PATCH-LEVEL CODEBOOK PARAMETERIZATION FOR DATASET DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dataset distillation aims to synthesize a compact dataset that retains the essential knowledge of a large-scale dataset, enabling efficient model training while significantly reducing storage and computational costs. Recent parameterization methods enhance storage efficiency through compact or shared representations, but recurring local structures may still be redundantly encoded across synthetic images, limiting storage efficiency. In this paper, we propose to parameterize a synthetic dataset using patch-level learnable codebooks and patch index maps, such that each synthetic image is generated by assembling learned representative patches according to the index map. By explicitly sharing patch-level representations across synthetic images, our method reduces repeated encoding of recurring local structures and improves the use of limited storage budgets. Furthermore, we employ multiple codebooks at different stages and combine their patches additively to further improve representation power. Extensive experiments on standard benchmarks demonstrate that our method achieves state-of-the-art performance across various dataset distillation methods.

1 INTRODUCTION

Deep neural networks have demonstrated remarkable performance in various computer vision tasks Dosovitskiy et al. (2021); Liu et al. (2021) by learning from massive amounts of data Deng et al. (2009); Lin et al. (2014). However, training deep neural networks on large-scale datasets requires high computing costs and large storage space. Dataset distillation Wang et al. (2018) effectively tackles these computational and storage overheads by synthesizing a compact dataset that contains the core knowledge of the original dataset. This is typically achieved by iteratively updating the raw pixels of the synthetic image so that models trained on it exhibit behavior similar to those trained on the full dataset, using objectives such as gradient matching (GM) Zhao et al. (2021), distribution matching (DM) Zhao & Bilen (2023), or trajectory matching (TM) Cazenavette et al. (2022).

To further improve storage efficiency, recent works parameterize the synthetic dataset, encoding each synthetic image into a compact form that reduces its redundancy. Rather than storing raw pixels, these methods represent each image through compact parameters from which the full image can be decoded. The storage cost is therefore determined by the size of the compact parameters rather than that of the decoded image. This separation allows more synthetic images to be used for model training under the same storage budget. A pioneering work, IDC Kim et al. (2022), stores each synthetic image at a reduced spatial resolution and upsamples it during model training, thereby increasing the number of available synthetic samples without additional storage. Subsequent methods adopt other compact forms, such as frequency coefficients Shin et al. (2023); Yang et al. (2024), generative latent representations Cazenavette et al. (2023); Zhong et al. (2025), implicit neural fields Shin et al. (2025), compressed latent codes Bao et al. (2025), or combinations of shared components Liu et al. (2022); Deng et al. (2022); Wei et al. (2023).

However, natural images often contain recurring local structures, such as edges, smooth regions, textures, and color patterns. As illustrated in Figure 1, similar local structures recur across spatial locations within an image as well as across images, both within and across classes. This recurrence suggests that a large portion of an image dataset can be represented by reusing a relatively small collection of common local patterns. Although existing methods exploit compact or shared repre-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sentations, recurring local structures are not explicitly represented as reusable learned components across synthetic images. Consequently, similar local information may still be encoded repeatedly across different synthetic images.

Motivated by this observation, we propose a patch-level parameterization method for dataset distillation that stores recurring local structures as reusable components to synthesize images. Our method explicitly captures such recurrence through shared local patches, avoiding repeated encoding of similar local information. Inspired by vector quantization (VQ) Van Den Oord et al. (2017); Esser et al. (2021), we repurpose the VQ codebook as a compact set of learnable 2D image patches that parameterizes the distilled dataset. Each synthetic image is constructed by selecting and tiling patches from the codebook according to a patch index map. Since the map stores only patch indices rather than raw pixel values, it requires significantly less memory than storing full pixel representations. To further improve the representational capacity, we propose a multi-codebook framework, where the outputs of several codebooks are jointly combined to synthesize images. This formulation increases the number of representable patch combinations while keeping the storage cost low, offering a more favorable trade-off between representational capacity and storage than enlarging a single codebook. Experimental results show that, under the same storage budget, the proposed method enables more diverse synthetic representations, resulting in superior dataset distillation performance compared with existing methods.

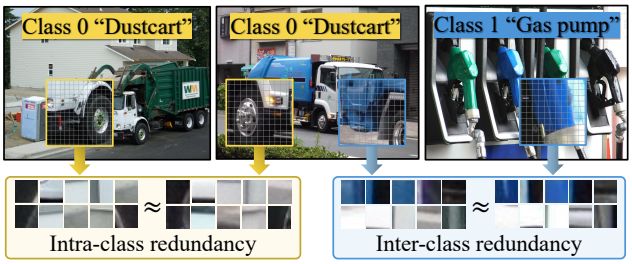


Figure 1: Redundancy among local image patches. Recurring local patterns are often shared both within the same class and across different classes.

Our contributions are summarized as follows:

- We propose a novel parameterization framework for distilled image datasets based on globally shared learnable codebooks and discrete per-image index maps, which explicitly reuse recurring local patterns across synthetic images and reduce redundant local encoding.
- We further extend the framework with multi-stage codebooks, which increase the number of possible patch combinations under a comparable storage budget.
- Extensive experiments on standard benchmarks demonstrate that our method consistently achieves state-of-the-art performance across multiple dataset distillation objectives, including GM, DM, TM, and SRe^{2}L.

2 Related Work

2.1 Dataset Distillation

Dataset distillation aims to synthesize a compact dataset such that models trained on it achieve performance comparable to those trained on the full dataset. Pioneering work Wang et al. (2018) directly optimizes synthetic images as learnable parameters through backpropagation. Due to the high computational cost of the bi-level formulation, subsequent works have developed various surrogate objectives. Methods such as Zhao et al. (2021) optimize the synthetic images such that the gradients computed on them closely match those computed on the original data, ensuring that a model trained on the synthetic images follows a similar parameter update direction to one trained on the original data. Subsequently, several works Cazenavette et al. (2022); Du et al. (2023); Guo et al. (2024); Liu et al. (2024) extend this idea beyond gradient directions, aligning the full parameter trajectories induced by training on the synthetic and original data. Rather than matching gradients or trajectories, another family of methods Zhao & Bilen (2023); Wang et al. (2022); Zhao et al. (2023); Zhang et al. (2024b;a); Deng et al. (2024) optimize the synthetic images by aligning the feature distributions of the synthetic and original data. More recently, decoupled approaches Yin et al. (2023); Sun et al. (2024); Tran et al. (2025); Cui et al. (2026); Li et al. (2026) scale dataset distillation to larger datasets by recovering synthetic images from a pretrained teacher with soft-label supervision.

Dataset distillation has also been extended to data modalities other than images, including text Li & Li (2021); Maekawa et al. (2023; 2024), point clouds Zhang et al. (2024c); Bo & Wang (2025); Yim et al. (2025); Kim et al. (2026), and graphs Jin et al. (2022b;a), demonstrating its broad applicability across domains.

2.2 Parameterization for Dataset Distillation

Early dataset distillation methods directly optimize synthetic images at the pixel level. However, this representation is inherently inefficient because the storage cost scales linearly with image resolution and the number of synthetic images. Consequently, recent studies have explored more compact parameterizations of synthetic datasets to encode richer information under the same storage budget. IDC Kim et al. (2022) introduces a method that stores synthetic images at reduced resolution and reconstructs full-resolution images through differentiable upsampling, thereby increasing the number of trainable synthetic samples without additional storage overhead. Several methods decompose the synthetic dataset into a set of shared components and combination coefficients. HaBa Liu et al. (2022) uses shared base images passed through a small network to generate synthetic images. RTP Deng et al. (2022) recovers synthetic images as a weighted sum of shared bases with learned weights. SPEED Wei et al. (2023) applies this idea in feature space, where shared components are combined and decoded into images through a lightweight decoder. NSD Yang et al. (2024) factorizes the dataset into spectral components combined with transformation matrices. Other methods represent each synthetic image in an alternative domain to reduce the storage cost required per image. PoDD Shul et al. (2025) reduces storage by generating synthetic samples as overlapping image-sized crops from a single learnable large image. FreD Shin et al. (2023) operates in the frequency domain, storing only the most informative frequency components of each image. RUO Bao et al. (2025) represents each image as an optimizable latent code decoded by a lightweight network and jointly optimizes data compression and the dataset distillation objective within a unified framework. DDiF Shin et al. (2025) represents each synthetic image as a compact implicit neural field. Overall, existing methods parameterize synthetic images using compact or shared representations, but do not explicitly decompose synthetic images into reusable local image-space components. Our method instead represents each image with globally shared learnable patches and index maps, so recurring local structures reuse the same stored parameters.

3 Method

3.1 Preliminary

Dataset Distillation. Dataset distillation aims to synthesize a compact dataset $\boldsymbol{\mathcal{S}} = \{\tilde{\mathbf{X}}_i\}_{i=1}^{|\boldsymbol{\mathcal{S}}|}$ from a large-scale original dataset $\boldsymbol{\mathcal{T}} = \{\mathbf{X}_i\}_{i=1}^{|\boldsymbol{\mathcal{T}}|}$, where $|\boldsymbol{\mathcal{S}}| \ll |\boldsymbol{\mathcal{T}}|$. The goal is to find the optimal synthetic dataset $\boldsymbol{\mathcal{S}}^*$ so that a model trained on $\boldsymbol{\mathcal{S}}^*$ achieves performance comparable to one trained on the original dataset. This goal can be formulated as the following bi-level optimization problem:

$$\boldsymbol{\mathcal{S}}^* = \arg\min_{\boldsymbol{\mathcal{S}}} \mathcal{L}(\boldsymbol{\mathcal{T}}, \boldsymbol{\theta}_{\boldsymbol{\mathcal{S}}}),\ \boldsymbol{\theta}_{\boldsymbol{\mathcal{S}}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\mathcal{S}}, \boldsymbol{\theta}), \tag{1}$$

where $\mathcal{L}(\cdot, \boldsymbol{\theta})$ denotes the loss of the model with parameters $\boldsymbol{\theta}$, and $\boldsymbol{\theta}_{\boldsymbol{\mathcal{S}}}$ denotes the network parameters trained on $\boldsymbol{\mathcal{S}}$. Since directly solving this bi-level optimization is computationally expensive, various surrogate objectives have been proposed to approximate this problem as follows:

$$\boldsymbol{\mathcal{S}}^* = \arg\min_{\boldsymbol{\mathcal{S}}}\ D(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{S}}), \tag{2}$$

where $D(\cdot, \cdot)$ denotes a distillation loss for matching gradient Zhao et al. (2021), feature distribution Zhao & Bilen (2023), or training trajectory Cazenavette et al. (2022).

Parameterization for Dataset Distillation. To further improve storage efficiency, recent works propose to represent the synthetic dataset $\boldsymbol{\mathcal{S}}$ not as raw pixels, but as a set of compact representations $\boldsymbol{\mathcal{Z}}$ decoded by a set of parameterized functions $\boldsymbol{\mathcal{G}}$. The optimization objective then becomes as follows:

$$\begin{aligned} \{\boldsymbol{\mathcal{G}}^\star, \boldsymbol{\mathcal{Z}}^\star\} &= \arg\min_{\{\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{Z}}\}}\ D(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{S}}), \\ \boldsymbol{\mathcal{S}} &= \{g_{\boldsymbol{\phi}}(\mathbf{Z}) \mid g_{\boldsymbol{\phi}} \in \boldsymbol{\mathcal{G}}, \mathbf{Z} \in \boldsymbol{\mathcal{Z}}\}, \end{aligned} \tag{3}$$

where $\boldsymbol{\phi}$ denotes the parameters of each decoder $g_{\boldsymbol{\phi}}$.

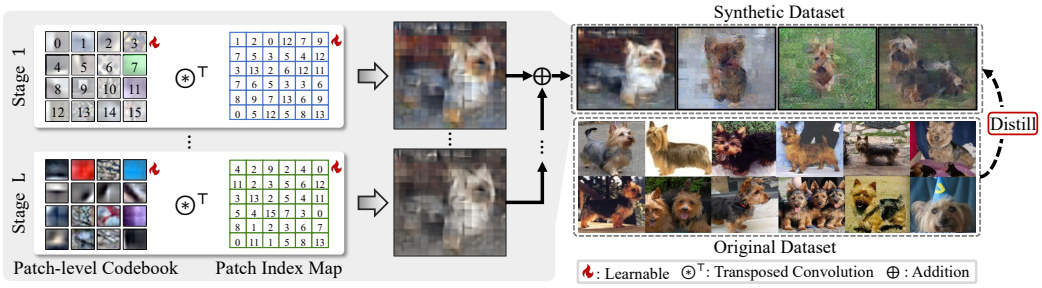


Figure 2: Overview of the proposed patch-level parameterization framework. Each synthetic image is constructed by selecting and assembling reusable patches from shared codebooks at multiple stages according to the patch index maps.

3.2 Patch-Level Codebook Parameterization

We propose to parameterize the synthetic dataset through patch-level learnable codebooks, where each synthetic image is generated by assembling patches selected from the shared codebooks. As illustrated in Figure 2, a patch-level codebook stores recurring local patterns as reusable patches, and each synthetic image is described by a learnable patch index map specifying which codebook patch to place at each spatial position. We store each patch in the form of Discrete Cosine Transform (DCT) coefficients rather than raw pixels, as visual patterns shared across datasets are typically dominated by smooth, low-frequency components, which are also naturally prioritized by gradient-based optimization Zhi-Qin et al. (2020). This representation provides an optimization bias toward low-frequency components.

Let $\boldsymbol{\mathcal{P}} = \{\mathbf{P}_k\}_{k=1}^{|\boldsymbol{\mathcal{P}}|}$ be a patch-level learnable codebook, where each $\mathbf{P}_k \in \mathbb{R}^{C\times H_p\times W_p}$ is a codebook patch of the spatial size $H_p \times W_p$ with C channels. To construct a synthetic image using $\boldsymbol{\mathcal{P}}$, we determine which codebook patch is assigned to each spatial position. Specifically, we consider $H_g \times W_g$ grids of spatial locations in a synthetic image $\tilde{\mathbf{X}} \in \mathbb{R}^{C\times H\times W}$ by sliding the $H_p \times W_p$ window with stride R, where $H_g = \lfloor \frac{H-H_p}{R} \rfloor + 1$ and $W_g = \lfloor \frac{W-W_p}{R} \rfloor + 1$. We define a binary patch index map $\mathbf{M} \in \{0,1\}^{|\boldsymbol{\mathcal{P}}|\times H_g\times W_g}$, whose (k,h,w)-th element indicates whether the k-th patch $\mathbf{P}_k$ is assigned to the spatial location of (h,w). Directly optimizing $\mathbf{M}$ for every synthetic image requires a large amount of memory, as each synthetic image requires its own index map of size $|\boldsymbol{\mathcal{P}}| \times H_g \times W_g$.

To mitigate this overhead, we employ a low-rank decomposition, introducing a tensor $\mathbf{B} \in \mathbb{R}^{d\times H_g\times W_g}$ along with a shared expander matrix $\mathbf{E} \in \mathbb{R}^{|\boldsymbol{\mathcal{P}}|\times d}$, where $d \ll |\boldsymbol{\mathcal{P}}|$. Let $\mathbf{A} = \mathbf{EB} \in \mathbb{R}^{|\boldsymbol{\mathcal{P}}|\times H_g\times W_g}$ denote the patch selection score tensor. Then the (k,h,w)-th element of $\mathbf{M}$ is determined as follows:

$$M_{k,h,w} = \begin{cases} 1, & \text{if } k = \arg\max_j \left(A_{j,h,w} + \epsilon_j\right), \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $\epsilon_j = -\log(-\log u_j)$ with $u_j \sim \mathcal{U}(0,1)$ is a Gumbel noise term that encourages diverse patch selection during training. To enable end-to-end optimization, during training, we use the hard one-hot selection in the forward pass. In the backward pass, we replace the non-differentiable discrete selection with a Gumbel-Softmax Jang et al. (2017) trick to compute the gradients.

3.3 Multi-Stage Codebook Parameterization

Using a single codebook limits the representational capacity of the synthetic dataset unless the codebook size $|\boldsymbol{\mathcal{P}}|$ is significantly increased, which incurs substantial memory overhead. In network architecture design, large-kernel convolutions are often decomposed into multiple smaller-kernel operations to improve representational capacity under the same parameter budget. This decomposition allows each operation to learn distinct and complementary features, which are then combined to form richer representations Szegedy et al. (2016); Howard et al. (2017). Motivated by this obser-

vation, we employ multiple codebooks for dataset distillation, enabling richer representations under the same storage budget.

We define a stage as the process of selecting patches from a codebook according to a patch index map. We generate a set of L learned codebooks, $\{\boldsymbol{\mathcal{P}}^{(l)}\}_{l=1}^{L}$, and obtain the corresponding binary patch index maps $\{\mathbf{M}^{(l)}\}_{l=1}^{L}$. For multi-stage selection, $\mathbf{B}$ is shared across all stages and expanded by $\mathbf{E} \in \mathbb{R}^{L|\boldsymbol{\mathcal{P}}|\times d}$ into $L|\boldsymbol{\mathcal{P}}|$ logits, which are partitioned into L independent index maps. Given $\{\mathbf{M}^{(l)}\}_{l=1}^{L}$ and $\{\boldsymbol{\mathcal{P}}^{(l)}\}_{l=1}^{L}$, we generate a final synthetic image $\tilde{\mathbf{X}} \in \mathbb{R}^{C\times H\times W}$ by combining the selected patches at multiple stages. Let $\mathbf{M}^{(l)}_{k,:,:} \in \mathbb{R}^{H_g\times W_g}$ denote the k-th channel of the binary index map at stage l, indicating the spatial locations to which the k-th codebook patch $\mathbf{P}^{(l)}_{k}$ is assigned. The unnormalized synthetic image $\tilde{\mathbf{X}}' \in \mathbb{R}^{C\times H\times W}$ is obtained by aggregating the codebook patches selected from all stages across the spatial grids, given by:

$$\tilde{\mathbf{X}}' = \sum_{l=1}^{L}\sum_{k=1}^{|\boldsymbol{\mathcal{P}}|} \mathbf{M}^{(l)}_{k,:,:} \circledast_{R}^{\top} \mathbf{P}^{(l)}_{k}, \tag{5}$$

where $\circledast_{R}^{\top}$ denotes the transposed convolution operator with stride R. Since the stride is typically smaller than the patch size, adjacent patches spatially overlap when mapped onto the full-resolution image. Consequently, the unnormalized image $\tilde{\mathbf{X}}'$ contains accumulated pixel values in these overlapping regions. To handle the spatial overlap, we obtain the final synthetic image $\tilde{\mathbf{X}}$ by averaging the values at each pixel.

We parameterize the synthetic dataset using the proposed multi-codebook framework. Let $\boldsymbol{\Theta}$ denote the set of all learnable parameters, which consists of the multiple codebooks $\{\boldsymbol{\mathcal{P}}^{(l)}\}_{l=1}^{L}$, per-image parameter matrices $\{\mathbf{B}_i\}_{i=1}^{|\boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}|}$, and a single expander matrix $\mathbf{E}$. The synthetic dataset is then parameterized as $\boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}$. The goal of dataset distillation is to jointly optimize all parameters in $\boldsymbol{\Theta}$, ensuring that the synthetic dataset captures the knowledge of the original dataset. We formulate this as an optimization problem to minimize the distillation objective function $D(\cdot,\cdot)$ as follows:

$$\boldsymbol{\Theta}^{*} = \arg\min_{\boldsymbol{\Theta}} D(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}), \text{ where } \boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}} = \{\tilde{\mathbf{X}}_i\}_{i=1}^{|\boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}|}. \tag{6}$$

Note that our framework is agnostic to the distillation objective function $D(\cdot,\cdot)$ which can be any standard metric measuring the discrepancy between the real and synthetic datasets, such as GM Zhao et al. (2021), DM Zhao & Bilen (2023), TM Cazenavette et al. (2022), or SRe^{2}L Yin et al. (2023).

3.4 Storage Budget Analysis

We analyze the storage cost of the multi-stage codebooks and the corresponding patch-index maps. During training, the index map stores a one-hot vector of size $|\boldsymbol{\mathcal{P}}|$ at each spatial location, but stores the patch index rather than the one-hot vector after training, requiring $\lceil \log_2 |\boldsymbol{\mathcal{P}}| \rceil$ bits. Assuming a general multi-codebook framework with L stages where each stage has the same codebook size $|\boldsymbol{\mathcal{P}}^{(l)}| = |\boldsymbol{\mathcal{P}}|$, the total number of patches is $L\cdot|\boldsymbol{\mathcal{P}}|$. Each patch costs $C\cdot H_p\cdot W_p\cdot 32$ bits. The index map for each synthetic image requires $L\cdot H_g\cdot W_g\cdot\lceil\log_2|\boldsymbol{\mathcal{P}}|\rceil$ bits. We select the hyperparameters to satisfy the following overall storage constraint:

$$\underbrace{L|\boldsymbol{\mathcal{P}}|CH_pW_p32}_{\text{codebooks}} + \underbrace{|\boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}|LH_gW_g\lceil\log_2|\boldsymbol{\mathcal{P}}|\rceil}_{\text{index maps}} \le \underbrace{|\boldsymbol{\mathcal{S}}|CHW32}_{\text{raw pixels}}. \tag{7}$$

Here, $|\boldsymbol{\mathcal{S}}_{\boldsymbol{\Theta}}|$ denotes the actual number of parameterized synthetic images, while $|\boldsymbol{\mathcal{S}}|$ denotes the number of raw synthetic images that can be stored under the same budget.

To illustrate the storage advantage of the multi-codebook design, consider $H_p = W_p = 4$, $R = 2$, $H = W = 32$, $C = 3$, $|\mathcal{S}_\Theta| = 50$, and $|\mathcal{S}| = 1$. The raw-pixel budget is $98{,}304$ bits. A single codebook with $L = 1$ and $|\mathcal{P}| = 32$ requires $49{,}152 + 56{,}250 = 105{,}402$ bits, which slightly exceeds the budget. In contrast, multi-stage codebooks with $L = 2$ and $|\mathcal{P}| = 8$ require $24{,}576 + 67{,}500 = 92{,}076$ bits, thereby fitting within the same budget. The multi-stage design also increases the number of possible patch combinations per location from 32 to 64, illustrating a more favorable trade-off between storage cost and patch-combination diversity.

Table 1: Comparison with existing methods on CIFAR-10 and CIFAR-100. "Whole" denotes the performance achieved by training on the full original dataset. The best and second best results are highlighted in **bold** and underlined, respectively.

Dataset		CIFAR-10			CIFAR-100		
IPC		1	10	50	1	10	50
Pixel-based	DM	26.0±0.8	48.9±0.6	63.0±0.4	11.4±0.3	29.7±0.3	43.6±0.4
	DC	28.3±0.5	44.9±0.5	53.9±0.5	12.8±0.3	25.2±0.3	—
	TM	46.3±0.8	65.3±0.7	71.6±0.2	24.3±0.3	40.1±0.4	47.7±0.2
Parameterized	IDC	50.0±0.4	67.5±0.5	74.5±0.2	—	—	—
	FreD	60.6±0.8	70.3±0.3	75.8±0.1	34.6±0.4	42.7±0.2	47.8±0.1
	HaBa	48.3±0.8	69.9±0.4	74.0±0.2	—	—	47.0±0.2
	RTP	66.4±0.4	71.2±0.4	73.6±0.5	34.0±0.4	42.9±0.7	—
	HMN	65.7±0.3	73.7±0.1	76.9±0.2	36.3±0.2	45.4±0.2	48.5±0.2
	SPEED	63.2±0.1	73.5±0.2	77.7±0.4	40.4±0.4	45.9±0.3	49.1±0.2
	NSD	68.5±0.8	73.4±0.2	75.2±0.6	36.5±0.3	46.1±0.2	—
	DDiF	66.5±0.4	74.0±0.4	77.5±0.3	42.1±0.2	46.0±0.2	49.9±0.2
	RUO	70.3±0.7	77.5±0.3	79.7±0.3	44.4±0.3	49.2±0.4	—
	Ours	**71.3±0.2**	**78.2±0.3**	**81.4±0.3**	**45.8±0.2**	**51.4±0.3**	**53.4±0.3**
Whole			84.8±0.1			56.2±0.3	

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

We evaluate our method on CIFAR-10/100 Krizhevsky et al. (2009) and six 128×128 ImageNet subsets Deng et al. (2009), following the evaluation protocol of prior works Shin et al. (2025); Bao et al. (2025). We also validate the scalability on full ImageNet-1K. To verify the compatibility of the proposed framework, we evaluate under four widely used distillation objectives, GM Zhao et al. (2021), DM Zhao & Bilen (2023), TM Cazenavette et al. (2022), and SRe^{2}L Yin et al. (2023). We use ConvNetD3 for CIFAR and ConvNetD5 for ImageNet subsets as the distillation architecture, and report the mean and standard deviation of test accuracy over 5 independent runs. We note that IPC (images per class) denotes the storage budget equivalent to storing raw pixel images, not the actual number of synthetic images used during training. Since parameterized images require less storage than raw pixels, the number of training data per class (TPC) can be significantly larger than the IPC. For CIFAR, we use 4×4 patches with stride 2, whereas for ImageNet, we use 8×8 patches with stride 4. The codebook size, number of stages, and TPC are adjusted for each experiment according to the corresponding storage budget. Further implementation details, including hyperparameter settings, are provided in the supplementary material.

4.2 PERFORMANCE COMPARISON

Performance on Standard Benchmarks. We first conduct a comprehensive evaluation of our method on the standard CIFAR-10 and CIFAR-100 benchmarks across three budget settings of IPC: 1, 10, and 50. As summarized in Table 1, our approach consistently achieves state-of-the-art (SOTA) performance across all IPC settings on both datasets. On CIFAR-100, where the larger number of classes makes efficient parameterization more important, the improvements are more pronounced. These results validate that our codebook parameterization effectively captures the redundancy in local visual patterns, enabling a more informative use of the storage budget compared to existing methods.

Scalability to High-Resolution. To validate the scalability of our approach to higher resolution images, we conduct experiments on six 128×128 ImageNet subsets at IPC=1. As shown in Table 2, our method consistently outperforms all existing methods across all six subsets. The improvements are particularly large on fine-grained subsets such as Woof by +6.0% and Meow by +5.9%, where subtle inter-class differences require precise local details to be preserved in the synthetic images.

Table 2: Comparison with existing methods on six 128×128 ImageNet subsets at IPC=1 using **TM** objective.

Subset	Whole	TM	GLaD	H-GLaD	IDC	FreD	HaBa	SPEED	NSD	DDiF	RUO	Ours
Nette	87.4	51.4	38.7	45.4	61.4	66.8	51.9	66.9	68.6	72.0	76.5	**80.8**
Woof	67.0	29.7	23.4	28.3	34.5	38.3	32.4	38.0	35.2	42.9	46.0	**52.0**
Fruit	63.9	28.8	23.1	25.6	38.0	43.7	34.7	43.4	39.8	48.2	50.8	**53.8**
Yellow	84.4	47.5	–	–	56.5	63.2	50.4	62.6	61.0	69.0	74.3	**77.8**
Meow	66.7	33.3	26.0	29.6	39.5	43.2	36.9	43.6	45.2	47.4	53.5	**59.4**
Squawk	87.5	41.0	35.8	39.7	50.2	57.0	41.9	60.9	52.9	67.0	74.8	**78.0**
Average	76.1	38.6	29.4	33.7	46.7	52.0	41.4	52.6	50.5	57.8	62.7	**67.0**

Table 3: Comparison with existing methods on six 128×128 ImageNet subsets at IPC=1. We evaluate our method using both **GM** and **DM** objectives.

Subset	**Gradient Matching (GM)**								**Distribution Matching (DM)**							
	Vanilla	GLaD	H-GLaD	IDC	FreD	DDiF	RUO	Ours	Vanilla	GLaD	H-GLaD	IDC	FreD	DDiF	RUO	Ours
Nette	34.2	35.4	36.9	45.4	49.1	61.2	67.2	**68.6**	30.4	32.2	34.8	48.3	56.2	69.2	71.9	**76.2**
Woof	22.5	22.3	24.0	25.5	26.1	35.2	33.1	**41.6**	20.7	21.2	23.9	27.0	31.0	42.0	46.4	**46.8**
Fruit	21.0	20.7	22.4	26.8	30.0	37.8	37.0	**45.8**	20.4	21.8	24.4	29.9	33.4	45.3	49.0	**51.2**
Yellow	37.1	–	–	–	–	–	58.9	**63.8**	36.0	–	–	–	–	–	69.2	**72.2**
Meow	22.0	22.6	24.1	25.3	28.7	39.1	42.0	**46.0**	20.1	22.3	24.2	30.5	33.3	45.8	48.8	**53.2**
Squawk	32.0	33.8	35.3	34.6	39.7	54.3	58.6	**67.4**	26.6	27.6	29.5	38.8	42.7	64.6	69.0	**73.2**
Average	28.1	27.0	28.5	31.5	34.7	45.5	49.5	**55.5**	25.7	25.0	27.4	34.9	39.3	53.4	59.1	**62.1**

On coarser subsets such as Nette and Squawk, where classes are more visually distinct, our method still achieves substantial gains of 4.3% and 3.2%, respectively. Compared to CIFAR benchmarks, the improvements on ImageNet subsets are consistently larger across all subsets. We attribute this to the higher resolution, which contains significantly more local redundancy across images, allowing our shared codebooks to exploit these recurring patterns more effectively.

Compatibility with Other Distillation Methods. Table 3 reports classification accuracies on six ImageNet subsets under the GM and DM objectives. Our method consistently achieves the best accuracy across all subsets under both objectives, improving over RUO Bao et al. (2025) and DDiF Shin et al. (2025) in all cases. This demonstrates that the proposed codebook parameterization is compatible with fundamentally different distillation losses, rather than being tailored to a specific optimization strategy. This compatibility arises because our method only reparameterizes the synthetic image space through differentiable patch assembly, allowing the same codebook parameters to be optimized with GM, DM, or TM objectives.

Cross-Architecture Generalization. To evaluate the robustness of our distilled dataset, we test its generalization performance across unseen architectures. Following previous works Shin et al. (2025); Bao et al. (2025), we distill synthetic images using ConvNetD5 and evaluate them on AlexNet Krizhevsky et al. (2012), VGG11 Simonyan & Zisserman (2015), ResNet18 He et al. (2016), and ViT Dosovitskiy et al. (2021), reporting the averaged accuracies in Table 4. Although the evaluation architectures are significantly more powerful than the ConvNetD5 used during distillation, the distilled datasets generally do not fully benefit from their increased capacity, reflecting the inherent difficulty of transferring distilled knowledge across architectures. Despite this challenge, our method maintains the best performance across all subsets, suggesting that the codebook parameterization produces synthetic images with transferable visual patterns applicable to both convolutional and transformer-based models.

Effectiveness under Matched TPC. Beyond storage efficiency, our codebook parameterization exploits recurring local structures in natural images by representing them with shared patch parameters. When the same patch is used across images or spatial locations, its gradients are accumulated on the same codebook entry, allowing repeated structures to be optimized jointly. To verify that the gain is not merely due to using less storage, we compare methods under matched TPC, where all methods use the same number of synthetic training samples per class. As shown in Table 5, our method nearly matches the TM baseline on CIFAR-10 with over $50\times$ less storage, retains most of the SRe^{2}L accuracy on ImageNet-1K with over $100\times$ less storage, and outperforms DDiF under the same setting.

Table 4: Cross-architecture generalization on ImageNet subsets (128×128) at IPC=1. Accuracies are averaged across AlexNet, VGG11, ResNet18, and ViT.

Subset	TM	GLaD	H-GLaD	IDC	FreD	DDiF	RUO	Ours
Nette	22.0	30.4	30.8	27.9	36.2	59.3	65.6	**67.4**
Woof	14.8	17.1	17.4	19.5	23.7	34.1	36.0	**40.9**
Fruit	17.1	21.1	21.5	23.9	23.6	39.3	42.8	**44.5**
Yellow	22.3	–	–	28.0	31.2	51.1	58.0	**62.2**
Meow	16.2	19.6	20.1	19.8	19.1	33.8	40.6	**43.1**
Squawk	25.5	28.2	28.8	29.9	37.4	54.0	61.2	**65.2**
Average	19.7	23.3	23.7	24.8	28.5	45.3	50.7	**53.9**

Table 5: Comparison under matched TPC settings. We report accuracy and storage budget on CIFAR-10 with TPC $= 50$ and ImageNet-1K with TPC $= 10$.

Dataset	Method	Budget (MB)	Acc.
CIFAR-10	TM	5.9	71.6
	+ DDiF	0.2	71.0
	+ Ours	0.1	71.3
ImageNet-1K	SRe^{2}L	5742.2	21.3
	+ DDiF	134.4	14.6
	+ Ours	45.6	20.8

Table 6: Performance comparison with existing methods on ImageNet-1K (224×224) at IPC=1 and IPC=10. All synthetic datasets are generated with ResNet-18.

IPC	Decoupled					Parameterized		
	SRe^{2}L	RDED	NRR-DD	FADRM	RETA	FreD	DDiF	Ours
1	0.1	6.6	11.6	14.7	16.8	13.0	30.4	**51.5**
10	21.3	42.0	46.1	50.9	53.2	36.5	59.1	**65.8**

Table 7: Effect of sharing scope at matched TPC$=$ 50 on CIFAR-10.

Sharing Scope	Budget	Acc.
Image-level	0.77	70.3
Class-level	0.20	70.7
Global (Ours)	0.15	71.6

Scalability to ImageNet-1K. Table 6 compares our method with both decoupled approaches and parameterization methods on ImageNet-1K. All parameterization methods, including ours, are applied on top of the SRe^{2}L Yin et al. (2023) pipeline. Our method achieves the best accuracy at both IPC settings. Notably, our method achieves 51.5% at IPC=1 and 65.8% at IPC=10, outperforming DDiF by substantial margins. These results demonstrate that our parameterization remains effective when scaled to the full ImageNet-1K dataset across different storage budgets.

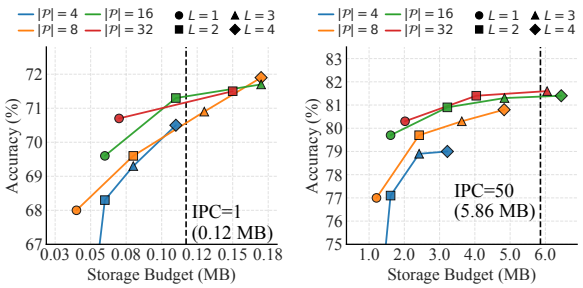
Effectiveness of Sharing Mechanism. To examine the effect of sharing scope, we compare image-level, class-level, and global sharing at matched TPC with $L = 1$ for all configurations. As shown in Table 7, global sharing achieves the highest accuracy while requiring the least storage among the three configurations. These results indicate that broader patch sharing provides a more favorable accuracy–storage trade-off.

4.3 Visualization

Figure 3 visualizes the learned patch-level codebooks and the corresponding synthetic images on CIFAR-10 at IPC=1. The learned patches can be broadly divided into two categories. One consists of grayscale patches that encode local structural information, such as edges and boundaries, while the other consists of color patches dominated by primary colors, including red, green, and blue. By selecting and combining these two types of patches through the index maps, our method constructs diverse visual patterns from a small set of shared patches. While the images generated at individual stages appear either overly smooth or noisy, the combined result exhibits both clear object structures and detailed textures. This suggests that the multi-stage codebook framework enables each stage to learn complementary visual patterns, producing more informative synthetic images.



Figure 3: Qualitative results on CIFAR-10. The left and middle panels show the first- and second-stage results, respectively. In each panel, the upper row presents the learned codebook patches, while the lower row shows the images reconstructed using the corresponding stage. The right panel shows the synthetic images obtained by combining both stages.



(a) (b)

Figure 4: Trade-off between the number of stages L and the codebook size $|\mathcal{P}|$ on CIFAR-10 under different storage budgets. (a) IPC$=1$. (b) IPC$=50$.

Table 8: Effect of multi-stage codebooks on CIFAR-10.

Method	IPC=1		IPC=50	
	Budget	Acc.	Budget	Acc.
Baseline	0.12	46.3	5.86	71.6
+ Single	0.12	69.4	4.03	81.0
+ Multi	0.11	**71.3**	4.03	**81.4**

Table 9: Ablation study on the codebook domain at IPC$=1$.

Dataset	C-10	C-100	Woof	Nette
Pixel	71.0	44.1	51.6	80.0
DCT	**71.3**	**45.8**	**52.0**	**80.8**

4.4 Ablation Study

Ablation on Multi-Stage Codebooks. Table 8 analyzes the effectiveness of the proposed multi-stage codebooks on CIFAR-10. At IPC=1, the baseline achieves 46.3%. Patch-level sharing with a single-stage codebook increases the accuracy to 69.4%, because the shared patch representations use the storage budget far more effectively by explicitly reusing recurring local structures. The multi-stage codebooks further improve the accuracy to 71.3% by combining complementary patches across multiple stages, improving over the baseline by 25.0%p in total. The same trend holds at IPC=50, where the single-stage codebook improves accuracy from 71.6% to 81.0% and the multi-stage design further raises it to 81.4%, confirming that the benefits of the multi-stage design are not limited to low storage budgets.

Trade-off Between $|\mathcal{P}|$ and L. Figure 4 analyzes the trade-off between the number of stages L and the codebook size $|\mathcal{P}|$ under two storage budgets. Figures 4(a) and (b) present the results at IPC$=1$ and IPC$=50$, respectively. Increasing either L or $|\mathcal{P}|$ generally improves accuracy by expanding the compositional capacity of the representation. A larger codebook provides a more diverse set of local patches, while additional stages enable these patches to be combined more flexibly. However, both factors increase the storage cost and therefore cannot be enlarged arbitrarily under a fixed budget. We consequently determine the best-performing combination of L and $|\mathcal{P}|$ for each budget. At IPC$=1$, $L=2$ with $|\mathcal{P}|=16$ achieves the highest accuracy of 71.3%, whereas at IPC$=50$, the larger available budget favors $L=2$ with $|\mathcal{P}|=32$.

Effectiveness of DCT Representation. Table 9 investigates the effect of the codebook representation domain. The **Pixel** baseline stores learnable parameters in the spatial domain, whereas the **DCT** variant represents them as frequency coefficients. DCT improves accuracy across all four benchmarks without additional storage cost, with larger gains on CIFAR-100 and ImageNette. These consistent improvements suggest that the frequency-domain representation provides a more effective parameterization for codebook optimization.

5 Conclusion

We proposed a novel parameterization method for dataset distillation based on multiple patch-level learnable codebooks that explicitly share local structural patterns across the synthetic images. By decomposing the synthetic dataset into shared patches and discrete per-image index maps, our approach effectively reduces the redundant storage of recurring local visual patterns. Furthermore, extending this formulation to a multi-codebook framework allows us to expand the number of representable patch combinations. Extensive experiments on CIFAR-10, CIFAR-100, ImageNet subsets, and ImageNet-1K demonstrate that our method achieves state-of-the-art performance across GM, DM, TM, and SRe^{2}L objectives. Notably, the proposed parameterization maintains competitive accuracy under matched TPC while substantially reducing storage, and remains effective when scaled to full ImageNet-1K. We believe that exploring patch-level redundancy offers a promising direction for further improving the efficiency of dataset distillation.

AI USE STATEMENT

In this work, we used generative AI tools to assist with interpreting experimental results. We did not use generative AI tools for other tasks requiring disclosure. Additionally, we used generative AI tools to assist with creating and editing software code, drafting and polishing parts of the manuscript, and creating or modifying scientific figures. All AI-assisted analyses, code, text, and figures were manually reviewed by the authors; generated code was tested for correctness, and interpretations of experimental results were checked against the reported experimental outcomes. We take responsibility for the final content of this work, including text, claims, code, figures, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed parameterization framework and storage analysis in the main paper. The appendix includes the experimental setup, optimization and evaluation protocols, hyperparameter settings for TM, GM, DM, and SRe^{2}L. These details are intended to facilitate reproduction of the reported experiments. Source code will be made publicly available upon publication.

REFERENCES

Youneng Bao, Yiping Liu, Zhuo Chen, Yongsheng Liang, Mu Li, and Kede Ma. Dataset distillation as data compression: A rate-utility perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 519–529, 2025.

Deyu Bo and Xinchao Wang. Point cloud dataset distillation. In *Proceedings of International Conference on Machine Learning*, 2025.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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

Jiacheng Cui, Xinyue Bi, Yaxin Luo, Xiaohan Zhao, Jiacheng Liu, and Zhiqiang Shen. Fadrm: Fast and accurate data residual matching for dataset distillation. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 38, pp. 62610–62655, 2026.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

Wenxiao Deng, Wenbin Li, Tianyu Ding, Lei Wang, Hongguang Zhang, Kuihua Huang, Jing Huo, and Yang Gao. Exploiting inter-sample and inter-feature relations in dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Zhiwei Deng, Jialian Wu, and Zhangyang Wang. Remember the past: Distilling datasets into addressable memories for neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, 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. In *International Conference on Learning Representations*, 2021.

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

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Ziyao Guo, Kai Wang, George Cazenavette, HUI LI, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. In *International Conference on Learning Representations*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022a.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022b.

Dongwook Kim, Jae-Young Yim, and Jae-Young Sim. Parameterization-based dataset distillation of 3d point clouds through learnable shape morphing. In *International Conference on Learning Representations*, 2026.

Jang-Hyun Kim, Seong Joon Park, Jinwoo Shin, and Taesup Moon. Dataset condensation via efficient synthetic-data parameterization. In *Proceedings of International Conference on Machine Learning*, 2022.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the Advances in Neural Information Processing Systems*, 25, 2012.

Muquan Li, Hang Gou, Yingyi Ma, Rongzheng Wang, Ke Qin, and Tao He. Fixed anchors are not enough: Dynamic retrieval and persistent homology for dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33963–33972, 2026.

Yongqi Li and Wenjie Li. Data distillation for text classification. *arXiv preprint arXiv:2104.08448*, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 2014.

Dai Liu, Jindong Gu, Hu Cao, Carsten Trinitis, and Martin Schulz. Dataset distillation by automatic training trajectories. In *Proceedings of the European Conference on Computer Vision*. Springer, 2024.

Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. In *Proceedings of the Advances in Neural Information Processing Systems*, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Aru Maekawa, Naoki Kobayashi, Kotaro Funakoshi, and Manabu Okumura. Dataset distillation with attention labels for fine-tuning bert. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 119–127, 2023.

Aru Maekawa, Satoshi Kosugi, Kotaro Funakoshi, and Manabu Okumura. Dilm: Distilling dataset into language model for text-level dataset distillation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 3138–3153, 2024.

Donghyeok Shin, Seungjae Shin, Byeongho Jeon, and Taesup Moon. Frequency domain-based dataset distillation. In *Proceedings of the Advances in Neural Information Processing Systems*, 2023.

Donghyeok Shin, HeeSun Bae, Gyuwon Sim, Wanmo Kang, and Il-chul Moon. Distilling dataset into neural field. In *International Conference on Learning Representations*, 2025.

Asaf Shul, Eliahu Horwitz, and Yedid Hoshen. Distilling datasets into less than one image. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=qsipSdfWeV`.

K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Peng Sun, Bei Shi, Daiwei Yu, and Tao Lin. On the diversity and realism of distilled dataset: An efficient dataset distillation paradigm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9390–9399, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Minh-Tuan Tran, Trung Le, Xuan-May Le, Thanh-Toan Do, and Dinh Phung. Enhancing dataset distillation via non-critical region refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10015–10024, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 30, 2017.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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

Xing Wei, Anjia Cao, Funing Yang, and Zhiheng Ma. Sparse parameterization for epitomic dataset distillation. In *Proceedings of the Advances in Neural Information Processing Systems*, 2023.

Shaolei Yang, Shen Cheng, Mingbo Hong, Haoqiang Fan, Xing Wei, and Shuaicheng Liu. Neural spectral decomposition for dataset distillation. In *Proceedings of the European Conference on Computer Vision*, pp. 275–290. Springer, 2024.

Jae-Young Yim, Dongwook Kim, and Jae-Young Sim. Dataset distillation of 3d point clouds via distribution matching. In *Proceedings of the Advances in Neural Information Processing Systems*, 2025.

Zeyuan Yin, Eric Xing, and Zhiqiang Shen. Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, pp. 73582–73603, 2023.

Hansong Zhang, Shikun Li, Fanzhao Lin, Weiping Wang, Zhenxing Qian, and Shiming Ge. Dance: Dual-view distribution alignment for dataset condensation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2024a.

Hansong Zhang, Shikun Li, Pengju Wang, Dan Zeng, and Shiming Ge. M3d: Dataset condensation by minimizing maximum mean discrepancy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024b.

Wenxiao Zhang, Ziqi Wang, Li Xu, Xun Yang, and Jun Liu. Informative point cloud dataset extraction for classification via gradient-based points moving. In *Proceedings of the ACM International Conference on Multimedia*, 2024c.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *Proceedings of International Conference on Machine Learning*, 2021.

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

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

Ganlong Zhao, Guanbin Li, Yipeng Qin, and Yizhou Yu. Improved distribution matching for dataset condensation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

John Zhi-Qin, John Xu, Luo Tao, Xiao Yanyang, and Ma Zheng. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5): 1746–1767, 2020.

Xinhao Zhong, Hao Fang, Bin Chen, Xulin Gu, Meikang Qiu, Shuhan Qi, and Shu-Tao Xia. Hierarchical features matter: A deep exploration of progressive parameterization method for dataset distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

A APPENDIX

Table 10: Hyperparameter settings for TM.

Dataset	**CIFAR-10**			**CIFAR-100**			**ImageNet**	
IPC	**1**	**10**	**50**	**1**	**10**	**50**	**1**	**10**
Expert epochs	2	2	2	2	2	2	2	2
Learning rate (Teacher)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Learning rate (Step size)	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-6}	10^{-6}
Max start epoch	40	40	40	40	40	40	40	40
Distillation steps	15000	15000	15000	15000	15000	15000	5000	5000
Synthetic steps	30	60	60	60	60	60	20	20
Learning rate ($\mathbf{\Theta}$)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Synthetic batch size	500	500	750	250	750	1250	256	256
Patch size H_p, W_p	4, 4	4, 4	4, 4	4, 4	4, 4	4, 4	8, 8	8, 8
Stride R	2	2	2	2	2	2	4	4
Codebook size $\lvert\boldsymbol{\mathcal{P}}\rvert$	16	16	32	16	16	64	64	64
Number of Stages L	2	2	2	2	3	3	2	3
Low-Rank dimension d	16	16	16	16	16	16	32	32
TPC	50	500	1500	50	250	400	128	512

Table 11: Hyperparameter settings for GM and DM on ImageNet subsets (128×128) at IPC=1.

Hyperparameter	**GM**	**DM**
Distillation Steps	2000	20000
Learning Rate ($\mathbf{\Theta}$)	0.01	0.01
Real batch size	256	256
Synthetic batch size	1280	2000
H_p, W_p	8, 8	16, 16
R	4	8
$\lvert\boldsymbol{\mathcal{P}}\rvert$	16	173
L	3	2
d	32	32
TPC	128	200

Table 12: Hyperparameter settings for SRe^{2}L on ImageNet-1K (224×224).

Hyperparameter	**SRe^{2}L**	
	IPC=1	**IPC=10**
Recovery iteration	4000	4000
Learning Rate ($\mathbf{\Theta}$)	0.05	0.05
Batch size	100	100
H_p, W_p	8, 8	8, 8
R	4	4
$\lvert\boldsymbol{\mathcal{P}}\rvert$	8	8
L	4	4
d	16	16
TPC	100	1000

A.1 EXPERIMENTAL DETAILS

A.1.1 EVALUATION SETUP

To evaluate with the ConvNet, we train the network from scratch for 1,000 epochs. We use the SGD optimizer with an initial learning rate of 0.01, a momentum of 0.9, and a weight decay of 5×10^{-4}, decaying the learning rate by a factor of 0.1 at epoch 500. For cross-architecture generalization, we evaluate the distilled dataset on four architectures, namely AlexNet Krizhevsky et al. (2012), VGG-11 Simonyan & Zisserman (2015), ResNet-18 He et al. (2016), and ViT Dosovitskiy et al. (2021). We use the Adam optimizer for 1,000 epochs with a learning rate of 10^{-3} for AlexNet, VGG-11, and ResNet-18, and 5×10^{-5} for ViT.

A.1.2 HYPERPARAMETER SETTINGS

We initialize the codebooks and index maps by fitting them to randomly selected images from the original dataset for 5,000 iterations using MSE loss with a learning rate of 0.1. We apply DSA Zhao & Bilen (2021) to both real and synthetic images, and ZCA whitening is applied to CIFAR-10/100 Krizhevsky et al. (2009), but not to the ImageNet Deng et al. (2009) subsets. The Gumbel-Softmax temperature is fixed at $\tau = 1.0$ throughout training. For trajectory matching (TM),

Table 13: Detailed performance evaluation of cross-architecture generalization. The best and second best results are highlighted in **bold** and underlined, respectively.

Network	Method	Nette	Woof	Fruit	Yellow	Meow	Squawk
AlexNet	TM	13.2±0.6	10.0±0.0	10.0±0.0	11.0±0.2	9.8±0.0	–
	IDC	17.4±0.9	16.5±0.7	17.9±0.7	20.6±0.9	16.8±0.5	20.7±1.0
	FreD	35.7±0.4	23.9±0.7	15.8±0.7	19.8±1.2	14.4±0.5	36.3±0.3
	DDiF	60.7±2.3	36.4±2.3	41.8±0.6	56.2±0.8	40.3±1.9	60.5±0.4
	RUO	64.1±1.3	32.6±1.8	40.7±0.5	49.9±2.4	36.9±1.6	55.9±2.5
	Ours	**67.2±1.9**	**40.5±1.4**	**44.4±1.2**	**61.8±1.0**	**44.6±3.3**	**66.3±1.6**
VGG-11	TM	17.4±2.1	12.6±1.8	11.8±1.0	16.9±1.1	13.8±1.3	–
	IDC	19.6±1.5	16.0±2.1	13.8±1.3	16.8±3.5	13.1±2.0	19.1±1.2
	FreD	21.8±2.9	17.1±1.7	12.6±2.6	18.2±1.1	13.2±1.9	18.6±2.3
	DDiF	53.6±1.5	29.9±1.9	33.8±1.9	44.2±1.7	32.0±1.8	37.9±1.5
	RUO	70.3±2.5	**42.1±2.9**	44.6±1.4	66.9±0.4	**49.6±1.4**	66.8±2.8
	Ours	**70.7±1.5**	41.9±1.3	**44.7±1.6**	**69.8±2.3**	47.1±2.1	**67.9±1.3**
ResNet-18	TM	34.9±2.3	20.7±1.0	23.1±1.5	43.4±1.1	22.8±2.2	–
	IDC	43.6±1.3	23.2±0.8	32.9±2.8	44.2±3.5	28.2±0.5	47.8±1.9
	FreD	48.8±1.8	28.4±0.6	34.0±1.9	49.3±1.1	29.0±1.8	50.2±0.8
	DDiF	63.8±1.8	37.5±1.9	42.0±1.9	55.9±1.0	35.8±1.8	62.6±1.5
	RUO	70.6±1.2	39.9±1.8	**44.6±0.6**	67.1±1.5	47.8±0.9	66.1±1.6
	Ours	**71.1±1.4**	**45.2±3.0**	42.7±2.4	**67.2±3.5**	**51.0±2.0**	**67.0±3.2**
ViT-B	TM	22.6±1.1	15.9±0.4	23.3±0.4	18.1±1.3	18.6±0.9	–
	IDC	31.0±0.6	22.4±0.8	31.1±0.8	30.3±0.6	21.4±0.7	32.2±1.2
	FreD	38.4±0.7	25.4±1.7	31.9±1.4	37.6±2.0	19.7±0.8	44.4±1.0
	DDiF	59.0±0.4	32.8±0.8	39.4±0.8	47.9±0.6	27.0±0.6	54.8±1.1
	RUO	57.5±1.2	29.5±0.3	41.4±1.1	48.2±0.8	28.0±0.9	55.8±1.4
	Ours	**60.6±1.2**	**36.2±1.7**	**46.3±1.8**	**49.9±3.6**	**29.6±0.7**	**59.7±0.4**

we follow Cazenavette et al. (2022) to generate expert trajectories. Across all experiments, we employ the Adam optimizer to optimize the synthetic data ($\boldsymbol{\Theta}$), and all experiments are conducted on NVIDIA A100 GPUs. Tables 10, 11, and 12 present optimized hyperparameters used for our experiments. As shown in Table 10, we use a consistent patch size for the CIFAR datasets across all IPC settings, while adjusting the number of stages and the codebook size. For the ImageNet subsets, however, larger patches are used to handle higher resolution. For gradient matching (GM) and distribution matching (DM) on the ImageNet subsets, the hyperparameters are configured differently, as shown in Table 11. Specifically, GM operates with relatively smaller patches to generate fewer training samples per class (TPC). In contrast, DM utilizes larger patches and a more extensive codebook to synthesize a greater number of training samples per class. For ImageNet-1K, we apply our parameterization on top of the SRe^{2}L Yin et al. (2023) pipeline, where the synthetic images are optimized with the recovery objective of SRe^{2}L and evaluated under its soft-label protocol. The corresponding hyperparameters are provided in Table 12.

A.2 Extended Experimental Results

A.2.1 Performance Evaluation

Cross-Architecture Generalization. Table 13 provides the detailed results of cross-architecture generalization. Our method achieves the best accuracy in most cases. On AlexNet Krizhevsky et al. (2012), our method consistently outperforms all baselines across all six subsets, with particularly large gains on Meow and Yellow. On VGG-11 Simonyan & Zisserman (2015) and ResNet-18 He et al. (2016), our method and RUO Bao et al. (2025) perform competitively, with our method leading on most subsets while RUO is better in a few cases, including Woof and Meow with VGG-11. On ViT Dosovitskiy et al. (2021), which has no convolutional inductive bias and is the hardest architecture for transfer, our method achieves the best accuracy on all six subsets, outperforming both DDiF Shin et al. (2025) and RUO by clear margins.

Table 14: Accuracies on ImageNet subsets at IPC=10 with TM.

Subset	TM	IDC	FreD	HaBa	SPEED	DDiF	Ours
Nette	63.0	70.8	72.0	64.7	72.9	74.6	**83.0**
Woof	35.8	39.8	41.3	38.6	44.1	44.9	**55.4**
Fruit	40.3	46.4	47.0	42.5	50.0	49.8	**59.2**
Yellow	60.0	68.7	69.2	63.0	70.5	70.5	**81.6**
Meow	40.4	47.9	48.6	42.9	52.0	50.6	**60.8**
Squawk	52.3	65.4	67.3	56.8	71.8	72.3	**82.2**
Average	48.6	56.5	57.6	51.4	60.2	60.5	**70.4**

Table 15: Accuracies on ImageNet subsets at IPC=1 with DM.

Subset	DM	IDC	FreD	DDiF	Ours
Nette	32.1	53.7	54.2	67.8	**75.6**
Woof	20.0	30.2	31.2	39.6	**49.0**
Fruit	19.5	33.1	32.5	43.2	**49.2**
Yellow	33.4	52.2	49.1	63.1	**74.6**
Meow	21.2	34.6	34.0	44.8	**53.8**
Squawk	27.6	47.0	43.1	67.0	**76.6**
Average	25.6	41.8	40.7	54.3	**63.1**

Performance with a Larger Budget. Table 14 reports the classification accuracies on six ImageNet Deng et al. (2009) subsets at 128×128 resolution under IPC=10 using TM Cazenavette et al. (2022). Our method substantially outperforms all baselines on every subset. For instance, our method surpasses DDiF Shin et al. (2025) by 8.4% on Nette, 10.5% on Woof, and 9.9% on Squawk. We attribute this to the increased budget at IPC=10, which allows our method to generate a larger number of synthetic images while sharing the codebook, amplifying the storage efficiency of our parameterization.

Performance at Higher Resolution. Table 15 reports the classification accuracies on six ImageNet Deng et al. (2009) subsets at 256×256 resolution under IPC=1 with DM Zhao & Bilen (2023). Our method outperforms all baselines on every subset, surpassing DDiF Shin et al. (2025) by 7.8% on Nette, 9.4% on Woof, and 9.6% on Squawk. The improvements at 256×256 are consistently larger than those at 128×128, which is consistent with our observation in the main paper that higher resolution images contain more local redundancy for our shared codebooks to exploit.

A.2.2 Ablation Studies

Patch Size and Stride. Table 16 examines the effect of patch size and stride. The experiments are conducted on CIFAR-10 at TPC=50, and the storage budget is measured in MB. Smaller patches with a small stride produce a dense grid of 31×31, achieving the highest accuracy of 71.8% but at a large budget of 0.46 MB due to the cost of storing a large index map. The setting 4×4 with $R = 2$ achieves a comparable accuracy of 71.3% at only 0.11 MB, offering a much better trade-off between accuracy and storage. Increasing the patch size to 8×8 or 16×16 significantly reduces the grid resolution, resulting in fewer spatial locations for patch placement and a notable drop in accuracy. Furthermore, the results demonstrate that allowing patches to overlap consistently yields better performance compared to non-overlapping settings. For instance, using a 4×4 patch with an overlapping stride of $R = 2$ achieves an accuracy of 71.3%, outperforming the non-overlapping 2×2 with $R = 2$ setting while maintaining an efficient storage budget of 0.11 MB.

Table 16: Ablation on patch size and stride with fixed $L = 2$ and $|\boldsymbol{\mathcal{P}}| = 16$.

$H_p \times W_p$	R	$H_g \times W_g$	**Budget**	**Acc.**
2×2	1	31×31	0.46	71.8
2×2	2	16×16	0.12	70.9
4×4	2	15×15	0.11	71.3
4×4	4	8×8	0.04	68.6
8×8	4	7×7	0.05	67.7
8×8	8	4×4	0.03	62.3
16×16	8	3×3	0.10	59.2

Low-Rank Dimension. Table 17 reports the effect of the low-rank dimension d used in the index map decomposition. The experiments are conducted on CIFAR-10 Krizhevsky et al. (2009) at TPC=50, and the parameter budget is measured in MB. Accuracy improves as d increases from 2 to 16, as a higher dimension allows the expander matrix to produce more discriminative selection scores. However, at $d = 32$, accuracy decreases to 70.8%, suggesting that an overly large d introduces unnecessary parameters that may hinder optimization. Note that d affects the

Table 17: Ablation on the low-rank dimension d.

d	**2**	**4**	**8**	**16**	**32**
Params.	0.87	1.72	3.44	6.87	13.74
Acc.	70.1	70.4	71.0	**71.3**	70.8

number of learnable parameters and the computational cost during optimization, but not the final storage cost, since the index maps are stored as discrete patch indices after training. The size of the learnable patch index map parameters grows linearly with d, from 0.87 MB at $d = 2$ to 13.74 MB at $d = 32$. Based on these results, we use $d = 16$ as the default setting for CIFAR experiments, while using $d = 32$ for ImageNet subsets.

Initialization Strategy. Table 18 compares two initialization strategies on CIFAR-10 under TPC=50. While random noise initialization achieves a competitive accuracy of 70.3% after 30,000 iterations, initializing the codebooks and index maps by fitting to real images yields a higher accuracy of 71.3% in just 15,000 iterations. Although starting from noise yields comparable performance, these results show that the image fitting is a more efficient strategy, achieving better accuracy with significantly fewer training iterations.

Table 18: Ablation on the initialization strategy.

Initialization	**Acc.**
Random noise	70.3
Real image fitting	**71.3**

Gumbel Temperature. We follow the standard setting commonly used in Gumbel-Softmax and do not tune τ. As shown in Table 19, performance remains relatively stable for $\tau \geq 1.0$, supporting the use of $\tau = 1.0$ as a robust default. In contrast, a smaller temperature of $\tau = 0.5$ leads to a noticeable performance drop, suggesting that overly sharp patch assignments can make the optimization less stable.

Table 19: Ablation on τ.

τ	0.5	1.0	2.0	5.0
CIFAR-10	66.2	71.3	71.7	71.2
CIFAR-100	39.5	45.8	45.1	45.0

A.2.3 Computational Efficiency

Table 20 compares the time cost for generating synthetic images at inference time. Since our method only requires a single transposed convolution to assemble patches from the codebook, the generation is extremely fast. In contrast, RUO Bao et al. (2025) requires a forward pass through a decoder network for each image. Our method is up to 198× faster than RUO on CIFAR-100 and 46× faster on ImageNet subsets, demonstrating the practical efficiency of our codebook-based parameterization.

Table 20: Time cost (ms) for generating synthetic images. Results are measured on a single NVIDIA A100 GPU.

Dataset	**CIFAR-10**			**CIFAR-100**		**ImageNet Subsets** (128×128)				
TPC	**64**	**240**	**718**	**48**	**120**	**1**	**8**	**15**	**51**	**102**
RUO	55.30	105.04	318.57	677.13	706.31	51.95	52.32	60.58	161.51	295.87
Ours	**0.81**	**1.84**	**5.01**	**3.42**	**8.15**	**1.07**	**1.13**	**1.68**	**4.23**	**7.94**

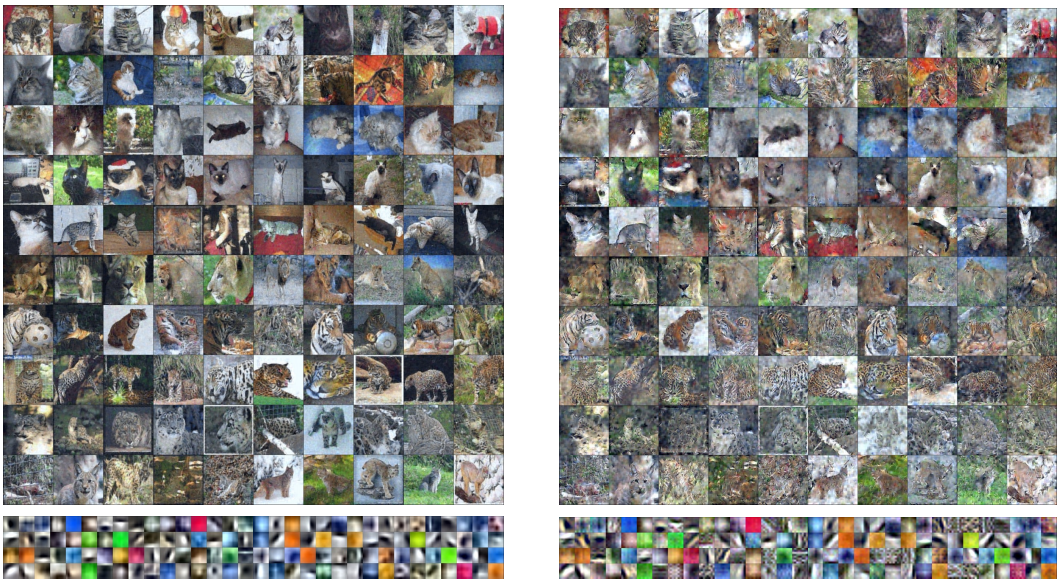
A.2.4 Qualitative Results

We provide additional qualitative results to further visualize the synthesized images generated by our proposed method across different datasets and configurations. First, Figures 5 to 10 display the distilled images optimized with TM across six different ImageNet subsets. Each figure compares the initial images and the final synthetic images, alongside their corresponding codebooks. Finally, Figures 11 to 16 provide qualitative comparison of different distillation objectives on the same ImageNet subsets. These figures visually compare the initial images with the final distilled images optimized using DM and GM, demonstrating how different matching strategies affect the visual patterns of the synthesized data.



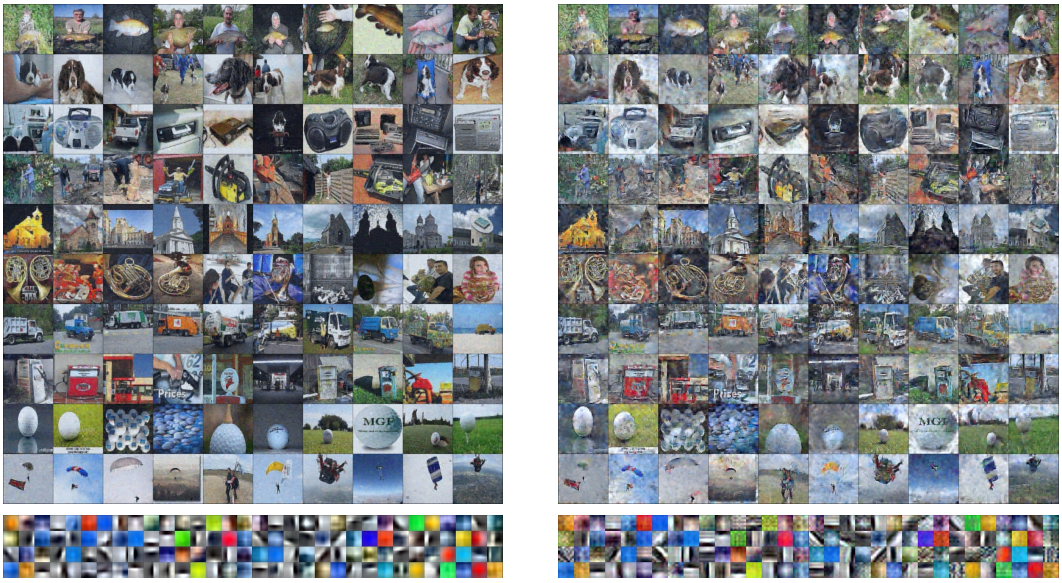
(a) (b)

Figure 5: Qualitative results on ImageFruit. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.



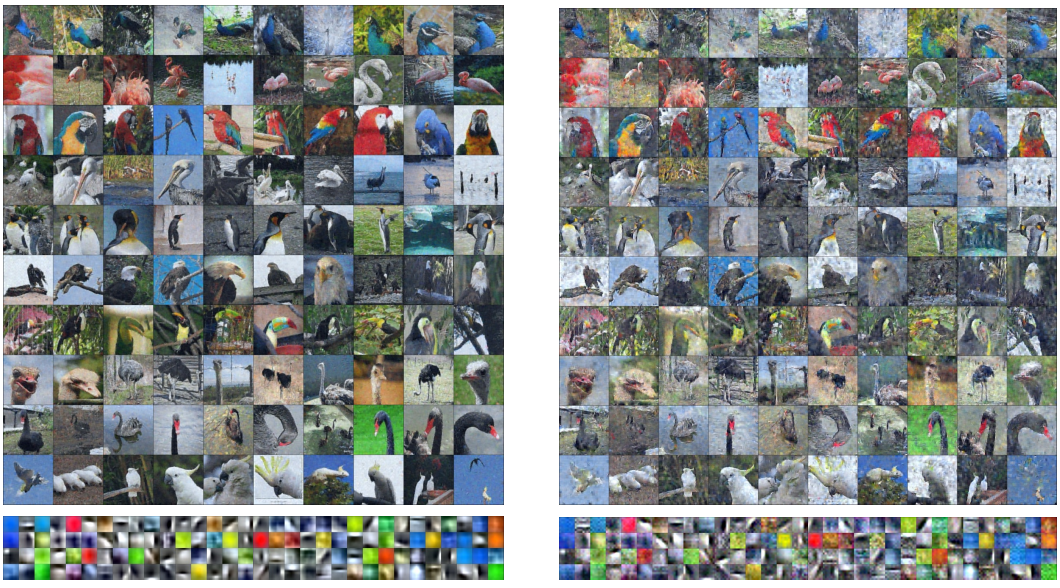
(a) (b)

Figure 6: Qualitative results on ImageMeow. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.



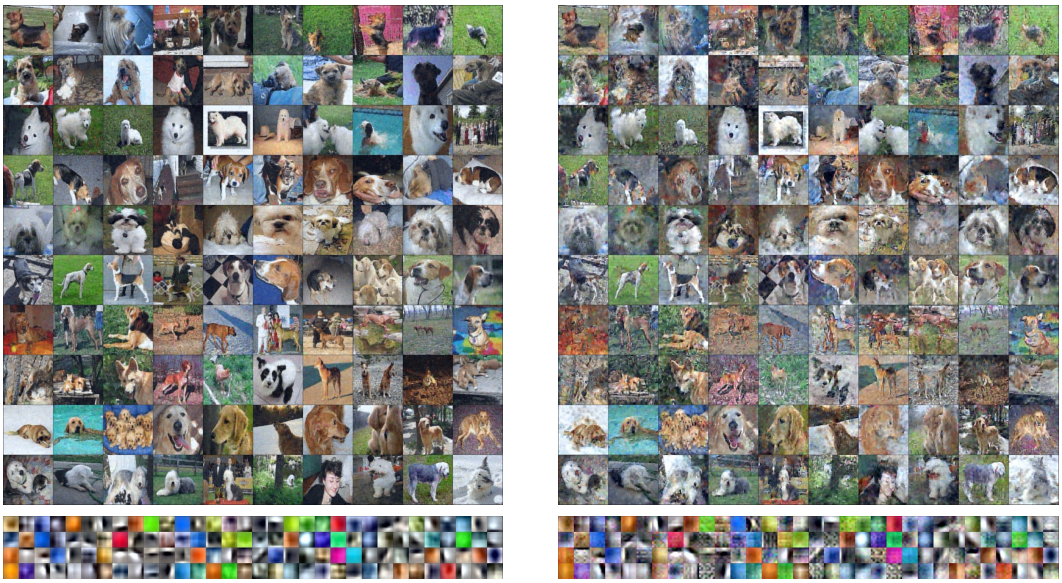
(a) (b)

Figure 7: Qualitative results on ImageNette. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.



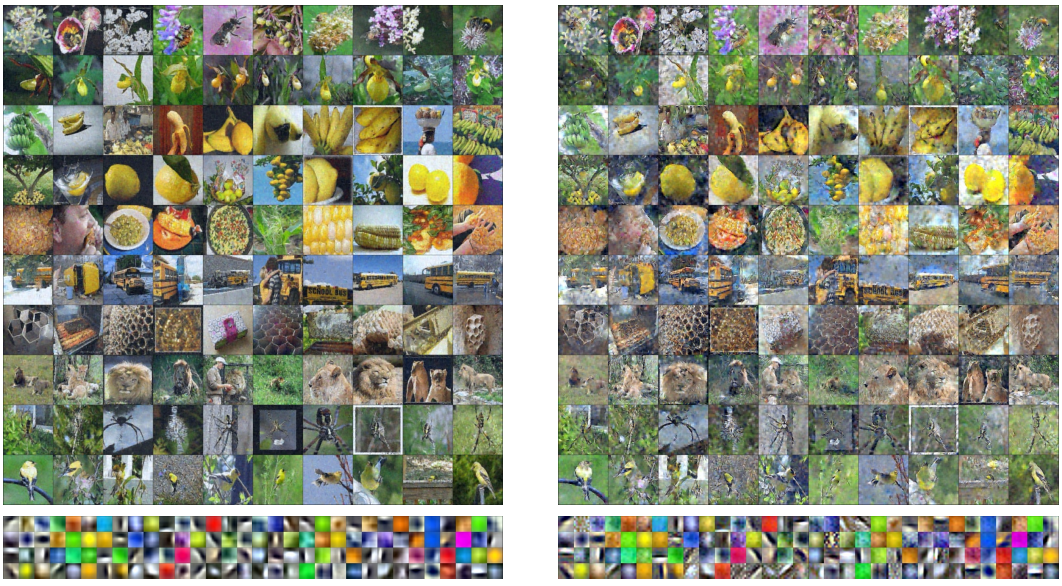
(a) (b)

Figure 8: Qualitative results on ImageSquawk. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.



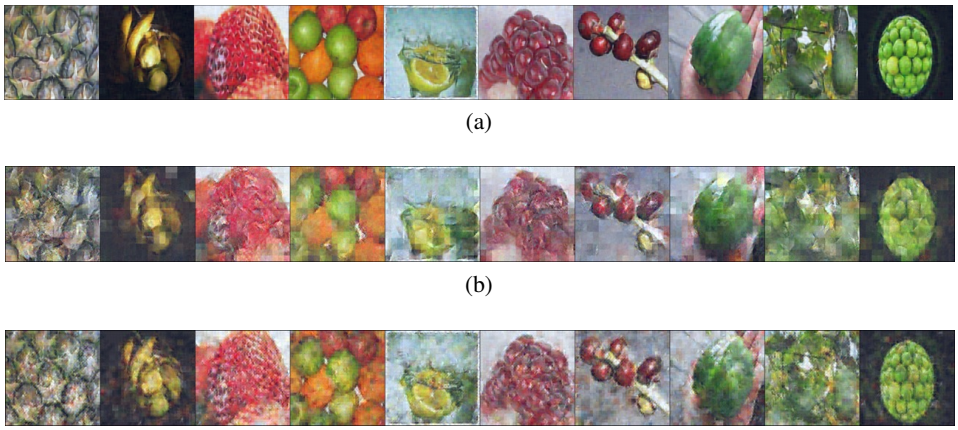
(a) (b)

Figure 9: Qualitative results on ImageWoof. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.



(a) (b)

Figure 10: Qualitative results on ImageYellow. (a) Initial images and (b) the final distilled images optimized using TM, with the codebooks (bottom). We visualize the first 10 images per class.

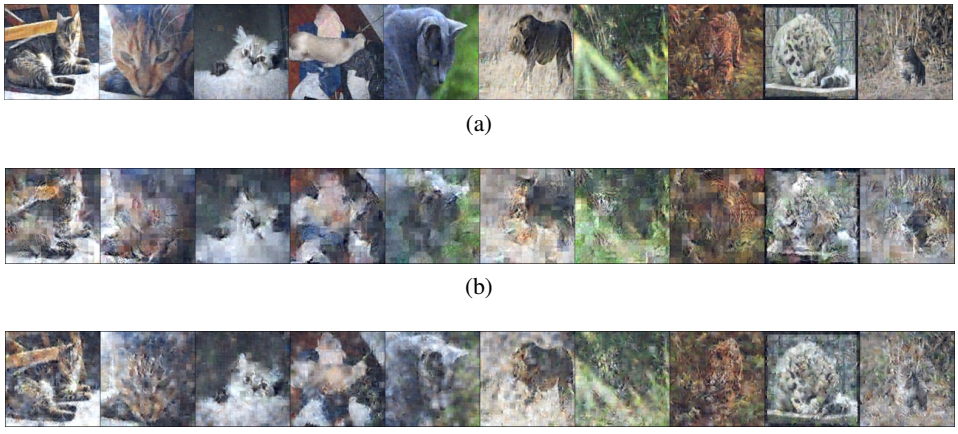


(a)

(b)

(c)

Figure 11: Qualitative comparison with different distillation objectives on ImageFruit. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.

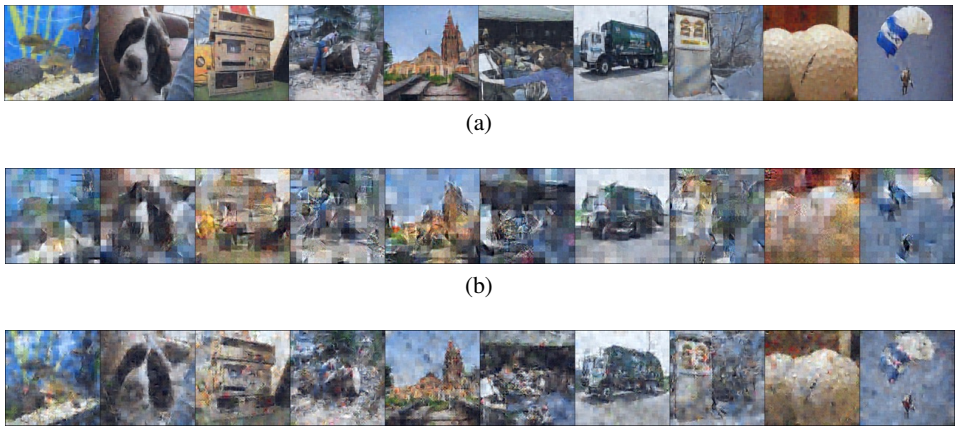


(a)

(b)

(c)

Figure 12: Qualitative comparison with different distillation objectives on ImageMeow. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.



(a)

(b)

(c)

Figure 13: Qualitative comparison with different distillation objectives on ImageNette. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.



(a)



(b)



(c)

Figure 14: Qualitative comparison with different distillation objectives on ImageSquawk. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.



(a)

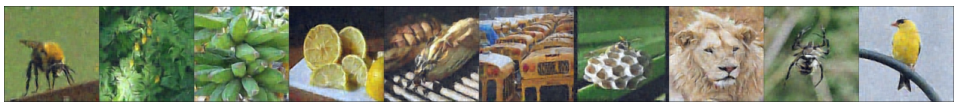


(b)



(c)

Figure 15: Qualitative comparison with different distillation objectives on ImageWoof. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.



(a)



(b)



(c)

Figure 16: Qualitative comparison with different distillation objectives on ImageYellow. (a) Initial images. The final distilled images optimized using (b) DM and (c) GM.