values of synthetic data x_i and labels y_i. This process yields the synthetic dataset corresponding to the minimum loss and the final learning rate.

In the internal for loop, forward propagation is first performed to pass the input data using the network to compute the resulting values and obtain the loss. Then, based on the discrepancy between the predicted values and the actual labels, the loss value is obtained. Subsequently, through backpropagation, the gradients of the loss function regarding

each parameter are computed from the output layer back to the input layer. Using these gradients, the weights of the network are updated by an optimization algorithm (e.g., gradient descent). After many iterations of the internal for loop, a network that best fits the synthetic dataset is finally obtained, i.e., $\hat{\theta}_{t+N}$. We detail the DPD distillation process in Algorithm 1.

4. Experiments

A comprehensive experimental evaluation was undertaken to assess the efficacy of the DPD method. Initially, DPD was evaluated on a range of datasets with varying resolutions, and its performance was benchmarked against leading contemporary methods. Subsequently, cross-architecture experiments were performed to confirm the generalizability of the synthetic datasets across diverse network architectures. Furthermore, ablation studies were carried out to comprehensively illustrate DPD's performance gains and to analyze the influence of hyperparameters on the outcomes. Finally, the generated synthetic dataset was applied to Neural Architecture Search (NAS), demonstrating its robustness and consistency in this important application.

4.1. Experimental Setup

Datasets To thoroughly assess the DPD model, we conduct experiments on a range of established datasets. These include CIFAR-10 and CIFAR-100, both comprised of 32x32 color images, but differing in their number of classes (10 and 100, respectively). Furthermore, we utilize Tiny ImageNet, a more complex dataset featuring 200 classes of 64x64 images, and structured with separate training, validation, and testing sets.

Implementation Details For the generation of expert trajectories, the approach of FTD is followed and its tuning of hyperparameters is retained. Although the distillation process stabilizes as the number of internal loops increases, this is reached at the expense of increased training costs and memory demands. Therefore, we employ a sequential generation strategy, matching only the early parameters of the trajectory during the initial distillation phase to capture most of the critical information from the real data, and after enough information is embedded into the synthetic data, we gradually match the later parameters to learn some edge cases or harder-to-classify instances in the distribution. Distillation becomes more stable with the implementation of this strategy. To ensure the effectiveness of dataset distillation across different IPC scenarios, the difficulty of the generation mode must be adjusted, so the DATM approach is used to control the difficulty of generating patterns by limiting the range of trajectory matching. Use the real dataset to filter out samples that the model can correctly classify f_{θ^*} and randomly initialize $\mathcal{D}_{syn}$ with these samples. In terms of equipment, due to our limited experimental computing budget, we were only able to find GPUs that were accessible within our budget. Therefore, our experiments were conducted on four RTX3090 GPUs.

Network Architectures In terms of networks, we are consistent with previous work in that we use instance-normalized networks by default, and batch-normalized networks are used with a "-BN" suffix to the name (e.g., ConvNet-BN). Where not otherwise specified, for CIFAR-10 and CIFAR-100, ConvNet with 3 layers was used for distillation, while for Tiny ImageNet, ConvNet with a depth of 4 was used for distillation. We also used ResNet

He et al. (2016), VGG Simonyan and Zisserman (2014), and AlexNet Krizhevsky et al. (2017) for cross-architecture experiments, as described in Section 4.4.

Algorithm 1 Difference-Driven Pruning Distillation

Input :

- $\{\mathcal{T}^*\}$: set of expert parameter trajectories
- N: update times of the surrogate network in each inner optimization
- M: update times between the start and target expert parameters
- T^-: Minimum start epoch, T: current epoch, T^+: Maximum start epoch
- I: interval for expanding the sampling range

Initialize synthetic dataset $\mathcal{D}_{syn} \sim \mathcal{D}_{sub}$ (Only correctly predicted samples)
for *each distillation step* **do**
  Randomly sample an expert trajectory $\mathcal{T}^* \in \{\mathcal{T}^*\}$ with $\mathcal{T}^* = \{\theta_i^*\}_0^T$
  Choose random start epoch where $T^- \leq i \leq T$
  Initialize student network: $\widehat{\theta}_i = \theta_i^*$ (target params θ_{i+M}^*)
  for $n = 0$ **to** $N - 1$ **do**
    Sample batch $b_{i+n} \sim \mathcal{D}_{syn}$ $\widehat{\theta}_{i+n+1} = \widehat{\theta}_{i+n} - \alpha \nabla \mathcal{L}(\widehat{\theta}_{i+n}, b_{i+n})$
  end
  Prune according to formula (3)-(9)
  Calculate loss $\mathcal{L} = \frac{\|\widehat{\theta}'_{i+N} - \theta^{*\prime}_{i+M}\|_2^2}{\|\theta_i^{*\prime} - \theta^{*\prime}_{i+M}\|_2^2}$
  Update $\mathcal{D}_{syn}$ and α with $\mathcal{L}$
  if *(step%I == 0)* ***and*** *($T < T^+$)* **then**
    $T = T + 1$
  end
end
Output: Distilled dataset $\mathcal{D}_{syn}$ and learning rate α

4.2. Main Results

To evaluate the performance of the DPD method, we conducted experiments on the CIFAR-10, CIFAR-100, and Tiny ImageNet datasets, examining the impact of varying data scales (IPC = 1/10/50). We compared DPD against a suite of classical distillation methods, including DC Zhao et al. (2020), DM Zhao and Bilen (2023), KIP Nguyen et al. (2020), MTT, TESLA, FTD, and DATM. The results in Table 2 demonstrate that DPD achieves significantly superior performance compared to other methods on both the CIFAR and Tiny ImageNet datasets. Regardless of the IPC size, our method is effective on the CIFAR-10 dataset. For example, DATM is improved by 1.1% on the CIFAR-10 datasets with IPC = 50.

Table 2: Results comparing the performance of ConvNet for distillation and evaluation with other distillation methods trained on the CIFAR and Tiny ImageNet datasets. We reproduce the DATM results and cite results from other baselines reported in the DATM paper. '-' indicates that previous work did not report their results on that dataset. TI is the abbreviation of Tiny ImageNet and AD is the abbreviation of All Dataset.

Dataset	IPC	Random	DC Zhao et al. (2020)	DM Zhao and Bilen (2023)	KIP Nguyen et al. (2020)	MTT Cazenavette et al. (2022)	TESLA Cui et al. (2023)	FTD Du et al. (2023)	DATM Guo et al. (2023)	**DPD (Ours)**	AD
CIFAR-10	1	15.4±0.3	28.3±0.5	26.0±0.8	49.9±0.2	46.2±0.8	48.5±0.8	46.0±0.4	46.1±0.3	**47.0±0.5**	
	10	31.0±0.5	44.9±0.5	48.9±0.6	62.7±0.3	65.4±0.7	66.4±0.8	65.3±0.3	63.7±0.5	**64.3±0.5**	84.0±0.1
	50	50.6±0.3	53.9±0.5	63.0±0.4	68.6±0.2	71.6±0.2	72.6±0.7	73.2±0.2	73.0±0.2	**74.1±0.5**	
CIFAR-100	1	4.2±0.3	12.8±0.3	11.4±0.3	15.7±0.2	24.3±0.3	24.8±0.4	24.4±0.4	27.9±0.2	**28.6±0.4**	
	10	14.6±0.5	25.2±0.3	29.7±0.3	28.3±0.1	39.7±0.4	41.7±0.3	42.5±0.2	47.2±0.4	**47.6±0.3**	55.6±0.1
	50	33.4±0.4	-	43.6±0.4	-	47.7±0.2	47.9±0.3	48.5±0.3	55.0±0.2	**55.8±0.2**	
TI	1	1.4±0.1	-	3.9±0.2	-	8.8±0.3	-	10.5±0.2	17.1±0.3	**17.6±0.2**	
	10	5.0±0.2	-	12.9±0.4	-	23.2±0.2	-	23.4±0.3	31.1±0.3	**31.9±0.2**	37.0±0.1
	50	15.0±0.4	-	24.1±0.3	-	28.0±0.3	-	28.2±0.4	39.7±0.3	**40.8±0.2**	

4.3. Parameter and Ablation Studies

The comparison between the different difference metric methods obtained and the results with the SOTA method DATM are shown in Figure 4(a) for the CIFAR-10 dataset distilled using the 3-layer ConvNet with the settings of IPC = 1 and $pruning\_rate = 0.1$. The results are found to be most significant when the difference metric method is abs. The effect of pruning rate was again explored by setting $pruning\_method = abs$ under the same configuration, and a grid search was used to obtain the value of $pruning\_rate$ in the set {0.05,0.075,0.1,0.125,0.15}, and the experimental results showed that the effect of $pruning\_rate$ was most significant when $pruning\_rate = 0.1$, there are better results and the performance results are shown in Figure 4(b).

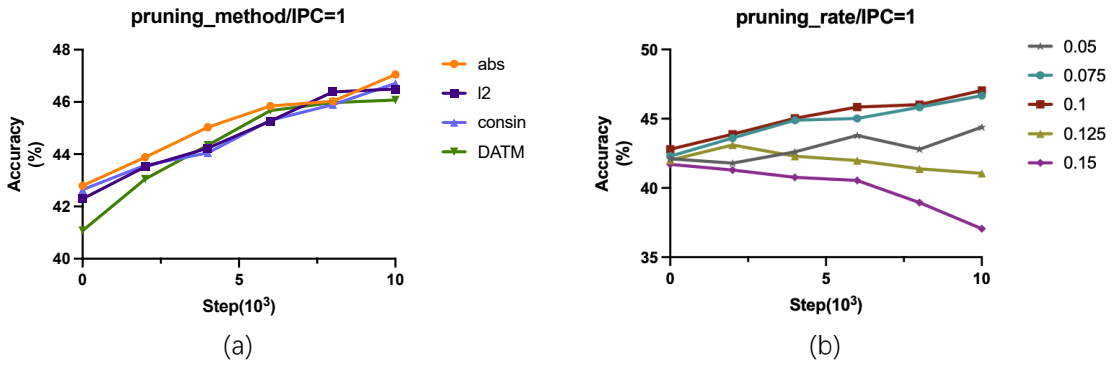

Figure 4: Comparison of different methods of measuring variance.

Such a result is because when the *pruning_rate* is large, a high pruning rate can lead to an oversimplification of the model, thus failing to capture important features or patterns in the data. This will cause the model's expressive power to be limited, which in turn does not effectively fit the training data, ultimately resulting in underfitting. Correspondingly, when the *pruning_rate* is on the small side, overfitting occurs due to a low pruning rate. The values of *pruning_rate* for all other configurations were determined by means of a grid search. More hyperparameter settings are shown in Table 3.

To further verify the effect of individual pruning components, under different IPC Settings, we performed distillation on the CIFAR-10 dataset using a 3-layer ConvNet with the default pruning Settings. As shown in the results in Table 4, the distillation effect of applying the pruning component alone is also relatively stable and can bring performance improvements to the previous methods.

Table 3: Hyperparameter settings for different datasets and IPCs.

Dataset	CIFAR-10			CIFAR-100			Tiny ImageNet		
	1	10	50	1	10	50	1	10	50
IPC	1	10	50	1	10	50	1	10	50
N	80	80	80	40	80	80	60	60	80
M	2	2	2	3	2	2	2	2	2
T^-	0	0	0	0	0	20	0	10	40
T	4	10	20	10	30	70	15	50	70
T^+	4	20	40	20	50	70	20	50	70
I	–	100	100	100	100	–	400	–	–
Learning Rate (Label)	5	2	2	10	10	10	10	10	10
Learning Rate (Pixels)	100	100	1000	1000	1000	1000	10000	100	100
pruning_rate	0.1	0.125	0.1	0.1	0.1	0.1	0.125	0.1	0.1

4.4. Cross-Architecture Generalization

Since different model architectures have different representation capabilities and structural characteristics, this mismatch may cause performance loss when migrating across architectures. Therefore, in this study, we explore the performance of DPD across architectures.

Table 4: Comparison of the effect of applying the pruning component and not applying the pruning component method under different IPC settings, the '+' after the method represents the pruning component used.

IPC	MTT	MTT+	TESLA	TESLA+	FTD	FTD+
1	45.60	46.80	48.12	48.60	46.20	46.30
10	64.50	65.28	65.98	66.78	64.38	65.48
50	71.20	72.30	71.67	72.50	72.94	73.16

The DPD method was thoroughly assessed using the CIFAR-10 benchmark dataset. For the experimental setup, three typical network architectures, ResNet, VGG, and AlexNet, were used for testing and the IPC was set to 50. For the synthetic dataset, a 3-layer convolutional neural network (ConvNet) was used for data extraction. As shown in Table 5, DPD demonstrates significant advantages through comparison experiments with the three baseline methods.

Table 5: Cross-architecture generalization results using ConvNet trained on CIFAR-100, IPC = 50. Replicated the DATM results and quoted from the DATM paper for MTT and FTD.

Method	ConvNet	ResNet18	VGG11	AlexNet
MTT	45.68	42.56	41.22	40.29
FTD	48.90	46.65	43.24	42.20
DATM	54.38	50.80	44.67	44.90
DPD(Ours)	55.16	52.47	45.26	45.23

4.5. Neural Architecture Search (NAS)

NAS is a method for automatically designing neural network architectures. The core of NAS lies in exploring various possible combinations of network structures to identify the architecture best suited for a specific task. However, evaluating each candidate architecture typically requires training on large datasets, resulting in extremely high computational costs. Data set distillation technology effectively reduces the training data volume by directly generating small subsets containing key information, thereby providing an efficient pathway for accelerating architecture evaluation in NAS.

Table 6: Top-k shows the Spearman rank correlation coefficients for the top 5, 10, and 20 searches on the synthetic and real datasets (1.00 is the best). The time column represents the total time of the entire neural architecture search process.

Method	top-5	top-10	top-15	Time(h)	Images No.
Real	1.00	1.00	1.00	113.4	50 000
DATM	0.83	0.67	0.58	5.8	500
DPD(Ours)	0.86	0.79	0.64	5.8	500

To examine the DPD method's ability to generalize across different architectural designs, we implemented NAS on CIFAR-10 with IPC set to 50. A search space containing 720 ConvNet was constructed by varying five types of parameters: width, depth, normalization mode, activation function, and pooling mode Bilen (2021). Each ConvNet undergoes training utilizing the proxy dataset, with its performance subsequently validated on the complete test set. Since the top-ranked architectures are more important, the top 20 architectures in terms of test accuracy are selected. The top 5, 10, and 20 ranked architectures are distilled from the dataset, and their search ranking correlations with the synthetic and

real datasets are used as evaluation metrics. Spearman's rho is employed to quantify the alignment between evaluation outcomes derived from the proxy and real datasets, effectively assessing their consistency.

According to the results in Table 6, DPD has a higher correlation than DATM regardless of the ranking. Especially at top-5, it possesses a correlation of 0.86, which is closer to the real dataset. This demonstrates that DPD can obtain a more stable synthetic dataset, exhibiting excellent universality for the NAS process.

5. Conclusion and Future Works

This paper addresses trajectory matching distillation with a new method. This method, by combining pruning with dataset distillation, can help the model better transfer knowledge between synthetic datasets and real datasets, avoiding compromising the model's real-world performance due to overfitting on synthetic data. Cross-architecture experiments have also confirmed that DPD has good generalization ability in different architectures.

Although the proposed pruning strategy boosts the training performance of the dataset distillation method based on optimization, when dealing with extremely large-scale datasets, the computational burden becomes significant. For example, datasets like ImageNet-1K, where the image size and the number of categories increase significantly, will slow down the entire pruning rate, and thereby make the entire distillation process sluggish. We will continue to explore and optimize data set distillation methods in future work.

References

Hakan Bilen. Explorer dataset condensation with gradient matching. 2021.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. 2020.

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.

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.

Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. *ArXiv*, abs/2206.02916, 2022.

Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022.

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.

Jianyang Gu, Kai Wang, Wei Jiang, and Yang You. Summarizing stream data for memory-constrained online continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12217–12225, 2024.

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.

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

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. 2019.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys*, 54(4), 2022.

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

Rui Song, Dai Liu, Dave Zhenyu Chen, Andreas Festag, Carsten Trinitis, Martin Schulz, and Alois Knoll. Federated learning via decentralized dataset distillation in resource-constrained edge environments. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2023.

Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *37th International Conference on Machine Learning: ICML 2020, Online, 13-18 July 2020, Part 12 of 15*, 2021.

Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. 2019.

Shaobo Wang, Yantai Yang, Shuaiyu Zhang, Chenghao Sun, Weiya Li, Xuming Hu, and Linfeng Zhang. Drupi: Dataset reduction using privileged information. *arXiv preprint arXiv:2410.01611*, 2024.

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

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16323–16332, 2023.

Enneng Yang, Li Shen, Zhenyi Wang, Tongliang Liu, and Guibing Guo. An efficient dataset condensation plugin and its application to continual learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE transactions on pattern analysis and machine intelligence*, 46(1):150–170, 2023.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. 2021.

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.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*, 2022.