MEMORY-EFFICIENT TRAJECTORY MATCHING FOR SCALABLE DATASET DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dataset distillation methods aim to compress a large dataset into a small set of synthetic samples, such that when being trained on, competitive performances can be achieved compared to regular training on the entire dataset. Among recently proposed methods, Matching Training Trajectories (MTT) achieves state-of-the-art performance on CIFAR-10/100, while having difficulty scaling to ImageNet-1k dataset due to the large memory requirement when performing unrolled gradient computation through back-propagation. Surprisingly, we show that there exists a procedure to exactly calculate the gradient of the trajectory matching loss with constant memory requirement (irrelevant to the number of unrolled steps). With this finding, the proposed memory-efficient trajectory matching method can easily scale to ImageNet-1K with $\sim$ 6x memory reduction while introducing only $\sim 2\%$ runtime overhead than original MTT. Further, we find that assigning soft labels for synthetic images is crucial for the performance when scaling to larger number of categories (e.g., 1,000) and propose a novel soft label version of trajectory matching that facilities better aligning of model training trajectories on large datasets. The proposed algorithm not only surpasses previous SOTA on ImageNet-1K under extremely low IPCs (Images Per Class), but also for the first time enables us to scale up to 50 IPCs on ImageNet-1K. Our method (TESLA) achieves 27.9% testing accuracy, a remarkable +18.2% margin over prior arts.

1 INTRODUCTION

In this paper, we study the problem of dataset distillation, where the goal is to distill a large dataset into a small set of synthetic samples, such that models trained with the synthetic samples can achieve competitive performance compared with training on the whole dataset (Wang et al., 2018). Different from core-set selection (Wolf, 2011; Rebuffi et al., 2017; Castro et al., 2018), synthetic samples are learned freely in the continuous space instead of being selected from the original dataset, so they often achieve better performance in the regime with higher compression rates (e.g., ≤ 50 synthetic images per class on CIFAR-10 which is 1% of the whole training dataset). Due to the importance of compressing a large dataset into smaller ones, many algorithms have been proposed in the past few years, including Gradient Matching (Zhao & Bilen, 2021b), Distribution Matching (Zhao & Bilen, 2021a), KIP (Nguyen et al., 2021) and Matching Training Trajectories (MTT) (Cazenavette et al., 2022). According to a recent benchmark (Cui et al., 2022), MTT achieves the best performance in terms of almost all the criteria such as accuracy, transferability, and performance under various compression ratios among all currently open-sourced methods (Wang et al., 2018; Zhao et al., 2020a; Bohdal et al., 2020; Zhao & Bilen, 2021b;a; Wang et al., 2022).

Despite achieving state-of-the-art performance, MTT cannot scale to large datasets due to its huge memory requirement (Zhou et al., 2022; Cazenavette et al., 2022; Cui et al., 2022). This is fundamentally due to the objective function of MTT, which unrolls T SGD updates with synthetic images and matches the resulting weights with a reference point (obtained by training on the original dataset). Since this objective function unrolls T optimization steps, back-propagating requires expanding and storing T gradient computational graphs in GPU memory and is prohibitive in large-scale problems. For instance, unrolling $T = 30$ steps on CIFAR-10 requires 47GB GPU memory (Cazenavette et al., 2022) with IPC 50, and thus it runs out of memory when scaling to ImageNet-1K. This has become the main issue when scaling MTT to large problems.

In this paper, we propose a memory-efficient version of MTT, which only requires storing a single gradient computational graph even when unrolling T steps. This reduces the memory complexity of MTT with respect to number of unrolled steps from linear to constant, while achieving an identical solution and with only marginal computational overhead. This is done by a novel procedure to cache and rearrange the gradient computation of the trajectory matching loss. Equipped with the proposed method, we are able to scale MTT to ImageNet-1K with 1, 2, 10, 50 IPCs. In the literature, there exists only one most recent paper that scales to ImageNet-1K with IPC up to 2, but it encounters memory and runtime issues (Section 3.3) when scaling to larger IPCs (Zhou et al., 2022).

When applying memory-efficient MTT to ImageNet-1K, we observe extremely slow convergence with sub-optimal performance when assigning hard labels to the synthetic images. We hypothesize that the missing ingredient is to use soft labels for synthesizing samples when dealing with a large number of classes, as soft labels allow information sharing across different classes. This is also observed in FrePo (Zhou et al., 2022), which jointly optimizes labels and synthetic images. However, allowing labels to be freely learned also makes the inner optimization of our matching-based method harder to solve, resulting in only marginal performance gains. To overcome this issue, we propose a soft label assignment (SLA) method that directly leverages the existing set of reference points (teacher models) in MTT for label assignment. Concretely, at every iteration, we pass the synthetic images to the sampled teacher model, and directly use its generated soft labels to guide the training of synthetic images. The proposed SLA is train-free and introduces zero hyperparameters. Empirically, the resulting algorithm significantly outperforms the original MTT on ImageNet-1K. Our contributions can be summarized below:

- We propose a novel computation method to reduce the memory requirement of MTT from $\mathcal{O}(T)$ to $\mathcal{O}(1)$, where T is the matching steps in MTT. This allows MTT to seamlessly scale to large datasets such as ImageNet-1K.
- We found assigning soft labels to synthetic images is crucial when scaling to datasets with a large number of labels (e.g., ImageNet-1K). However, naively learning soft labels works poorly for MTT. To overcome this issue, we propose Soft Label Assignment (SLA) - a novel hyperparameter-free method that directly injects soft labels into MTT from its reference models.
- By combining the above-mentioned innovations, our method, codenamed TESLA (TrajEctory Matching with Soft Label Assignment), outperforms state-of-the-art results under 1 and 2 IPCs on ImageNet-1K. Further, TESLA is the first in the field that scales to ImageNet-1K with IPC=10 and 50, 25X times larger than the next competitor.

2 Related Work

The dataset distillation problem was first formally proposed by Wang et al. (2018), where the goal is to compress a large dataset into a small set of synthetic samples. Although the compression stage could be computationally intensive, the distilled synthetic set can be used in multiple applications such as continuous learning (Wang et al., 2018; Zhao et al., 2020a), federated learning (Zhou et al., 2020; Xiong et al., 2022) and neural architecture search (Zhao & Bilen, 2021b; Wang et al., 2021). Many data distillation algorithms have been proposed in the past few years, and they can be roughly categorized into two types: matching-based approaches and kernel-based approaches.

Matching-based Approaches: Zhao et al. (2020a) tries to generate synthetic datasets by matching gradients between two surrogate models trained on distilled dataset and the real dataset. Zhao & Bilen (2021b) shows this gradient matching framework can be enhanced by introducing differentiable augmentations during training. However, matching gradient requires high memory usage and computation time, so Zhao & Bilen (2021a) further proposes to match the features generated by the surrogate model using distilled dataset and real dataset. Another recent work (Kim et al., 2022) proposes an IDC method that focuses on learning lower resolution synthetic images and upsampling, which can be applied to most of the existing methods and thus is orthogonal to our work.

Recently, Cazenavette et al. (2022) proposed a data distillation method based on Matching Training Trajectories. This method achieves state-of-the-art performance on all the medium-sized datasets (e.g., CIFAR-10, CIFAR-100) and furthermore, according to DC-bench Cui et al. (2022), MTT outperforms other published work on not only accuracy but also transferability and stability (under various kinds of augmentations and IPC settings). The main idea of MTT is to generate the synthetic dataset by directly matching the model parameters trained using synthetic datasets and real datasets. However, the scalability of MTT is limited by its high memory requirement since it involves back-propagation through T optimization steps (Cui et al., 2022; Zhou et al., 2022). Therefore, MTT fails to scale to large datasets such as ImageNet-1K.

Kernel-based Approaches: Dataset distillation is intrinsically a bi-level optimization problem, where the inner optmization computes the model parameters given the synthetic dataset, and the outer optimization optimizes the synthetic dataset to minimize the loss of the resulting model. Solving this bi-level objective is challenging if one applies an iterative inner solver such as stochastic gradient descent. Inspired by the Neural Tangent Kernel (NTK), Nguyen et al. (2020; 2021) use kernel ridge regression with NTK to obtain a closed form solution for the inner problem, and thus reducing the original bi-level optimization into a single-level optimization problem. This method is known as KIP. However, the distillation process requires thousands of GPU hours due to the NTK computation. To reduce the computational cost, FrePo (Zhou et al., 2022) only considers the neural network parameters of the last layer as learnable while keeping other parameters fixed. With this approximation, FrePo is able to obtain a closed form solution of ridge regression. Although FrePo is much faster than KIP, it still requires the storing of all the computational graphs and a heavy matrix inversion operation. Therefore it has difficulty scaling to problems with larger IPCs.

3 Method

Matching Training Trajectories: MTT (Cazenavette et al., 2022) proposes to generate the synthetic dataset by directly matching the model parameters trained using synthetic datasets with those trained on real datasets, which leads to the following loss function:

$$\mathcal{L} = \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 / \|\theta^*_t - \theta^*_{t+M}\|_2^2. \tag{1}$$

Here θ^*_t represents the model parameter trained on real images at step t. Starting from θ^*_t, $\hat{\theta}_{t+T}$ denotes the model parameter trained on the synthetic dataset after T steps and θ^*_{t+M} denotes the model parameter trained on the real dataset after M steps. The goal of MTT is to have models trained on synthetic dataset with T steps match the same results with teacher models trained from much more M steps on real data. Therefore we have $T \ll M$. We assume the model is updated by the standard SGD rule as below, where β is the student model learning rate:

$$\hat{\theta}_{t+i+1} = \hat{\theta}_{t+i} - \beta \nabla \ell(\hat{\theta}_{t+i}; \tilde{X}_i). \tag{2}$$

Here $\tilde{X}_i$ is a batch of (potentially augmented) synthetic images sampled from synthetic dataset $\tilde{X}$.

3.1 Scability of current MTT method

Although MTT achieves state-of-the-art performances on small datasets, it fails to scale to real-world large datasets such as ImageNet-1K similar to most existing condensation methods (Zhao & Bilen, 2021b;a; Nguyen et al., 2020; 2021; Wang et al., 2022). The poor scalability significantly limits its practicality.

Before presenting our method, we start by demonstrating that the bottleneck of MTT's poor scalability lies in its unrolled gradient computation. To show this, we expand the MTT loss function defined in Equation 1 as follows. θ^*_t and θ^*_{t+M} in the denominator are all from pretrained model trajectories, thus they can be treated as constants. Unrolling T steps of SGD update leads to

$$\hat{\theta}_{t+T} = \theta^*_t - \beta \nabla_\theta \ell(\theta^*_t; \tilde{X}_0) - \beta \nabla_\theta \ell(\hat{\theta}_{t+1}; \tilde{X}_1) - ... - \beta \nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}).$$

Plugging this back into Equation 1, It becomes

$$\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 \; = \; \|\theta^*_t - \beta \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) - \theta^*_{t+M}\|_2^2. \tag{3}$$

To minimize $\mathcal{L}$, MTT needs to take the derivative of equation 3 w.r.t. synthetic images. This involves computing and storing the computation graphs for T high order gradient terms, where T is the length of the trajectory. As the dataset size increases, the number of steps to train a model (trajectory length) also increases linearly, assuming everything else stays the same. As a result, **the memory requirement for optimizing MTT loss becomes extremely large as we scale to larger datasets.** Also naively reducing/fixing matching step length leads to suboptimal performance, as redundant information can be encoded into multiple images (Cazenavette et al., 2022).

3.2 Matching training trajectories with constant memory

In this section, we present a computational method to resolve the scalability issue of MTT while obtaining the same solution. Surprisingly, we found that with a careful rearrangement of the computation orders, the memory complexity of MTT can be reduced from linear to constant w.r.t. the trajectory matching step - storing only one computational graph.

As we are computing the squared error of student and teacher model parameter differences, Equation 3 can be further expanded as following

$$\begin{aligned}\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 = \sum_{i=0}^{T-1} \|\beta \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\|_2^2 - \beta(\theta^*_t - \theta^*_{t+M})^T (\sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)) + \\ \beta^2 \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)^T (\sum_{j \neq i} \nabla_\theta \ell(\hat{\theta}_{t+j}; \tilde{X}_j)) + C.\end{aligned} \tag{4}$$

Here C is a constant and it's equal to $\|\theta^*_t\|_2^2 + \|\theta^*_{t+M}\|_2^2$ in this case. It can be noticed that each term in the first two summations only involves the gradient of a single batch, so their gradients can be calculated sequentially without maintaining N computational graphs. Only the third term $\beta^2 \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)^T (\sum_{j \neq i} \nabla_\theta \ell(\hat{\theta}_{t+j}; \tilde{X}_j))$ involves two synthetic batch $\tilde{X}_i$ and $\tilde{X}_j$.

On large datasets such as ImageNet-1K, stochastic gradient descents can be used to update the student model instead of performing gradient descent using the whole synthetic dataset (Cazenavette et al., 2022). Let $\tilde{X}_i$ and $\tilde{X}_j$ be two non-overlapping synthetic batches randomly drawn from the synthetic dataset, $\sum_{j \neq i} \nabla_\theta \ell(\hat{\theta}_{t+j}; \tilde{X}_j)$ in the equation above becomes a constant with respect to $\tilde{X}_i$ and can be easily computed using $G - \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ where G is the sum of the gradients for T training steps that can be computed as $\sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$. **Thus, we are able to further reduce the memorization of computational graphs down to just 1 regardless of the matching steps**. The gradient of $\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2$ can thus be computed as the following where only 1 computational graph is needed at any point of time:

$$\begin{aligned}\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_i} &= \beta^2 \frac{\partial}{\partial \tilde{X}_i} \|\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\|_2^2 - \beta(\theta^*_t - \theta^*_{t+M})^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) \\ &+ \beta^2 (G - \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i))^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i).\end{aligned} \tag{5}$$

Therefore, in our algorithm, we will first compute the gradients $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ sequentially to get the trajectory and G. Then we conduct another pass to compute the gradient for each $\tilde{X}_i$ based on equation 5. This will reduce the memory cost while requiring two rounds of computation. However, we found that in practice making two passes only lead to negligible runtime overhead, probably because each gradient computation in our case is more light weighted (see Figure 1b).

3.3 Memory complexity v.s. other methods

In this section, we discuss our method's memory usage analytically and compare it with other methods. We focus on comparing our method with the original MTT, as well as FrePo, the only existing method that scales to ImageNet-1K under limited IPCs (1 and 2). We use T to denote SGD steps to match trajectories and $X/\tilde{X}$ to denote the whole real and synthetic dataset respectively. $\tilde{X}_i \sim \tilde{X}$

then represents a batch of data $\tilde{X}_i$ sampled from entire distilled dataset. For simplicity, we further make a moderate approximation that the memory footprint of the computation graph scales linearly w.r.t. the batch size[1], and use $\mathcal{G}$ to denote the size of computation graph for a single input image.

v.s. MTT: As MTT has to store the computation graphs for the entire matching trajectory, its memory consumption can be written as $\mathcal{O}(T|\tilde{X}_i|\mathcal{G})$ (Equation 3). For a predefined batch size $|\tilde{X}_i|$, T increases linearly w.r.t. the dataset size, which significantly limits the MTT's scalability. In constrast, our method retains a memory complexity of $\mathcal{O}(|\tilde{X}|)$, which is independently of T thanks to the loss decomposition presented in Equation 5.

v.s. FrePo: We also compare our methods with FrePo - the previous SOTA on ImageNet-1K with IPC 1 and 2. FrePo learns the synthetic images by optimizing the following loss:

$$\mathcal{L}(\tilde{X}, X) = \frac{1}{2}\|Y_t - K^{\theta}_{X\tilde{X}}(K^{\theta}_{\tilde{X}\tilde{X}} + \lambda I)^{-1}\tilde{Y}\|_2^2 \tag{6}$$

$$K^{\theta}_{X\tilde{X}} = f(X, \theta)f(\tilde{X}, \theta)^T, \qquad K^{\theta}_{\tilde{X}\tilde{X}} = f(\tilde{X}, \theta)f(\tilde{X}, \theta)^T,$$

where $f(X, \theta)$ maps X to the latent feature in the last hidden layer of a network parameterized by θ. Noticably, the second term in Equation 6 is the analytical solution to the inner optimization, hence it uses full batch (Zhou et al., 2022). It can be seen that FrePo's loss function involves the Gram matrix $K^{\theta}_{\tilde{X}\tilde{X}} \in \mathbb{R}^{|\tilde{X}|\times|\tilde{X}|}$, which is computed from feeding all synthetic images into the model. As a result, FrePo not only incurs quadratic complexity w.r.t. the synthetic dataset size, but also requires storing the computation graphs of the entire synthetic dataset in one pass. Its overall memory consumption can thus be written as $\mathcal{O}(|\tilde{X}|\mathcal{G}_{frepo} + |\tilde{X}|^2)$[2]. For ImageNet-1K with IPC 50, there are $50,000$ synthetic images, which becomes computationally prohibitive to run given its memory complexity. Moreover, in terms of runtime, FrePo's matrix inversion operation also incurs an extra cubic runtime overhead: $\mathcal{O}(|\tilde{X}|^3)$, whereas our method does not involve any superlinear terms.

3.4 Soft Labels

Using learned soft labels for synthetic images is a commonly adopted technique in kernel-based distillation methods like FrePo. Concretely, labels of the synthetic dataset are treated as a learnable parameter that can be jointly optimized with synthetic images. Compared with one-hot hard labels, the learned soft label allows information to flow across classes, thereby increasing the compression efficiency. As a result, it is shown to be critical to the performance of FrePo on datasets with large number of labels such as ImageNet-1K, especially under low IPCs. For example, FrePo reports 7.5% test accuracy on ImageNet IPC=1 with soft labels, compared with only 1.6% using hard labels.

The failure of hard labels can also be observed when we scale matched-based MTT to ImageNet-1K: we found that using hard labels on our memory efficient MTT also leads to poor results (0.7% under IPC=1). However, while kernel-based methods benefit greatly from label learning, it only shows marginal gains in our case (Section 4.4). We conjecture that, although learnable labels bring extra flexibility, updating the labels alongside with synthetic images $\tilde{X}$ and model weight $\hat{\theta}$ also makes the inner optimization of MTT more challenging to solve.

To unleash the power of soft labels for MTT, we introduce a novel train-free method for assigning soft labels to the synthetic images. Recall that the goal of MTT is to match the parameters of the student model trained on synthetic images to the teacher model trained on real images. Therefore, we can directly leverage the pre-trained teacher models to generate soft labels. Concretely, at every iteration, after sampling a trajectory of a teacher model, we pass the synthetic image to the teacher model, store the generated soft labels, and use these labels to estimate the gradients of the student model's trajectory. The gradients computed from synthetic images and their soft labels will then be used to form the MTT loss. Our method can be viewed as a form of knowledge distillation (Hinton et al., 2015), where the knowledge is distilled from the teacher model to the student model through the generated soft labels. Therefore, it not only helps with learning synthetic images, but also enriches the information condensed into the synthetic dataset.

[1] This is not strictly the case since some components of the backward graph are independent to the batch size, but the scaling law for the rest of the graph is roughly linear.

[2] Note that for a single image, the computation graph of FrePo is a bit smaller than ours since we need to back-propagation through the dot product of a gradient and a constant vector.

The proposed Soft Label Assignment (SLA) requires no additional training and does not induce any extra hyperparameters. The only design choice is which teacher model checkpoint to use for label assignment. We discuss two options below:

Teacher Model @ Target Step: Since our method samples a section of the teacher model's trajectory at every iteration, it is natural to use the teacher model at the target matching step (i.e. θ^*_{t+M}) to generate soft labels. This option is intuitive, as our objective for a single iteration is to match the teacher model at the sampled target step. Empirically, SLA using target-step teacher model achieves remarkably strong performance, leading to 7% to 13.4% absolute accuracy gain on ImageNet-1K across different IPCs.

Teacher Model @ Last Epoch: Since all teacher models are pre-trained prior to optimizing synthetic images, one may wonder whether we can always use the fully-trained teacher models to generate soft labels. Although a fully-trained teacher model outperforms its intermediate checkpoints, it could also be far away from the sampled trajectory where the matching actually occurs. As a result, the generated soft labels might not be suitable for guiding the matching process. Indeed, empirically we found that the performance of SLA using fully-trained teachers is much worse than that of target-step teacher (Figure 2a). Therefore, we use the first option for all main experiments.

The proposed algorithm, **Traj**E**ctory matching with **S**oft **L**abel **A**ssignment (TESLA), which combines the memory-efficient gradient computation of trajectory matching loss and the soft label assignment method, is summarized in Algorithm 1.

Algorithm 1 **Traj**E**ctory matching with **S**oft **L**abel **A**ssignment (TESLA)

```
Input: $f$ : teacher model; $\Theta$ : teacher model's trajectories; $K$: number of iterations; $T$: number of matching steps; $\beta$: learning rate for student model; $\alpha$: learning rate for the synthetic images.
for iter = 1 . . . K do
    Sample $\theta^*_t$ and $\theta^*_{t+M} \in \Theta$, set $G = 0$
    Initialize $\tilde{Y} = f(\theta^*_{t+M}; \tilde{X})$                ▷ Soft Label Assignment (SLA)
    for $i = 1, \ldots, T$ do
        Compute $g_i = \nabla_\theta \ell(\theta^*_t; \tilde{X}_i)$
        Update $\hat{\theta}_{t+i} = \hat{\theta}_{t+i-1} - \beta g_i;\ G = G + g_i$
    end for
    for $i = 1, \ldots, T$ do
        Compute $g_i = \nabla_\theta \ell(\theta^*_t; \tilde{X}_i)$ and keep the computational graph
        Update $\tilde{X}_i = \tilde{X}_i - \frac{\alpha}{\|\theta^*_t - \theta^*_{t+M}\|_2^2} \cdot \frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|}{\partial \tilde{X}_i}$ based on $g_i$ and equation 5
    end for
end for
```

4 Experimental Results

4.1 Experiment setup

Experiment Settings: We evaluate TESLA on 3 datasets including CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-1K (Russakovsky et al., 2015) (appendix A.2). On CIFAR-10/100, we follow other methods and learn 1/10/50 image(s) per class. For ImageNet-1K, we resize it to 64×64 resolutions following Zhou et al. (2022). We learn 10/50 images per class together with 1 and 2 that are reported by previous works. For the surrogate model, we use the same ConvNet architecture as DSA/DM/MTT. The model's convolutional layer consists of 128 filters with kernel size 3×3 followed by Instance normalization(Ulyanov et al., 2016), RELU activation and an average pooling layer with kernel size 2×2 and stride 2.

Following MTT, we apply ZCA whitening on CIFAR-10/100. On ImageNet-1K, we don't apply any data preprocessing techiniques. Simiar to MTT, we apply the same DSA (Goodfellow et al., 2016; Radford et al., 2015; Tran et al., 2020; Zhao et al., 2020b) augmentation during training and evaluation. When the dataset is simple and doesn't contain many classes such as CIFAR-10/100, soft label is not needed (Zhou et al., 2022). We find label learning most effective on ImageNet-1K.

All experiments are conducted using one single NVIDIA RTX A6000 GPU with 49GB of memory. See appendix A.12 for detailed hyperparameters.

Evaluation and baselines: Following prior works (Zhao & Bilen, 2021b;a; Cazenavette et al., 2022; Zhou et al., 2022; Cui et al., 2022), we evaluate the distilled datasets by training five randomly initialized models on them, and report the mean and standard deviation of their accuracy on the real test set. For baseline methods, we directly list numbers from their original paper when they are available. Since most prior methods do not conduct experiments on ImageNet-1K, we try our best to apply them on ImageNet-1K. Otherwise, we mark them as absent in Table 1 and Table 3. More details can be found in appendix A.9. For KIP, we use their open-sourced dataset to measure the performance since their original work uses a 1024-wide model for evaluation compared to the 128-wide model for other methods and has an extra convolutional layer. FrePo uses a model that doubles the number of filters when the feature map size is halved while other works use the same number of filters for all convolutional layers (Zhao & Bilen, 2021b;a; Cazenavette et al., 2022), thus the model used by FrePo has a lot more parameters[3] than other methods. We still report FrePo's original results due to the lack of open-sourced code and publicly available dataset.

4.2 EMPIRICAL RESULTS

We compare TESLA against previous SOTA methods and report the performance in Table 1. On smaller datasets, our method outperforms prior arts with the same model architecture. On ImageNet-1K, TESLA outperforms FrePo and DM with IPC 1 and 2. On 10 and 50 IPCs where all existing methods fail to scale, TESLA is able to achieve 17.8% and 27.9% respectively. Note that the ConvNet model trained on full ImageNet-1K can only reach 33.8% accuracy (upperbound). In this sense, TESLA can match 52.7% of the upperbound performance with only 0.78% of the whole training dataset on IPC=10, and 82.5% with only 3.9% of the training dataset.

Table 1: Test accuracies of models trained on synthetic dataset.

Dataset	IPC	Random	DSA	DM	KIP[1]	FrePo[2]	MTT	**TESLA(Ours)**[3]	Whole Dataset
CIFAR10	1	15.4±0.3	36.7±0.8	31.0±0.6	40.6±1.0 (**49.9±0.2**)	46.8±0.7	46.3±0.8	**48.5±0.8**	86.0±0.1
	10	31.0±0.5	53.2±0.8	49.2±0.8	47.2±0.7 (62.7±0.3)	65.5 ±0.6	65.3±0.7	**66.4±0.8**	
	50	50.6±0.3	66.8±0.4	63.7±0.5	57.0±0.4 (68.6± 0.2)	71.7±0.2	71.6±0.2	**72.6±0.7**	
CIFAR100	1	5.3±0.2	16.8±0.2	12.2±0.4	12.0±0.2 (15.7±0.2)	**27.2**±0.4	24.3±0.3	**24.8±0.4**	56.7±0.2
	10	18.6±0.25	32.3±0.3	29.7±0.3	29.0±0.3 (28.1±0.1)	41.3±0.2	40.6±0.4	**41.7±0.3**	
	50	34.7±0.4	42.8±0.4	43.6±0.4	-	44.3±0.2	47.7±0.2	**47.9±0.3**	
ImageNet-1K	1	0.5±0.1	-	1.5±0.1	-	7.5±0.3	-	**7.7±0.2**	33.8±0.3
	2	0.9±0.1	-	1.7±0.1	-	9.7±0.2	-	**10.5±0.3**	
	10	3.6±0.1	-	-	-	-	-	**17.8±1.3**	
	50	15.3±2.3	-	-	-	-	-	**27.9±1.2**	

[1] FrePo uses a different model with much more parameters. We still mark FrePo result as bold if it outperforms other methods.
[2] KIP's performance is measured with the dataset released by the author. Performances in quotas are from the original paper under different settings.
[3] Our performances are achieved using slightly different hyperparameters than MTT, see appendix A.12.
Entries marked as absent are due to scability issues. See appendix A.9 for detailed reasons.

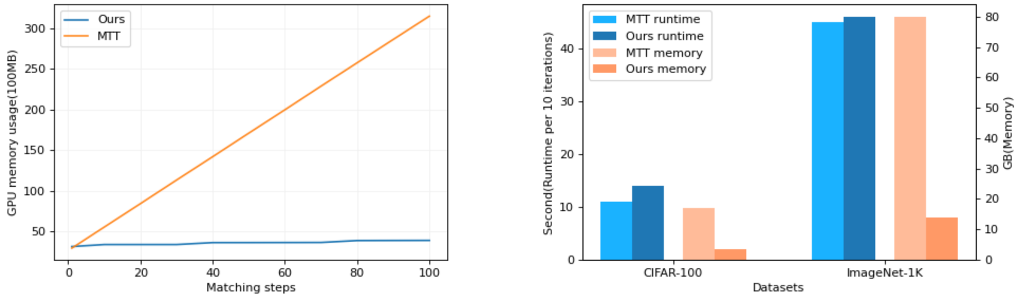
4.3 TRAINING COST ANALYSIS

As discussed in Section 3.2, a key benefit of our method over MTT is constant memory consumption w.r.t. the matching steps, with only marginal runtime overhead. In this section, we empirically benchmark and compare the memory and runtime of our methods against MTT[4].

We first compare the GPU memory consumption between our method and MTT. For this experiment, we keep everything else the same between two methods, and only vary the matching steps. The results are shown in Figure 1a (The numerical results can be found in appendix Table 4.). The memory consumption of the original MTT increases linearly with the number of synthetic steps, while our methods remains constant. This observation aligns with our theoretical analysis in Section 3.3. In principle, the constant memory reduction allows us to scale to arbitrarily large IPCs.

[3] 22.6M trainable parameters from FrePo compared to 2.5M trainable parameters from other methods on ImageNet-1K with a 4-layer ConvNet.

[4] FrePo has not released their code, so we can only compare with FrePo analytically.



(a) GPU memory usage comparison between MTT and TESLA. Results are measured on CIFAR100 with batch size 100 under different matching steps.

(b) GPU memory and runtime comparison between MTT and TESLA on different datasets. Results are measured with batch size 100 and 50 matching steps.

Figure 1: Memory and runtime comparison between MTT and TESLA.

We proceed to test the runtime overhead, alongside with memory consumption across different dataset[5]. For this experiment, we fix the synthetic training step to 50 which is one of the settings used in MTT (Cazenavette et al., 2022) and batch size to 100. The results are summarized in Figure 1b (See Appendix Table 5 for numerical results). On CIFAR-100, our method obtains $\sim$ 5x memory reduction over MTT, while only introduces $\sim 27\%$ overhead runtime. **On ImageNet-1K, TESLA obtains $\sim$ 6x memory reduction with only $\sim$ 2% extra time[6] compared to MTT**.

4.4 Ablation study on soft labels

We conduct two ablation studies on ImageNet-1K to compare the effectiveness of soft labels. First, we study our method with soft labels and hard labels and show the results in Table 2. Our method with soft labels outperforms hard labels by a large margin, e.g. 7% on IPC 1 and 13.4% on IPC 10, showing the effectiveness of soft labels. We proceed to investigate several other soft label strategies as follows.

Table 2: Ablation study on testing accuracy (%) using hard label versus soft label on ImageNet-1K. The results are measured at 1500 iterations.

	IPC			
	1	2	10	50
Hard label	0.7±0.1	1.1±0.1	4.4±0.3	18.1±1.5
TESLA	7.7±0.2	10.5±0.3	17.8±1.3	27.9±1.2

Label Learning: In this experiment, we study the strategy of learning labels instead of generating them from teacher models. We initialize the pre-softmax logits so that the probability after softmax is close to one-hot (appendix A.6). The results are plotted in Figure 2a. While learning labels do slightly improve the performance, the margin of gain is far less compared with those reported on kernel-based methods such as FrePo and KIP. The algorithm still fails to update the synthetic dataset effectively, even with the extra flexibility of the learned labels. Note that we also experiment with different label learning strategies, such as directly initializing and optimizing post-softmax labels (hence allowing each label to move beyond 0-1 range), but the results are similar.

Target (Ours) vs Last Epoch: We also study using soft labels generated by the teacher model using the target step versus the last epoch parameters. It's natural to think that a better-trained model will capture more statistics of the training data, thus generating better soft labels. However, we find out that this doesn't work with trajectory matching. As shown in Figure 2a, the algorithm also fails to learn effectively with last epoch parameters.

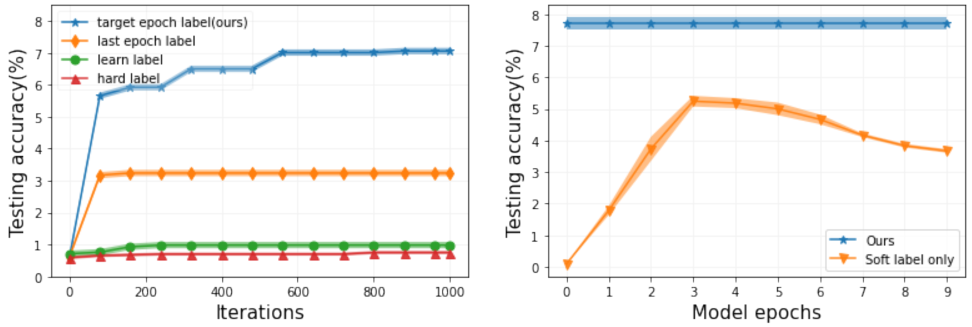
Secondly, we study the effect of using soft labels only. we fix the synthetic images and measure the impact of soft labels produced by the teacher model with parameter θ^*_{t+M}. On ImageNet-1K IPC 1, we achieve state-of-the-art performances by iteratively setting θ^*_{t+M} as parameters from one of the first 9 epochs[7] of the teacher model (SLA step in Algorithm 1). In the ablation study, we randomly select 1 image per class and generate their labels using teacher models from epoch 0 to epoch 9.

[5] We don't measure it on CIFAR-10 because synthetic dataset is too small even with IPC 50

[6] MTT's runtime on ImageNet-1K is estimated since MTT is OOM under our settings. See appendix A.7

[7] Same as MTT, we always sample θ^*_{t+M} from teacher trajectories after a full epoch. One epoch contains multiple SGD steps

The results are shown in Figure 2b. It can be seen that around 5.3% accuracy can be achieved by initializing the labels using teacher models without updating synthetic images. And our method is able to achieve around 7.7% testing accuracy by integrating soft labels with our memory-efficient implementation of MTT.



(a) Comparison on different label strategies. The Y-axis shows the maximum accuracy achieved until that iteration.

(b) Performance of SLA alone without updating synthetic images. Top flat line shows the performance of TESLA baseline.

Figure 2: Ablation study on soft labels. Experiments are conducted on ImageNet-1K with IPC 1.

4.5 Cross-Architecture Generalization

Following previous works (Zhao & Bilen, 2021a; Cazenavette et al., 2022; Zhou et al., 2022; Cui et al., 2022), we evaluate the transferability of our condensed dataset in training new architectures unseen in the synthetic dataset generation phase. The experiment is conducted on CIFAR-10, CIFAR-100 and ImageNet-1K under 10 IPCs. Besides the baseline vanilla ConvNet model, we report performance on ResNet18 and ViT (Dosovitskiy et al., 2020; Cui et al., 2022). As shown in Table 3, our method transfers well across datasets and models, outperforming previous methods by a sizable margin. This shows that the proposed method can be empirically effective in distilling generalizable information into the synthetic dataset. We are not able to get FrePo's performances due to the lack of open-sourced code and publicly available distilled dataset.

Table 3: Test accuracy of different methods on ConvNet versus transferred to other architectures. All methods are evaluated with 10 IPCs.

	CIFAR-10			CIFAR-100			ImageNet-1K		
	ConvNet	ResNet18	ViT	ConvNet	ResNet18	ViT	ConvNet	ResNet18	ViT
Random	31.0±0.5	29.6±0.9	26.2±0.5	18.6±0.3	15.8±0.2	14.1±0.2	3.6±0.1	1.4±0.1	3.2±0.0
DSA	53.0±0.4	42.1±0.6	31.9±0.4	32.2±0.4	21.9±0.4	19.6±0.2	-	-	-
DM	47.6±0.6	38.2±1.1	34.4±0.5	29.2±0.3	18.7±0.5	17.1±0.3	-	-	-
KIP	47.2±0.4	38.8±0.7	15.9 ±1.1	29.0±0.3	20.1±0.5	12.1±0.7	-	-	-
MTT	65.3±0.7	46.1±1.4	34.6±0.6	40.6±0.4	26.8±0.6	20.4±0.2	-	-	-
Ours	**66.4±0.8**	**48.9±2.2**	**34.8±1.2**	**41.7±0.3**	**27.1±0.7**	**21.0±0.3**	**17.8±1.3**	**7.7±0.1**	**11.0±0.2**

Missing entries are due to scalability issues, see appendix A.9 for detailed reasons.

5 Conclusion

We propose a novel method to reduce the current state-of-the-art method: MTT's heavy memory requirements from $O(T)$ to $O(1)$ (with respect to number of unrolling steps) with negligible overhead time. We also introduce the use of soft labels to guide the matching process of model training trajectories. By combining the two, we are able to scale dataset distillation onto ImageNet-1K with IPC 10 and 50 for the first time in the field and achieve state-of-the-art performances on IPC 1 and 2. We analyze the complexity of our methods both analytically and empirically and compare it to other methods. We also show that our distilled data transfer well to other models with completely different architectures such as ViT. We hope our method can pave the way for future works to explore and expand dataset distillation methods on large-scale real-world datasets.

Ethics Statement

The condensed dataset used in this paper are all generated from the following standard non-private dataset: CIFAR-10, CIFAR-100, and ImageNet-1K. Therefore, we are not aware of any ethical concern of the benchmark. However, the end users should be aware of potential data leakage through condensed dataset, when they try to apply any condensation methods included in our work to their tasks at hand.

Reproducibility

Our method is justified in Section 3.3 analytically and verified empirically in Section 4.3. We attach our implementation code in the supplemental materials for maximum degree of reproducibility. Our work is easily reproduible. We have described our algorithm in full detail in Algorithm 1 with exact corresponding mathematical equations. At the same time, we share our implementation details in the appendix together with the hyperparameters we use to generate the results in this work.

References

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

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

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

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. In *NeurIPS (Dataset and Benchmark)*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

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 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11102–11118. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/kim22c.html.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2020.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. In *Advances in Neural Information Processing Systems*, 2021.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

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

Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. *workshop on applications of computer vision*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. Towards good practices for data augmentation in gan training. *arXiv preprint arXiv:2006.05338*, 2:3, 2020.

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

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022*, 2022.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representation*, 2021.

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

Gert W Wolf. Facility location: concepts, models, algorithms and case studies. series: Contributions to management science. *International Journal of Geographical Information Science*, 25(2):331–333, 2011.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. Feddm: Iterative distribution matching for communication-efficient federated learning, 2022.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *arXiv preprint arXiv:2110.04181*, 2021a.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pp. 12674–12685. PMLR, 2021b.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2020a.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020b.

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

Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022.