Scaling Up Dataset Distillation to ImageNet-1K with Constant Memory

Justin Cui [1] **Ruochen Wang** [1] **Si Si** [2] **Cho-Jui Hsieh** [1]

Abstract

Dataset Distillation is a newly emerging area that aims to distill large datasets into much smaller and highly informative synthetic ones to accelerate training and reduce storage. Among various dataset distillation methods, trajectory-matching-based methods (MTT) have achieved SOTA performance in many tasks, e.g., on CIFAR-10/100. However, due to exorbitant memory consumption when unrolling optimization through SGD steps, MTT fails to scale to large-scale datasets such as ImageNet-1K. Can we scale this SOTA method to ImageNet-1K and does its effectiveness on CIFAR transfer to ImageNet-1K? To answer these questions, we first propose a procedure to exactly compute the unrolled gradient with constant memory complexity, which allows us to scale MTT to ImageNet-1K seamlessly with $\sim$ 6x reduction in memory footprint. We further discover that it is challenging for MTT to handle datasets with a large number of classes, and propose a novel soft label assignment that drastically improves its convergence. The resulting algorithm sets new SOTA on ImageNet-1K: we can scale up to 50 IPCs (Image Per Class) on ImageNet-1K on a single GPU (all previous methods can only scale to 2 IPCs on ImageNet-1K), leading to the best accuracy (only 5.9% accuracy drop against full dataset training) while utilizing only 4.2% of the number of data points - an 18.2% absolute gain over prior SOTA.

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. 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). Recently, a line of methods based on matching trajectory has demonstrated state-of-the-art performance on smaller datasets (Cui et al., 2022; Cazenavette et al., 2022): When using only 50 synthetic images per class which yields a 100x compression rate, MTT only incurs 13.4% accuracy drop compared to training on the whole CIFAR-10 dataset.

Despite achieving state-of-the-art performance, MTT cannot scale to large datasets due to its huge GPU 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 GPU 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

[1]Department of Computer Science, University of California, Los Angeles [2]Google Research. Correspondence to: Justin Cui <justincui@ucla.edu>, Cho-Jui Hsieh <chohsieh@cs.ucla.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

memory and runtime issues (Section 4.2) 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 method to reduce the memory usage 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; Zhao & Bilen, 2021b) propose to generate synthetic datasets by matching gradients between two surrogate models trained on distilled dataset and the real dataset. 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. Other recent works (Kim et al., 2022; Deng & Russakovsky, 2022; Lee et al., 2022) focus on learning lower-resolution synthetic images and upsampling, which can be applied to most of the existing methods and thus are 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 works on not only accuracy but also transferability and stability (under various kinds of augmentations and IPC settings). The idea of MTT and its scalability issues will be discussed in the next section.

Kernel-based Approaches: Dataset distillation is intrinsically a bi-level optimization problem, where the inner optimization computes the model parameters given the synthetic dataset, and the outer optimization optimizes the synthetic dataset to minimize the loss of the resulting model. 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, known as KIP, 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 faster than KIP, it still requires storing all computational graphs and a heavy matrix inversion operation. Therefore it has difficulty scaling to larger IPCs.

3. Background

3.1. Matching Training Trajectories

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:

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

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 and usually $T \ll M$. We assume the model is updated by the standard SGD rule as below, where β is the student model learning rate:

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

Here $\tilde{X}_i$ is a batch of (potentially augmented) synthetic images sampled from synthetic dataset $\tilde{X}$(See Appendix Figure 6 for an illustration).

3.2. Scability of the 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 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

$$\hat{\theta}_{t+T} = \theta^*_t - \beta \nabla_\theta \ell(\theta^*_t; \tilde{X}_0) - \beta \nabla_\theta \ell(\hat{\theta}_{t+1}; \tilde{X}_1) - ... - \beta \nabla_\theta \ell(\hat{\theta}_{t+T-1}; \tilde{X}_{T-1}). \quad (3)$$

Plugging this back into Equation (1), it becomes

$$\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 = \|\theta^*_t - \beta \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) - \theta^*_{t+M}\|_2^2. \quad (4)$$

To minimize $\mathcal{L}$, MTT needs to take the derivative of Equation (4) 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 GPU memory requirement for optimizing MTT loss becomes extremely large on 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).

4. Our proposed method

In this section, we will discuss how to handle the scalablity issue of MTT, and propose our method: TrajEctory matching with Soft Label Assignment (TESLA).

4.1. MTT with constant memory

In this subsection, we present a computationally efficient way to resolve the scalability issue of MTT while obtaining the same solution. Surprisingly, we found that with a careful rearrangement of the computation, the memory complexity of MTT can be reduced from linear to constant w.r.t. the number of trajectory matching steps - storing only one computational graph. **Note that our approach does not introduce any approximation** – the gradient computed by our method is exactly identical to the original one computed by back-propagating the entire loss. We have also empirically verified that the proposed approach leads to identical updates as the original unrolled version of MTT.

As we are computing the squared error of student and teacher model, Equation (4) can be further expanded as

$$\|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2 = 2\beta(\theta^*_{t+M} - \theta^*_t)^T (\sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)) + \beta^2 \| \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\|^2 + C, \quad (5)$$

where $C = \|\theta^*_t - \theta^*_{t+M}\|_2^2$ is a constant so can be ignored for gradient computation. It can be noticed that each term in the first summation only involves the gradient of a single batch, so their gradients can be calculated sequentially without maintaining N computational graphs. Only the second term $\| \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\|^2$ involves information from multiple batches together.

Backpropagating from MTT loss to a mini-batch Computing the gradient with respect to $\{\tilde{X}_i\}_{i=0}^{T-1}$ leads to

$$\frac{\partial \|\hat{\theta}_{t+T} - \theta^*_{t+M}\|_2^2}{\partial \tilde{X}_i} = 2\beta(\theta^*_{t+M} - \theta^*_t)^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i) + 2\beta^2 G^T \frac{\partial}{\partial \tilde{X}_i} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i), \quad (6)$$

where $G = \sum_{i=0}^{T-1} \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$. Our key finding of this expansion is as follows: since G can be precomputed in MTT (1st for-loop in Algorithm 1) while updating $\hat{\theta}$, the only computation graph required for computing (6) is the

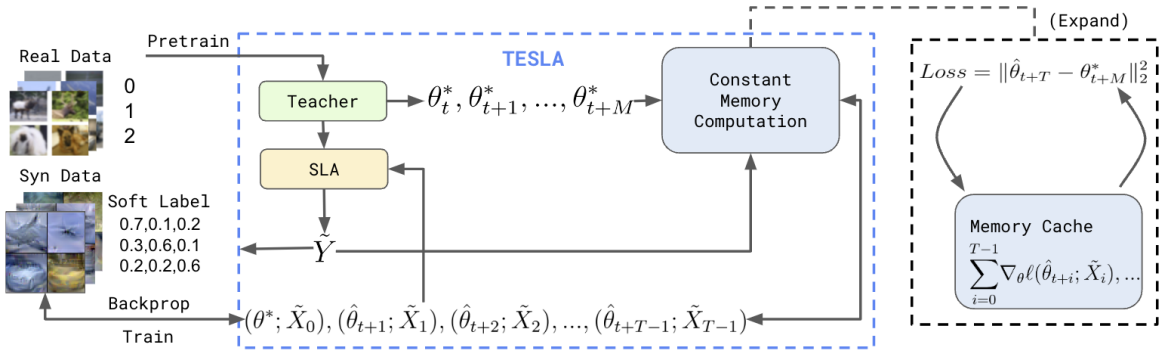

Figure 1: Illustration of trajectory matching with soft label assignment (TESLA). Our method differs from MTT in two aspects: 1). The constant memory computation module calculates the exact trajectory matching loss as MTT but with constant memory complexity. 2). The train-free Soft-Label Assignment (SLA) module leverages the pretrained teacher to distill soft labels to the synthetic data points.

gradient on batch $\tilde{X}_i$: $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$. As the computation graph of $\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$ is not required in the derivative of any other batch $\tilde{X}_{j \neq i}$, it can be released right after use. Therefore, the memory complexity of computing MTT loss becomes independent of the step t, thus constant w.r.t. T.

Backpropagating from a mini-batch to synthetic images To further backpropagate from a mini-batch to the synthetic images, we need to define the connections between batches $\{\tilde{X}_i\}_{i=0}^{T-1}$ and synthetic images $\{x_j\}_{j=1}^{n}$. Assume each batch i contains B samples denoted as $\tilde{X}_i = [\tilde{X}_{i,1}, \ldots, \tilde{X}_{i,B}]$ (concatenation of B vectors) and for all $b = 1, \ldots, B$, $\tilde{X}_{i,b} = x_{\pi(i,b)}$ where $\pi(i, b)$ indicate which data point is sampled as the b-th instance in the i-th batch. To propagate gradients to synthetic images, we have

$$\frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial x_j} = \sum_{(i,b):\pi(i,b)=j} \frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial \tilde{X}_{i,b}}. \quad (7)$$

We can see that (7) is essentially doing gradient accumulation: whenever we obtain the gradient to a batch $\tilde{X}_i$, we accumulate that gradient to x_j if $x_j \in \tilde{X}_i$. This process does not require storing any computation graphs for the backward pass. And since we only need to store one accumulated gradient vector for each synthetic image, the memory is also constant w.r.t. T.

Since both (6) and (7) do not require maintaining all T computational graphs of $\{\nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)\}_{i=0}^{T-1}$, **the overall memory complexity is constant with respect to** T. As a result, we can compute the gradient to all the synthetic images with constant memory, at the cost of computing MTT loss consecutively over each batch. In practice, we found that this consecutive computation incurs negligible runtime overhead, as the total cost is almost identical to the original MTT computation (see Figure 3).

4.2. Memory complexity v.s. other methods

In this section, we discuss our method's GPU 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 (up to 2). We also discuss about other bi-level optimization methods in A.2.

In the following discussion, 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 computation graph size 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 (4)). 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 contrast, our method retains a memory complexity of $\mathcal{O}(|\tilde{X}_i|\mathcal{G})$, which is independently of T thanks to the loss decomposition presented in Equation (6).

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:

$$\mathcal{L}(\tilde{X}, X) = \frac{1}{2}\|Y_t - K_{X\tilde{X}}^{\theta}(K_{\tilde{X}\tilde{X}}^{\theta} + \lambda I)^{-1}\tilde{Y}\|_2^2 \quad (8)$$

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

where $f(X, \theta)$ maps X to the latent feature in the last hidden layer of a network parameterized by θ. Noticably, the second

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

term in Equation (8) 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.

4.3. 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, soft labels allow 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 scaling matching-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 5.4). 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 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 in the gradient computation of Equation (3) for the student model's updates. 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.

The proposed Soft Label Assignment (SLA) requires no additional training and 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 4). Therefore, we use the first option for all main experiments.

The proposed algorithm, **T**raj**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 and Figure 1.

5. Experimental Results

5.1. Experiment setup

Experiment Settings: We evaluate TESLA on 3 datasets including CIFAR-10/100 (Krizhevsky et al., 2009) and ImageNet-1K (Russakovsky et al., 2015) (Appendix A.3). 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 ker-

[2]For a single image, the computation graph of FrePo is slightly smaller than ours since we back-propagate through all layers.

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

Input: f : teacher model; Θ : teacher model's trajectories; K: number of iterations; T: number of matching steps; β: learning rate for student model; α: learning rate for the synthetic images.
for iter $= 1 \ldots$ K **do**
  Sample θ_t^* and $\theta_{t+M}^* \in \Theta$, set $G = 0$, $\hat{\theta}_t = \theta_t^*$
  Initialize $\tilde{Y} = f(\theta_{t+M}^*; \tilde{X})$ {Soft Label Assignment}
  for $i = 1, \ldots, T$ **do**
    Compute $g_i = \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$
    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 $g_i = \nabla_\theta \ell(\hat{\theta}_{t+i}; \tilde{X}_i)$
    Compute $\frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial \tilde{X}_i}$ based on g_i and Equation (6)
    $\frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial x_{\pi_{i,b}}} \mathrel{+}= \frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial \tilde{X}_{i,b}}$ for all b
  end for
  Update x_j using $\frac{\partial \|\hat{\theta}_{t+T} - \theta_{t+M}^*\|_2^2}{\partial x_j}$ for all sampled j
end for

nel 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 preprocessing techiniques. Simiar to MTT, we apply the same DSA (Radford et al., 2015; Goodfellow et al., 2016; 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 soft label most effective on ImageNet-1K. 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 2. More details can be found in Appendix A.10. 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.

5.2. Empirical results

We compare TESLA against the random baseline and previous SOTA methods including DSA (Zhao & Bilen, 2021b), DM (Zhao & Bilen, 2021a), KIP (Nguyen et al., 2021), FrePo (Zhou et al., 2022) and the original MTT. The results are presented 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 only incurs 16% and 5.9% accuracy drop compared to training with the whole ImageNet-1K dataset while using only 0.83% and 4.2% of its training dataset size. This is an 18.2% accuracy improvement to the prior art (Zhou et al., 2022).

5.3. Training cost analysis

As discussed in Section 4.1, 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 2 (The numerical results can be found in Appendix A.11). The memory consumption of the original MTT increases linearly with the number of synthetic steps, while it remains constant for our method. This observation aligns with our theoretical analysis in Section 4.2. In principle, the constant memory reduction allows us to scale to arbitrarily large IPCs. We proceed to test the runtime overhead, alongside memory consumption across different dataset[5]. For this experiment, we fix the synthetic training step to 50 and batch size to 100. The results are summarized in Figure 3 (See Appendix A.11 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**.

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

[4] FrePo is only compared analytically due to lack of open-source code.

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

[6] MTT's runtime on ImageNet-1K is estimated since MTT is

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 (28.3±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***	

* Soft labels are used when table entries are marked with *.
[1] KIP's performance is measured with the dataset released by the author. Performances in quotas are from the original paper under different settings.
[2] FrePo uses a different model with much more parameters. We still mark FrePo result as bold if it outperforms other methods.
[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.10 for detailed reasons.

Table 2: Test accuracy 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**

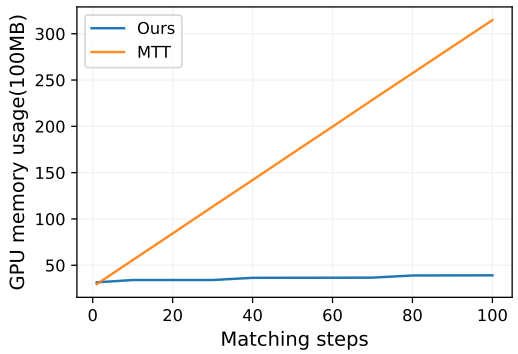

Figure 2: GPU memory usage comparison between MTT and TESLA. MTT scales linearly w.r.t. the number of matching steps, whereas our method uses constant memory. Results are measured on CIFAR100 with batch size 100 under varying matching steps.

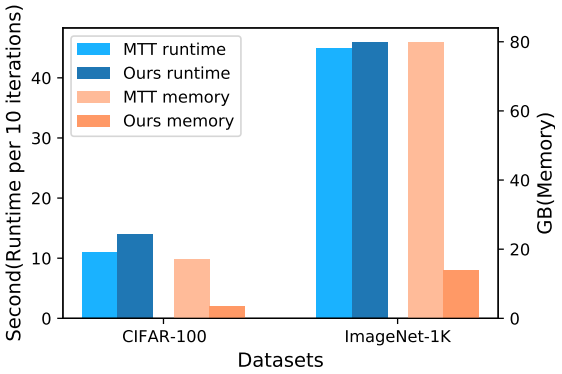

Figure 3: GPU memory and runtime comparison between MTT and TESLA on different datasets. Results are measured with a batch size of 100 and 50 matching steps.

5.4. 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 3. 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.

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.7). The results are plotted in Figure 4. While learning labels do slightly improve the performance, the margin of gain

OOM under our settings. See Appendix A.8

Table 3: Ablation study on testing accuracy (%) using hard vs soft label on ImageNet-1K 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

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 soft labels generated by the teacher model at the target step versus the last epoch. It's natural to think that a better-trained model will capture more training data statistics, thus generating better soft labels. However, we find that this doesn't work with trajectory matching. As shown in Figure 4, the algorithm fails to learn effectively with last epoch parameters.

We found that soft label assignment benefits synthetic image learning. To show this, we study the effect of soft label assignment technique alone, by fixing the synthetic images($\tilde{X}$). Then we measure the impact of soft labels($\tilde{Y}$) 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. The results are shown in Figure 5. 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.

5.5. 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 2, our method transfers well

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

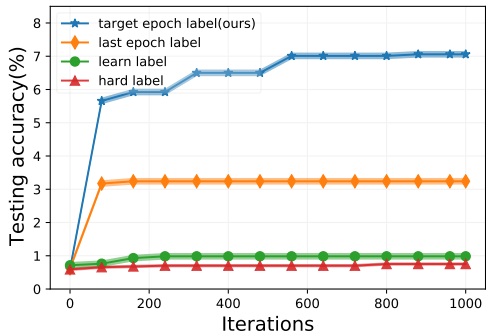

Figure 4: Ablation study on different label strategies. Y-axis shows the maximum accuracy achieved until that iteration.

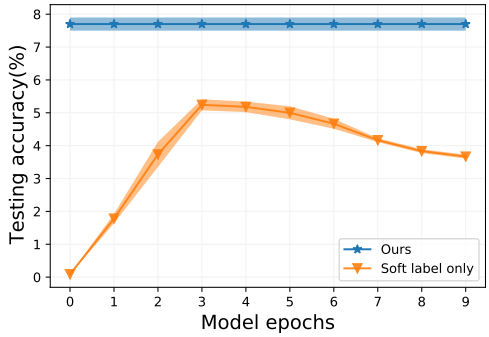

Figure 5: Ablation study on performance of SLA without updating synthetic images. Top flat line shows the performance of TESLA baseline.

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.

6. Conclusion

We present a novel method to reduce the previous SOTA: MTT's heavy memory cost from $\mathcal{O}(T)$ to $\mathcal{O}(1)$ with negligible run-time overhead. We also propose soft label assignment to guide the matching process of model training trajectories. By combining the two, we are able to scale dataset distillation to ImageNet-1K with IPC 10 and 50 for the first time, achieving SOTA performance. Moverover, our distilled dataset transfer well to across 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.

Limitation Despite having constant memory cost with respect to the number of unrolled steps, our method still requires storing some checkpoints of the teacher model, similar to the original MTT. Removing the requirement of a teacher model will be an interesting future research direction to pursue. Further, the study of dataset distillation is

in the very early stage so it remains important to close the performance gap to the full model.

Acknowledgements

The authors thank the support from NSF IIS-2008173, IIS-2048280 and Cisco research award.

References

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J.-Y. Dataset distillation by matching training trajectories. *arXiv preprint arXiv:2203.11932*, 2022.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. 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.

Cui, J., Wang, R., Si, S., and Hsieh, C.-J. Dc-bench: Dataset condensation benchmark. In *NeurIPS (Dataset and Benchmark)*, 2022.

Deng, Z. and Russakovsky, O. Remember the past: Distilling datasets into addressable memories for neural networks. *arXiv preprint arXiv:2206.02916*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT Press, 2016.

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

Kim, J.-H., Kim, J., Oh, S. J., Yun, S., Song, H., Jeong, J., Ha, J.-W., and Song, H. O. Dataset condensation via efficient synthetic-data parameterization. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (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.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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

Lee, S., Chun, S., Jung, S., Yun, S., and Yoon, S. Dataset condensation with contrastive signals. *arXiv preprint arXiv:2202.02916*, 2022.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2020.

Nguyen, T., Novak, R., Xiao, L., and Lee, J. Dataset distillation with infinitely wide convolutional networks. In *Advances in Neural Information Processing Systems*, 2021.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., and Bradski, G. Kornia: an open source differentiable computer vision library for pytorch. *workshop on applications of computer vision*, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., and You, Y. Cafe learning to condense dataset by aligning features. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022*, 2022.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representation*, 2021.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

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

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

Xiong, Y., Wang, R., Cheng, M., Yu, F., and Hsieh, C.-J. Feddm: Iterative distribution matching for communication-efficient federated learning, 2022.

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

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

Zhao, B., Mopuri, K. R., and Bilen, H. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2020a.

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33: 7559–7570, 2020b.

Zhou, Y., Pu, G., Ma, X., Li, X., and Wu, D. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

Zhou, Y., Nezhadarya, E., and Ba, J. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022.

A. Appendix

A.1. Bi-level optimization

Dataset distillation is originally done through bi-level optimization(Wang et al., 2018; Zhao et al., 2020a; Zhao & Bilen, 2021b; Cazenavette et al., 2022). Suppose we have a dataset $\mathcal{D} = \{X, Y\}$, the goal of dataset distillation is to get a subset $\mathcal{S} = \{\hat{X}, \hat{Y}\}$ where $|\mathcal{S}| \ll |\mathcal{D}|$ such that the following goals can be achieved $\mathbf{E}_{(x,y)\sim P_{\mathcal{D}}}\mathcal{L}_{\theta}(x, y) \simeq \mathbf{E}_{(\hat{x},\hat{y})\sim P_{\mathcal{S}}}\mathcal{L}_{\hat{\theta}}(\hat{x}, \hat{y})$. Here $\mathcal{L}$ is a loss function. θ and $\hat{\theta}$ are the parameters generated by using the original dataset and the distilled dataset. It can naturally be divided into a bi-level optimization process where the inner loop updates $\hat{\theta} = \underset{\hat{\theta}}{argmin}\mathcal{L}_{\hat{\theta}}(\hat{x}, \hat{y})$ and the outer loop computes $\mathcal{S}^* = \underset{\mathcal{S}}{argmin}\mathcal{M}_{\hat{\theta}}(\mathcal{S}, \mathcal{D})$. Here $\mathcal{M}$ is a surrogate function that's aimed to identify the difference between $\mathcal{S}$ and $\mathcal{D}$. For example, (Zhao et al., 2020a) tries to match the training gradients generated by using $\mathcal{S}$ and $\mathcal{D}$, in (Cazenavette et al., 2022), the training trajectories are used.

A.2. Complexity of other distillation methods

In the main text, we focus on the complexity analysis of FrePo and MTT. Here we discuss the complexity of other methods. (Wang et al., 2018; Zhao et al., 2020a; Zhao & Bilen, 2021b) also uses bi-level optimization. However, compared to MTT and FrePo, they achieve lower performances and cannot scale to large datasets. (Zhao & Bilen, 2021a) takes another approach by matching the distributions of synthetic dataset with real dataset. The method is fast due to its single level of optimization. However, it suffers from great accuracy loss when applying to large datasets (Table 1).

A.3. Datasets

On 3 datasets including CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-1K (Russakovsky et al., 2015). CIFAR-10/100 includes 50,000 training and 10,000 testing images in 32×32 resolution from 10 and 100 classes respectively. ImageNet-1K is a standard large-scale dataset consists of 1,000 classes with 1,281,167 training and 50,000 testing images. We resize ImageNet-1K images to 64×64 resolutions following (Zhou et al., 2022). We do not evaluate our methods on toy datasets such as MNIST (LeCun et al., 1998) or Fashion-MNIST(Xiao et al., 2017) as the performances of different methods on MNIST-variants are too close and we target large datasets.

A.4. Data preprocessing

ZCA whitening is first used in KIP. We reimplement the author's custom ZCA in order to evaluate its performance through the publicly released dataset. However, for the ZCA preprocessing in our work, we follow the same Kornia (Riba et al., 2019) ZCA as the original MTT work. We also follow the same ZCA settings as MTT. On ImageNet-1K, we don't apply ZCA at any of the IPCs.

A.5. Models

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. The original KIP uses a larger model with a width of 1024 for evaluation compared to 128 used by other methods and it has one more conv layer than the model used by other methods. FrePo uses another different model that doubles the number of filters when the feature map size is halved. Also batch normalization is used by FrePo instead of instance normalization. A 3-layer ConvNet is used for CIFAR-10/100 and a 4-layer ConvNet is used for ImageNet-1K.

A.6. Hardwares

All of our experiments are run using one NVIDIA A6000 GPU with 49GB of memory. When measuring the memory consumption used by the original MTT, if it doesn't fit into one GPU, we use two NVIDIA A6000 GPUs just for measuring purpose.

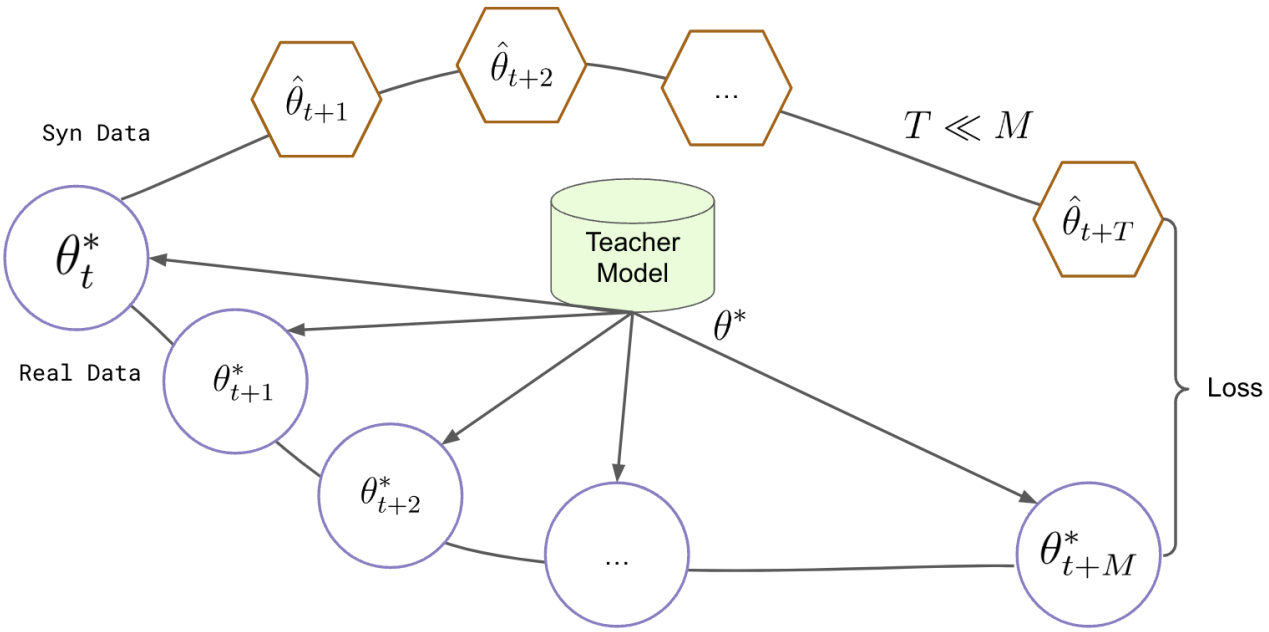

Figure 6: Illustration of matching training trajectories(**MTT**).

A.7. Label learning

In our experiment with label learning, we consider the following 2 implementation options. Firstly we try to initialize labels using one-hot float vectors. Then we directly feed the one-hot vector to the surrogate model and use back-propagation to update the labels. In this scenario, we observe that the classes probabilities will sum up to an arbitrary number after being updated. This causes the cross-entropy loss function used by the surrogate model to be unstable. We observe poor performance by following this implementation.

The second implementation is used in our experiment in Figure 4. Instead of initializing the labels with class probabilities that sum up to 1, we initialize it with logits. In our case, we set the class index bit to 10 and the rest to 0. Then before feeding it into the loss function, we apply a softmax operation to convert logits to class probabilities. The algorithm now is able to learn stably. And we noticed the class weight shift from the class index bit to other classes. However, as mentioned in the main text that it's still not able to learn effectively because of the large label space.

We also consider using different loss functions other than cross entropy. For example, FrePo uses MSE loss in all its experiments. However, we also notice poor performances. As mentioned in (Zhou et al., 2022), cross-entropy loss usually outperforms MSE loss which aligns with our observations.

A.8. Training cost analysis

We show the runtime and memory analytically in the main text in Section 4.2 and empirically in Figure 2 and Figure 3. Here we include the numerical results in Appendix A.11 and Appendix A.11. Note that on ImageNet-1K, the memory usage of MTT is acquired by running it on 2 NVIDIA RTX A6000 GPUs, each of which has 49GB of memory. For MTT's runtime on ImageNet-1K, it's estimated by observing linearity between batch size and runtime. It can be easily seen that our memory cost is constant with respect to the number of synthetic steps. From Appendix A.11 we can see that, our algorithm uses only one fifth of MTT's memory with only 27% more time[8]. Thus our algorithm is able to easily scale to larger datasets with large IPCs.

[8]Although our method only computes the first order gradient twice, it's fast and memory efficient compared to the meta gradient computation w.r.t synthetic images which is only computed once.

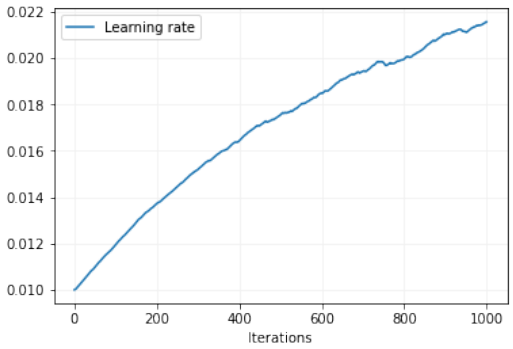

Figure 7: Learning curve for student model learning rate

Table 4: Peak memory usage comparison on CIFAR-100 between MTT and ours using batch size 100. The units are in MB. *Matching step* means how many gradient descent steps to run before performing model training trajectory matching.

	matching step										
	1	10	20	30	40	50	60	70	80	90	100
MTT	2961	5555	8431	11329	14197	17093	19965	22867	25737	28607	31469
Ours	3155	3405	3405	3405	3643	3647	3653	3661	3895	3905	3913

The results are measured using one NVIDIA A6000 GPU.

A.9. Augmentation

We use the same DSA augmentation as other works (Zhao & Bilen, 2021b; Cazenavette et al., 2022; Zhou et al., 2022). Unlike FrePo which only applies augmentation at evaluation stage, we apply data augmentation at both training and evaluation stage. Similar to previous work such as MTT, we observe better performances. We also try different augmentations such as Autoaugment (Cubuk et al., 2018) and Randaugment (Cubuk et al., 2020) besides DSA during evaluation, however, we usually observe downgraded performs.

A.10. Competitors

In Table 1, we ar not able to get the performance of some methods. We list the reasons here. For DSA, we are not able to get the performances on ImageNet-1K because of memory constraints. For DM, we are only able to get the performances for IPC 1 and 2. We have to perform early stopping when the testing accuracy plateaus because it takes 5 minutes to perform each iteration and the original number of iterations is set to 20,000 by the author. DM gets OOM error when setting IPC to 10 on ImageNet-1K. For KIP, the work is not open-sourced. The author only released distilled datasets on CIFAR-10 with IPC 1,10, 50 and CIFAR-100 with IPC 1 and 10. For FrePo, as it's a recent work, there is no open-sourced code or publicly released dataset yet. Therefore, we report the numbers from its original work. Also as reported by the authors, FrePo is not able to scale to TinyImageNet with IPC 50 and ImageNet-1K with IPC 10.

A.11. Learning learning rate

We also notice that learning the student model learning rate is also extremely helpful in generating the synthetic datasets. We show an example figure of learning rate change for ImageNet-1K IPC 1. It can be seen from Figure 7 that as the training progresses, the learning rate will keep increasing. Under our settings, we set its initial values to 0.01. After 1000 iterations, the learning rate goes up to 0.02. We notice that this not only stabilizes the training process, but also improves the testing accuracy when evaluating the distilled datasets.

Table 5: Memory and runtime for MTT and ours. The results are measured using batch size 100 and 50 matching steps. The memory is the peak memory used and the runtime is measured for 10 iterations(one iteration includes 50 matching steps). Memory diff is calculated by dividing MTT memory by our memory. Time diff is calculated by dividing our time by MTT time.

	MTT memory	Our memory	memory diff	MTT time	Our time	time diff
CIFAR-100	17.1±0.1GB	3.6±0.1GB	4.75X	11.5±0.5sec	14.5±0.5sec	1.25X
ImageNet-1K	79.9±0.1GB	13.9±0.1GB	5.75X	45.0±0.4sec	46.0±0.8sec	1.02X

The results are measured using one NVIDIA A6000 GPU.

A.12. Hyperparameters

To facilitate the reproduce of our work, we list all the hyperparameters used in our experiments in this table. We use slightly different hyperparameters on CIFAR-10/100 as listed in Appendix A.12 than the original MTT while keeping everything else the same such as model architectures and augmentations. Theoretically, MTT should have the same performances as us on CIFAR-10/100 with smaller IPCs where batch is not needed.

Table 6: Hyperparameters used to get the distilled dataset.

Dataset	Model	IPC	Matching Steps	Teacher Epochs	Max Start Epoch	Batch Size	ZCA
CIFAR-10	ConvNetD3	1	50	2	3	-	Y
		10	30	2	20	-	Y
		50	30	3	40	-	N
CIFAR-100	ConvNetD3	1	20	3	20	-	Y
		10	15	3	30	-	N
		50	50	2	40	100	Y
ImageNet-1K	ConvNetD4	1	10	3	6	100	N
		2	15	3	10	100	N
		10	20	3	10	500	N
		50	100	3	25	500	N

A.13. Example distilled image

We show example distilled images for the 3 dataset used in this work for easier references. For CIFAR-10, we show 10 images per class, for CIFAR-100, we show 1 image per class and for ImageNet-1K, we show 1 image per class for the 1K classes. (more pages after this paragraph)

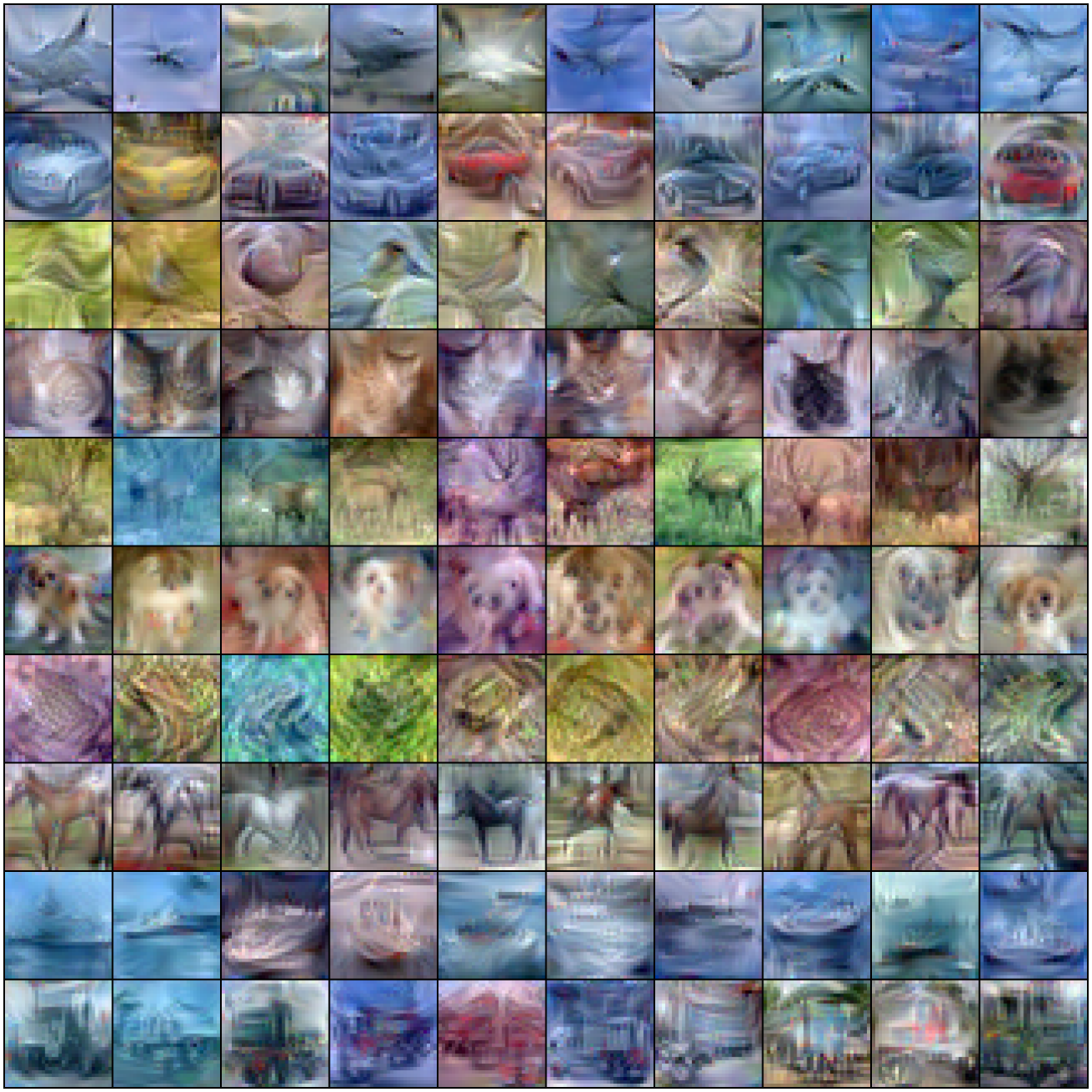

Figure 8: CIFAR10 IPC 10

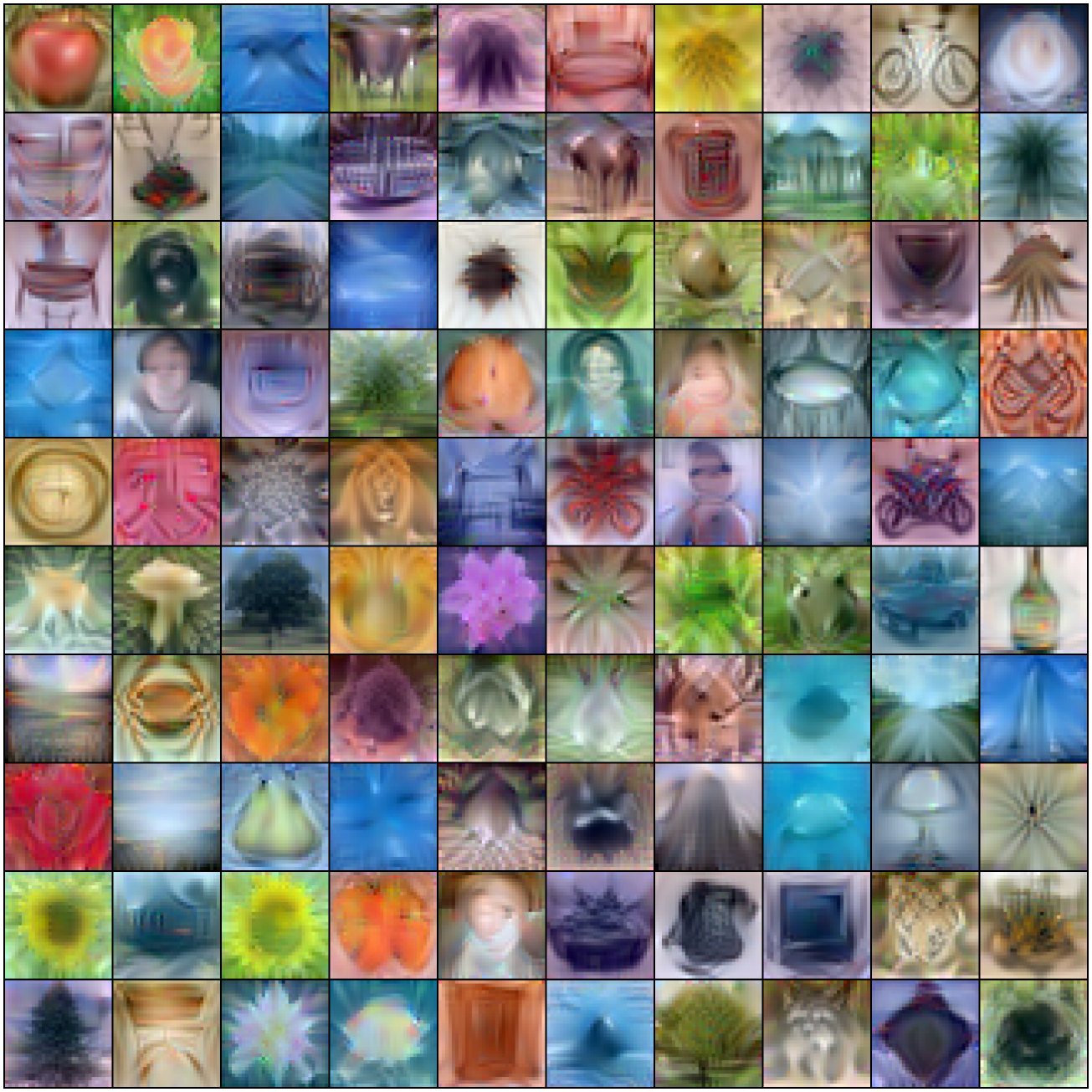

Figure 9: CIFAR100 IPC 1

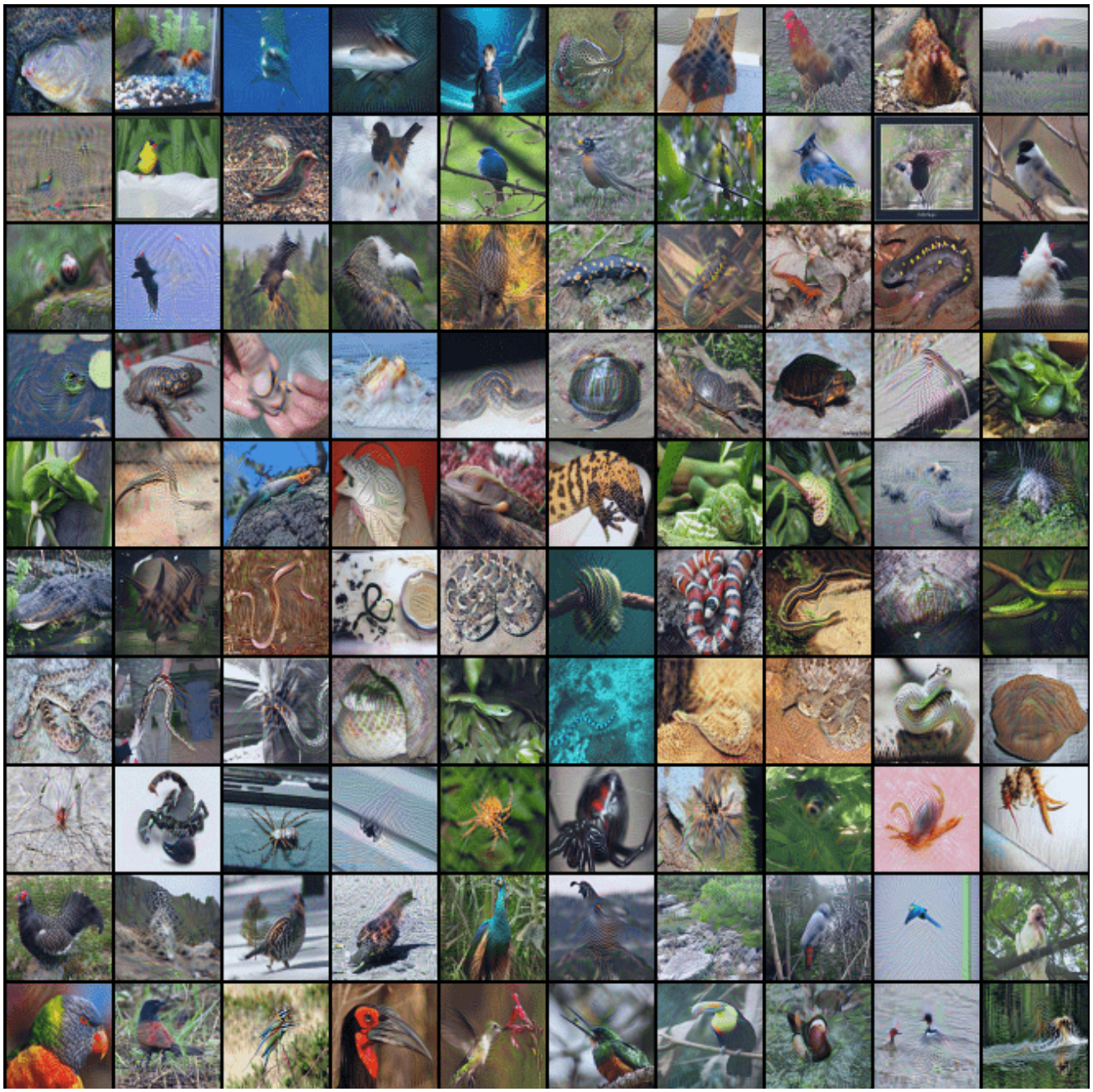

Figure 10: ImageNet-1K IPC 1, Class ID[0:99]

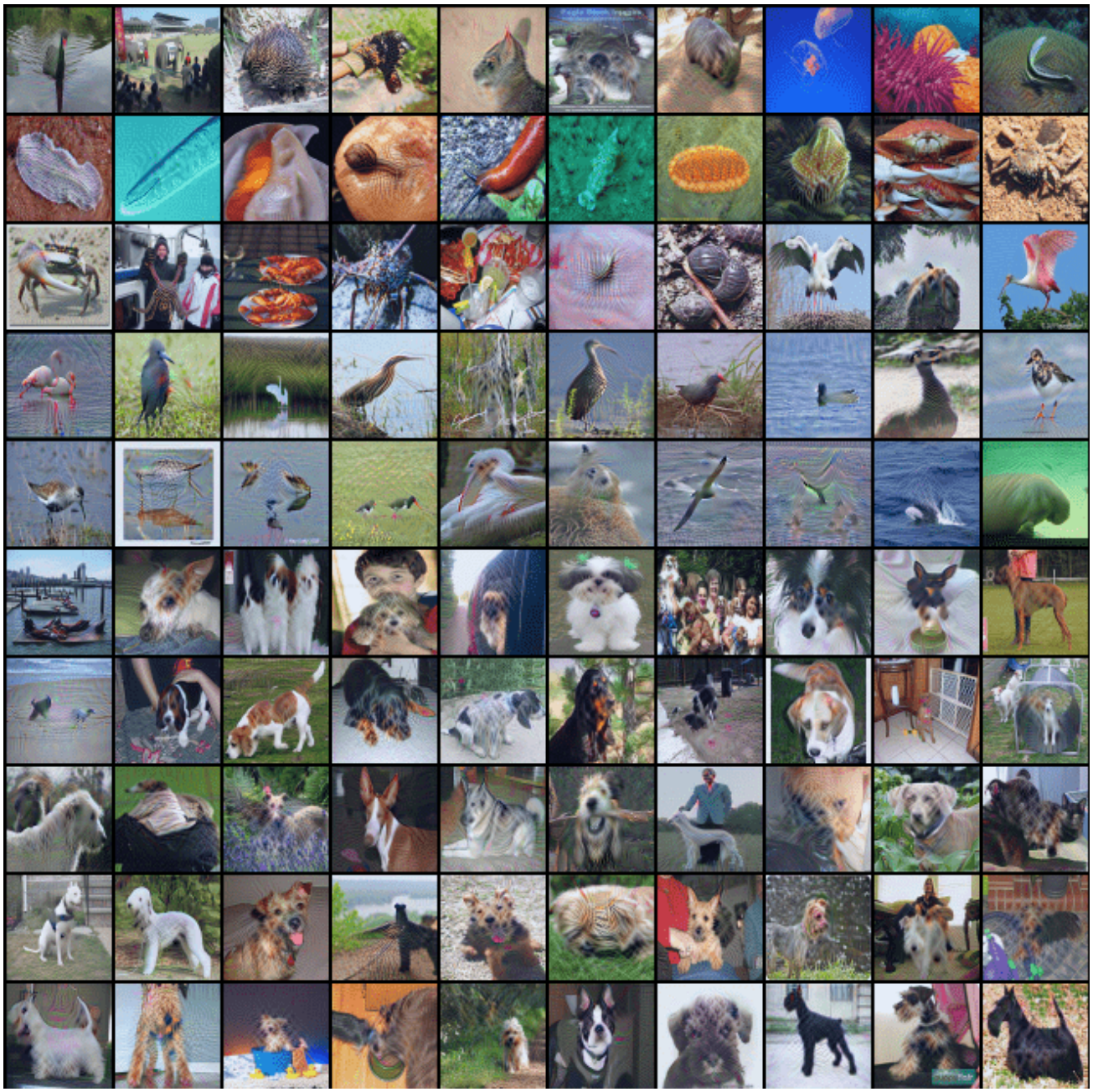

Figure 11: ImageNet-1K IPC 1, Class ID[100:199]

Figure 12: ImageNet-1K IPC 1, Class ID[200:299]

Figure 13: ImageNet-1K IPC 1, Class ID[300:399]

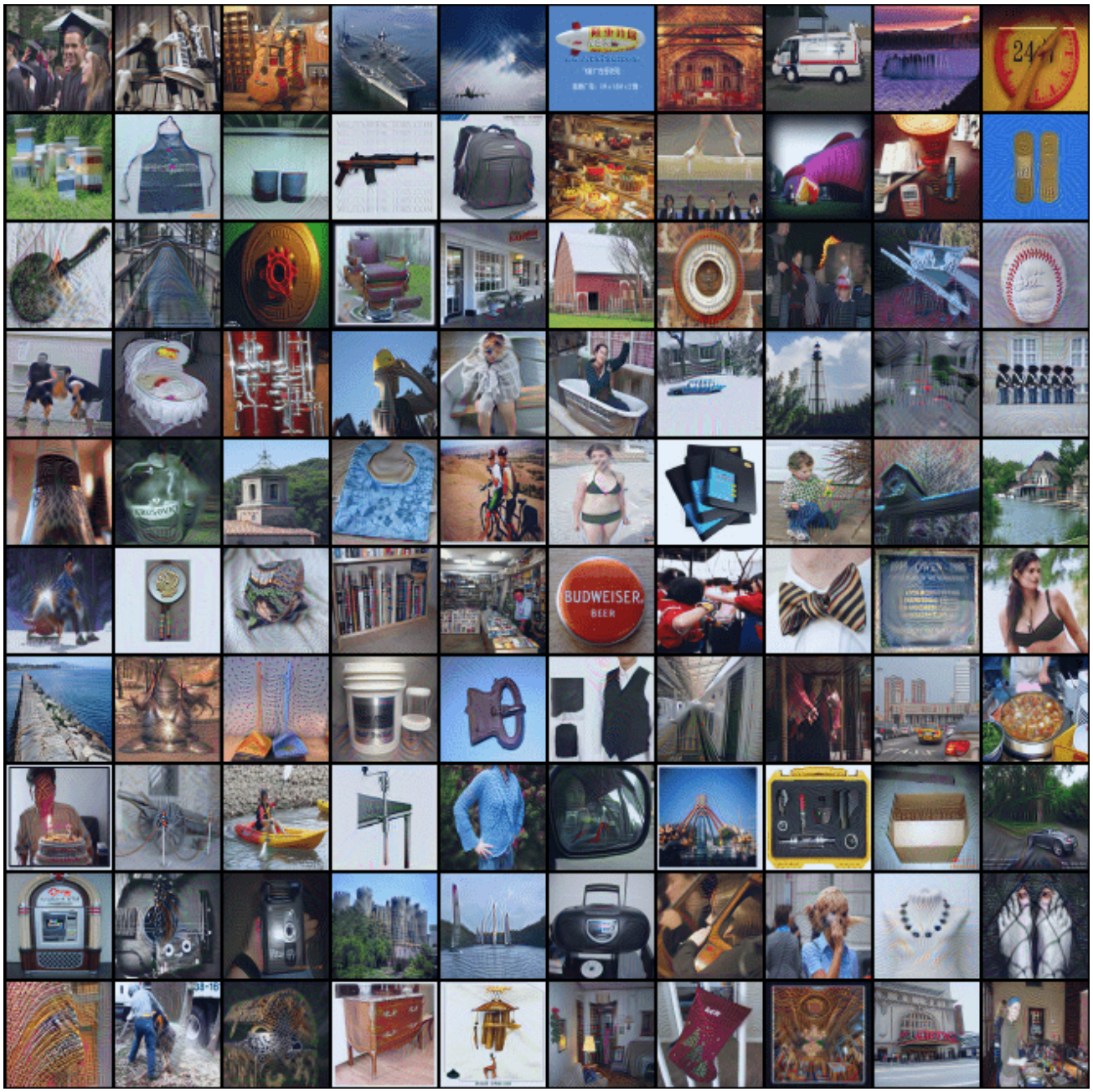

Figure 14: ImageNet-1K IPC 1, Class ID[400:499]

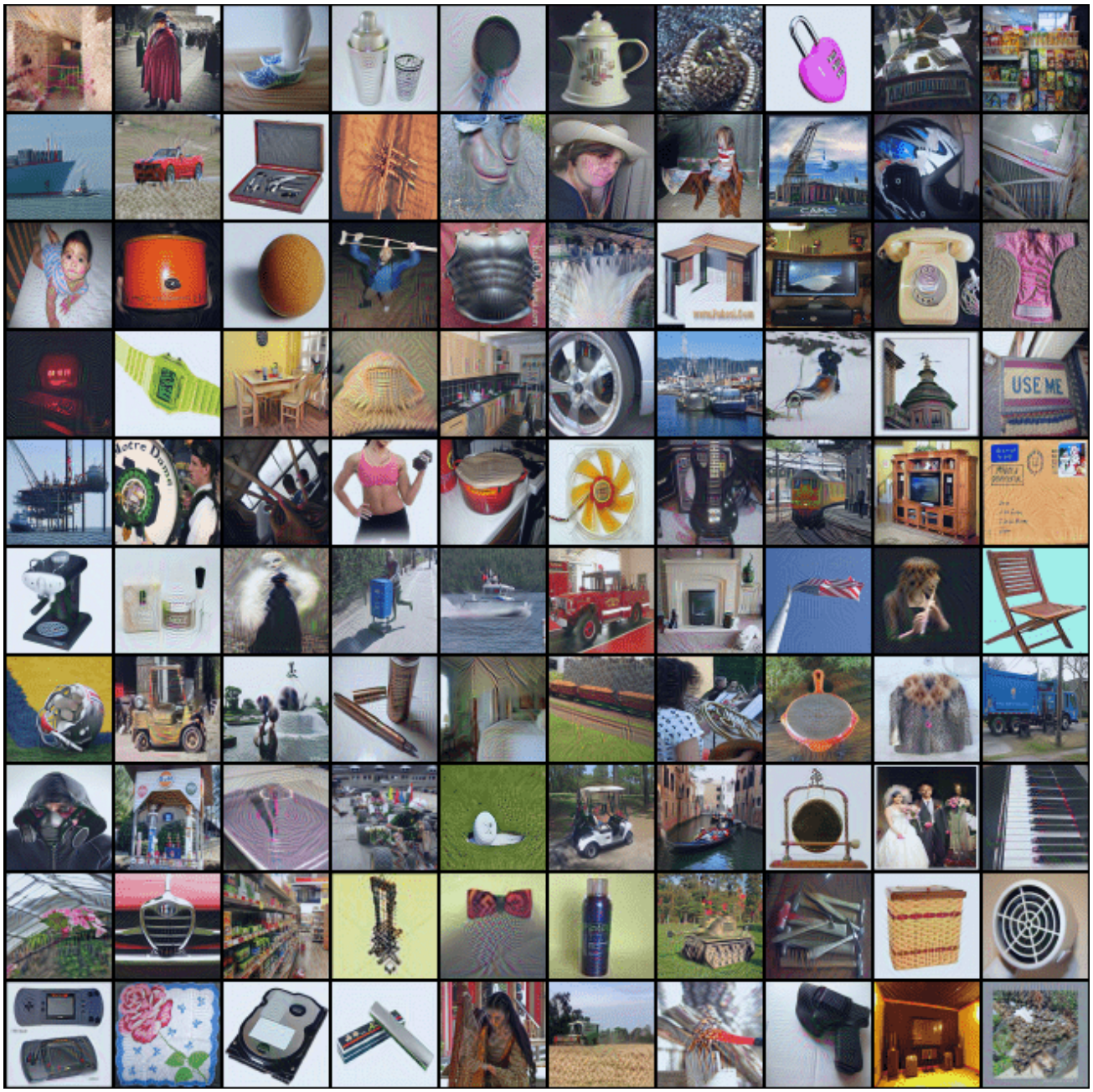

Figure 15: ImageNet-1K IPC 1, Class ID[500:599]

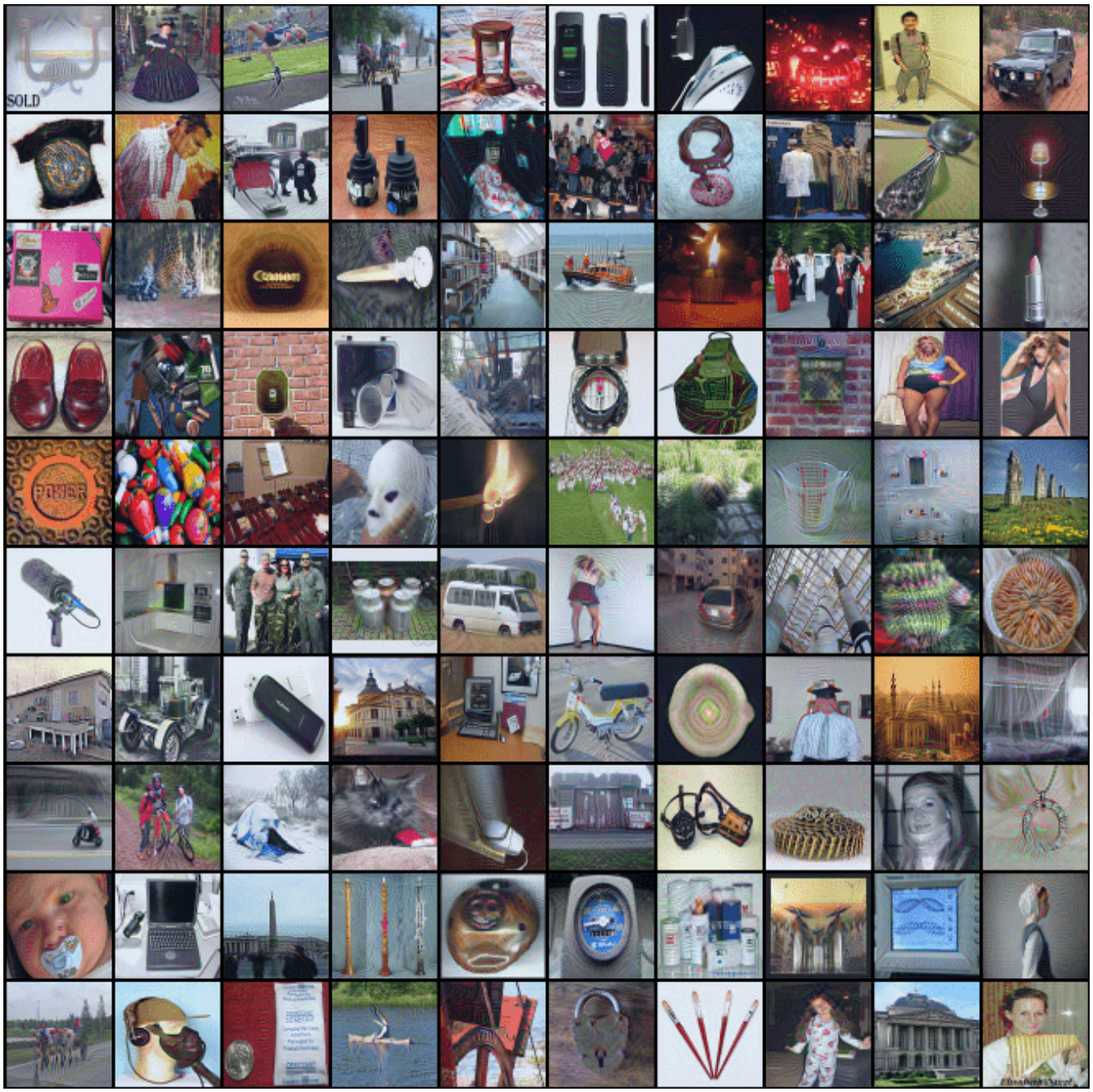

Figure 16: ImageNet-1K IPC 1, Class ID[600:699]

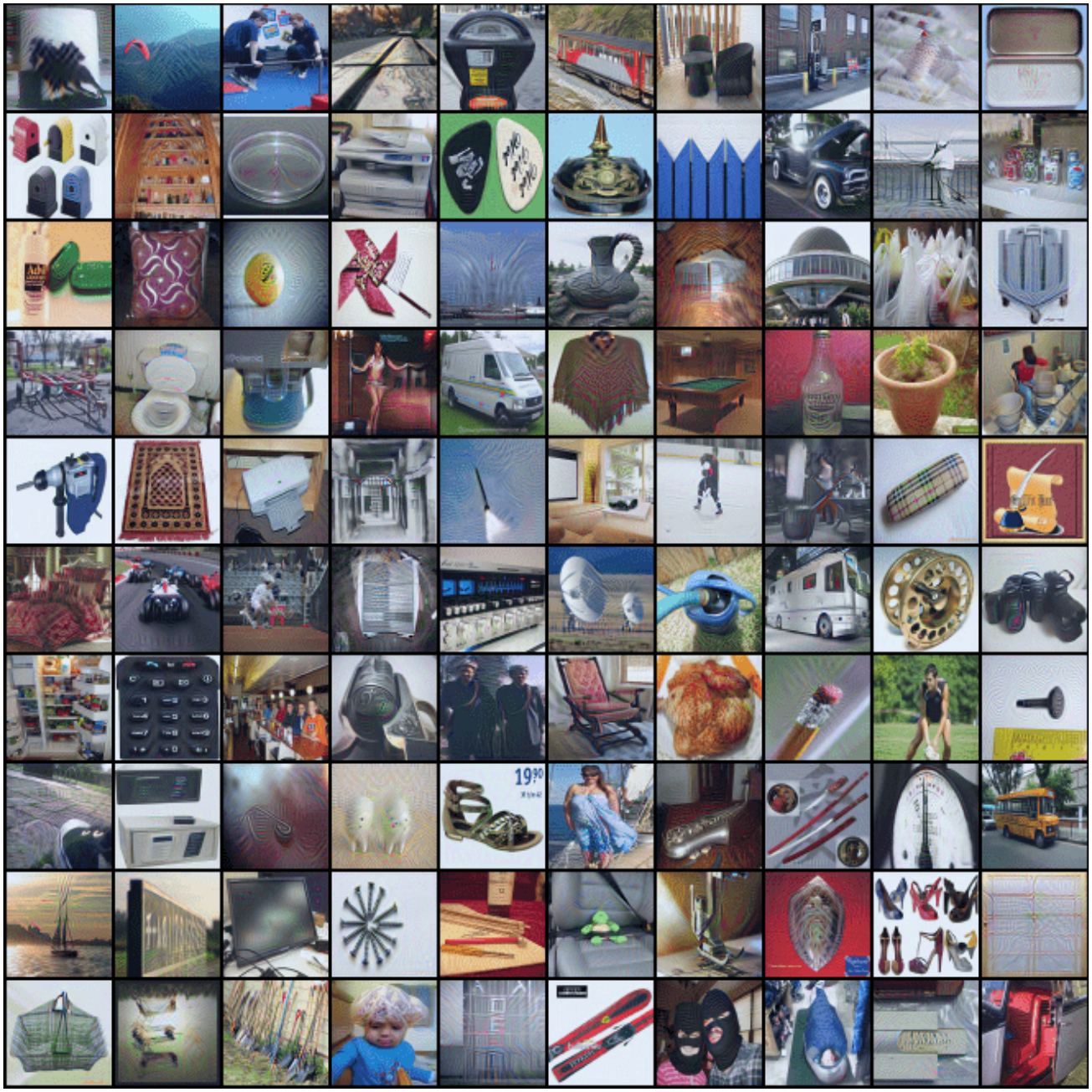

Figure 17: ImageNet-1K IPC 1, Class ID[700:799]

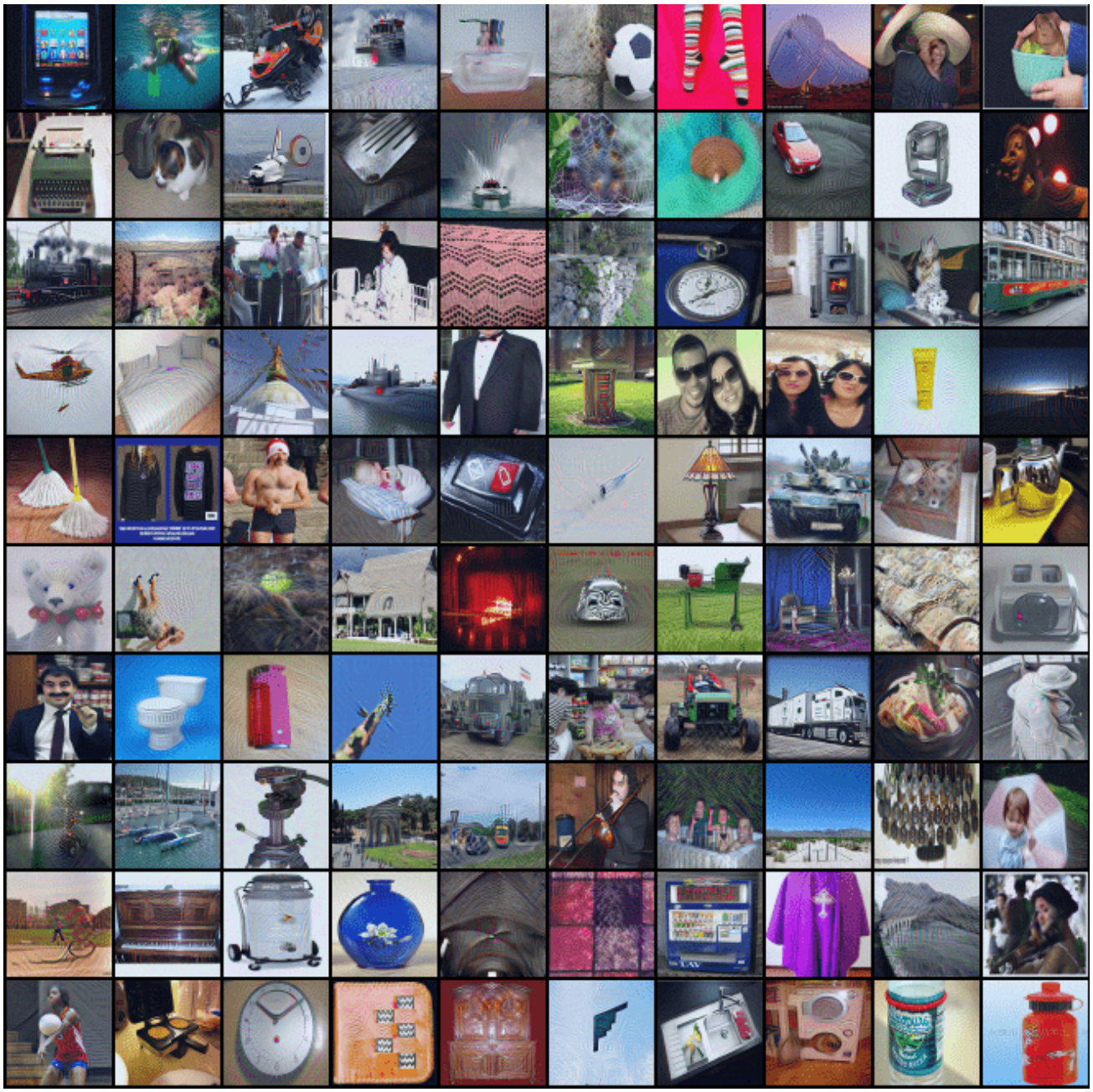

Figure 18: ImageNet-1K IPC 1, Class ID[800:899]

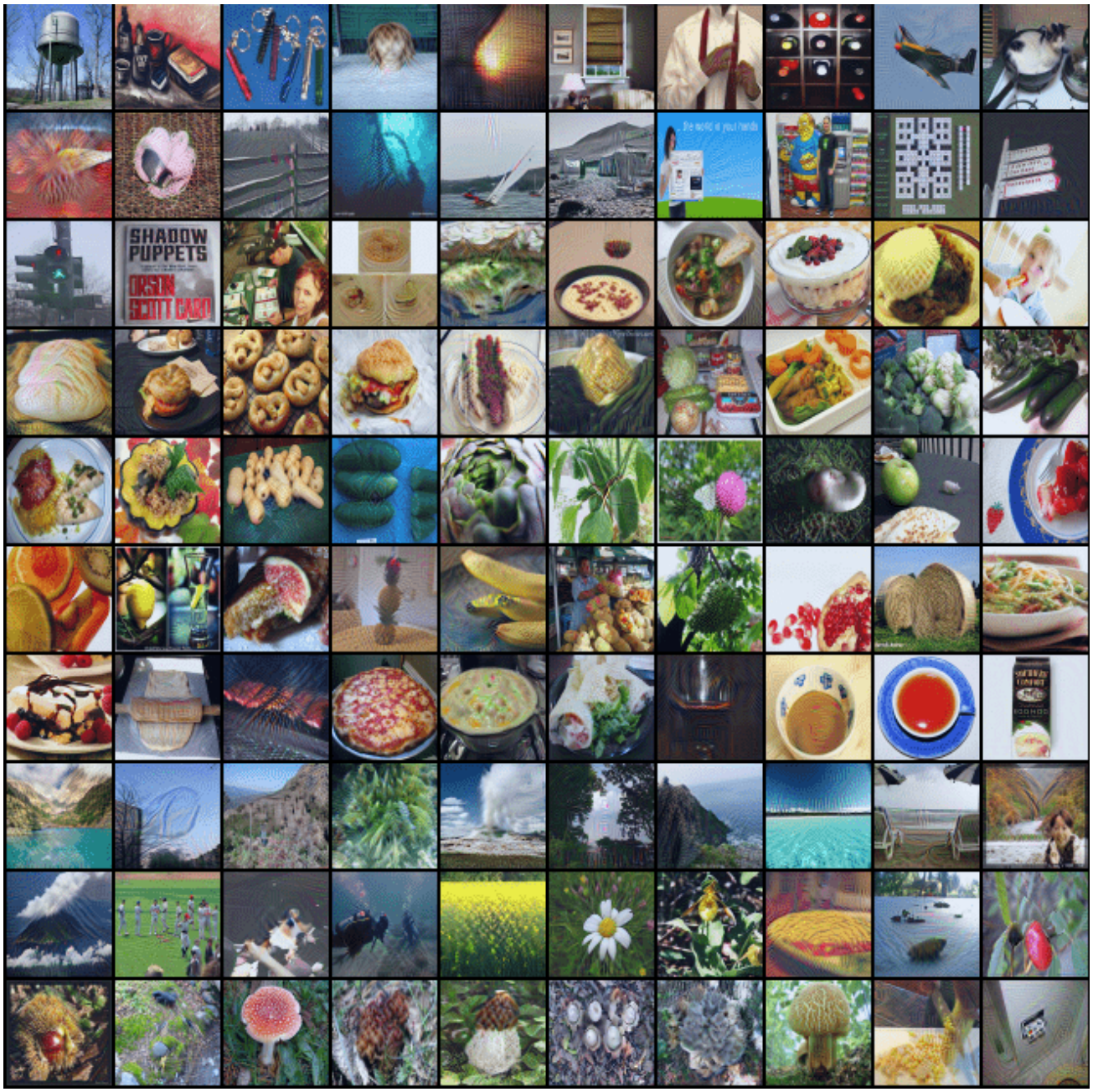

Figure 19: ImageNet-1K IPC 1, Class ID[900:999]