# Unlocking Dataset Distillation with Diffusion Models

**Brian B. Moser**[1,2,4] **Federico Raue**[2,4] **Sebastian Palacio**[3] **Stanislav Frolov**[1,2]

**Andreas Dengel**[1,2]

[1] University of Kaiserslautern-Landau, Kaiserslautern
[2] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern
[3] ABB AG, Mannheim
[4] Equal Contribution
first.second@dfki.de

## Abstract

Dataset distillation seeks to condense datasets into smaller but highly representative synthetic samples. While diffusion models now lead all generative benchmarks, current distillation methods avoid them and rely instead on GANs or autoencoders, or, at best, sampling from a fixed diffusion prior. This trend arises because naive backpropagation through the long denoising chain leads to vanishing gradients, which prevents effective synthetic sample optimization. To address this limitation, we introduce Latent Dataset Distillation with Diffusion Models (LD3M), the first method to learn gradient-based distilled latents and class embeddings end-to-end through a pre-trained latent diffusion model. A linearly decaying skip connection, injected from the initial noisy state into every reverse step, preserves the gradient signal across dozens of timesteps without requiring diffusion weight fine-tuning. Across multiple ImageNet subsets at $128 \times 128$ and $256 \times 256$, LD3M improves downstream accuracy by up to 4.8 percentage points (1 IPC) and 4.2 points (10 IPC) over the prior state-of-the-art. The code for LD3M is provided at `https://github.com/Brian-Moser/prune_and_distill`.

## 1 Introduction

Large-scale datasets fuel the advancements in modern computer vision but demand substantial computational resources and raise scalability concerns [27, 2, 25, 14]. Dataset distillation emerges as a compelling solution, aiming to synthesize a small set of information-rich samples that preserve the essence of the original dataset [35, 42, 4]. While early methods operated directly in pixel space, a promising recent direction involves leveraging powerful generative models as priors [13, 1]. By optimizing compact latent codes instead of raw pixels, these approaches, exemplified by GLaD using StyleGAN-XL [5], can generate higher-resolution synthetic images ($128 \times 128$ and beyond) that generalize better across diverse network architectures.

However, GAN-based priors like in GLaD suffer from complex multi-space latent optimization and require cumbersome inversion processes for initialization [5, 43, 3, 21]. Diffusion models [18, 28], having surpassed GANs as the state-of-the-art image generators [9], represent a natural next step. Yet, *inherent* vanishing gradients severely hinder their direct application to dataset distillation. Optimizing latents *through* the long denoising chain leads to exponentially decaying gradients, which prevents effective learning of the synthetic data [19, 17].

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Mathematically, if $\mathcal{Z}$ is the initial latent code to be optimized and $\mathbf{z}_0$ the final denoised state after $T$ steps, the gradient $\partial\mathcal{L}/\partial\mathcal{Z}$ is a product of Jacobians:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{Z}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_0} \cdot \left[\prod_{t=1}^{T} \frac{\partial \mathbf{z}_{t-1}}{\partial \mathbf{z}_t}\right] \cdot \frac{\partial \mathbf{z}_T}{\partial \mathcal{Z}}. \tag{1}$$

If the norm of each Jacobian $\partial\mathbf{z}_{t-1}/\partial\mathbf{z}_t$ is bounded by $\lambda < 1$, the product term $\prod_{t=1}^{T}(\partial\mathbf{z}_{t-1}/\partial\mathbf{z}_t)$ diminishes towards zero as $T$ increases. This gradient decay is empirically observable and significant: Our analysis confirms that in standard diffusion models, gradient norms for $\mathcal{Z}$ decrease nearly tenfold as $T$ increases from 10 to 90 (see Table 1).

Table 1: Gradient norms for the initial latent code $\mathcal{Z}$ across different maximum diffusion steps $T$ for fixed inputs. The decrease in norm as $T$ increases empirically demonstrates the vanishing gradient problem, which severely hinders the optimization of $\mathcal{Z}$ through the standard reverse diffusion process.

| $T$ | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|---|---|
| $\|\mathcal{L}/\partial\mathcal{Z}\| \times 10^4$ | 58.1 | 33.5 | 19.8 | 15.4 | 13.9 | 12.1 | 10.4 | 8.7 | 6.5 |

This critical bottleneck has forced prior diffusion-based distillation attempts to circumvent end-to-end optimization entirely, resorting instead to sampling or selecting fixed representations from pre-trained models [12, 34, 16]. These approaches, while computationally faster, forfeit the fine-grained gradient-matching optimization crucial for potential benefits like enhanced privacy of distilled data or their robustness against adversarial attacks [7, 37, 44, 10].

To unlock diffusion models for true dataset distillation, we introduce **L**atent **D**ataset **D**istillation with **D**iffusion **M**odels (LD3M). Our core contribution is a tailored modification to the reverse diffusion process (Equation 7) that introduces linearly decaying residual connections, specifically designed to enhance gradient flow for optimizing latent representations $\mathcal{Z}$ and conditioning codes $\mathbf{c}$ in the context of dataset distillation. This mechanism enables, for the first time, effective end-to-end optimization of distilled latent codes $\mathcal{Z}$ and class embeddings $\mathbf{c}$ *through* a pre-trained latent diffusion model. LD3M is readily compatible with existing distillation objectives and diffusion model architectures. Experiments across numerous ImageNet subsets demonstrate that LD3M significantly outperforms the state-of-the-art at $128 \times 128$ and $256 \times 256$ resolutions, achieving superior cross-architecture generalization, *i.e.*, 4.8 percentage points (1 IPC) and 4.2 points (10 IPC), and faster distillation times.

## 2 Preliminaries

**Dataset Distillation:** Let $\mathcal{T} = (X_r, Y_r)$, where $X_r \in \mathbb{R}^{N\times H\times W\times C}$, be a real image classification dataset and $N$ its cardinality. The goal is to compress $\mathcal{T}$ into a small synthetic set $\mathcal{S} = (X_s, Y_s)$, where $X_s \in \mathbb{R}^{M\times H\times W\times C}$, where $M$ is the total number of synthetic samples with $M = \mathcal{C}\cdot IPC$, $\mathcal{C}$ the number of classes and $IPC$ the Images Per Class (IPC). We aim to achieve $M \ll N$ with

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

where $\mathcal{L}$ is a distillation loss between the distilled set $\mathcal{S}$ and the real dataset $\mathcal{T}$. Common choices for $\mathcal{L}$ include matching gradients (Dataset Condensation, DC [42]), feature distributions (Distribution Matching, DM [41]), or model parameter trajectories (Matching Training Trajectories, MTT [4]). We refer to the supplementary material for detailed definitions.

**Dataset Distillation with Generative Priors:** To improve the quality, resolution, and generalization of distilled images, recent work incorporates deep generative models as priors [5]. Instead of optimizing pixels directly, they optimize latent codes $\mathcal{Z} \in \mathbb{R}^{M\times h\times w\times d}$ with $h\cdot w\cdot d \ll H\cdot W\cdot C$, fed into a pre-trained generator $\mathcal{D}$. The optimization objective becomes

$$\mathcal{Z}^* = \arg\min_{\mathcal{Z}} \mathcal{L}(\mathcal{D}(\mathcal{Z}), \mathcal{T}). \tag{3}$$

## 3 Related Work

**GAN Priors (GLaD):** GLaD [5] pioneered distillation with generative priors using a pre-trained StyleGAN-XL [30]. While successful, it inherits GAN-specific drawbacks: (1) Optimizing the

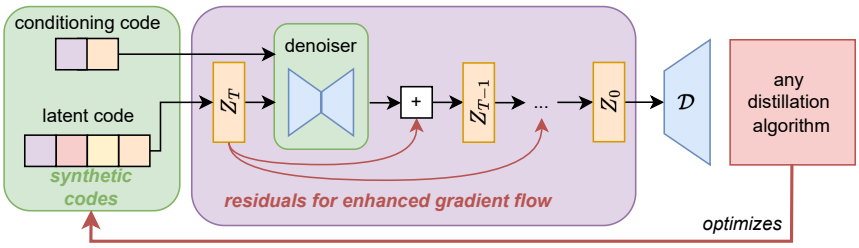


Figure 1: The LD3M Framework. Learnable latent codes $\mathcal{Z}$ and conditioning codes $\mathbf{c}$ are optimized. $\mathcal{Z}$ is noised to initialize the reverse diffusion at $\mathbf{z}_T$. A pre-trained LDM denoiser iteratively refines the state ($\mathbf{z}_t \to \mathbf{z}_{t-1}$). **Key innovation:** Residual connections (red arrows) inject $\mathbf{z}_T$ with linearly decaying weight into each step (Equation 7), enhancing gradient flow. The final latent $\mathbf{z}_0$ is decoded ($\mathcal{D}$) into images $\mathcal{S}$, which are optimized using a standard distillation algorithm.

complex, multi-level latent space ($\mathbf{W}^+$) required for high quality is computationally demanding [43]. (2) Initializing latent codes $\mathcal{Z}$ from real images $\mathbf{x}$ requires solving a costly GAN inversion problem ($\min_{\mathcal{Z}} \mathcal{L}(\mathbf{x}, \mathcal{D}(\mathcal{Z}))$) [3, 36], hindering standard initialization practices [21].

**Prior Diffusion-based Distillation Attempts:** The inherent vanishing gradient problem (§1) significantly challenges using diffusion models for end-to-end distillation. Faced with this gradient barrier, existing diffusion methods for distillation have adopted non-optimization strategies:

- **Sampling/Selection Methods:** Minimax Diffusion [16] and D4M [34] use criteria to *select* or *sample* representative latents from a diffusion model, avoiding backpropagation to latents entirely. While faster, this *bypasses gradient-based optimization that is needed for privacy or robustness*, also crucial motivations for dataset distillation [7, 37, 44, 10].
- **Autoencoder-Only Methods:** Duan et al. [12] leverage the pre-trained autoencoder from LDM but *do not utilize the diffusion process itself*. They optimize latent codes $\mathcal{Z}$ that are directly decoded by $\mathcal{D}$, essentially using only the autoencoder, not its core denoising mechanism. This simplifies optimization but fails to exploit the diffusion prior.

In contrast, LD3M is designed to directly optimize latent codes $\mathcal{Z}$ by enabling gradient flow *through* the reverse process.

**Residual Connections for Gradient Flow:** Enhancing gradient flow in deep networks is commonly addressed using residual connections, famously introduced in ResNets [17] or LSTMs [19]. Similar ideas have appeared within diffusion models; for instance, SAGE [23] used residuals connecting to the initial noise to enable adversarial latent search. While conceptually related, our contribution differs significantly: we introduce structured, *linearly decaying* residuals injected *at every step* from the *initial noisy latent* $\mathbf{z}_T$, explicitly designed for end-to-end optimization of distilled latent codes through the diffusion chain for the unique characteristics of dataset distillation.

**Decoupled Distillation Methods:** Distinct from methods optimizing synthetic data via generative priors, another line of work decouples distillation from end-to-end training. Methods like SRe2L [40] and others [39, 32] leverage statistics (e.g., from BatchNorm) of pre-trained networks to recover informative images, offering scalability benefits but following a fundamentally different optimization strategy than gradient-matching approaches like LD3M.

## 4 Latent Dataset Distillation with Diffusion Models (LD3M)

Our approach enables end-to-end optimization of synthetic data directly through a pre-trained LDM. As shown in Figure 1, LD3M optimizes both initial latent codes $\mathcal{Z}$ and conditioning codes $\mathbf{c}$. These are processed by a modified reverse diffusion process to generate expressive latent states $\mathbf{z}_0$, which are then decoded into images, $\mathcal{S} = \mathcal{D}(\mathbf{z}_0)$. This section details the core components: our modified diffusion sampling process designed to boost gradient flow (§4.1), the efficient initialization strategy (§4.2), and the gradient checkpointing used for memory efficiency (§4.3).

### 4.1 Sampling Process

We leverage a pre-trained LDM [28] without fine-tuning its weights. Standard LDM operation involves a reverse diffusion process $p_\theta$ that iteratively denoises a state $\mathbf{z}_t$, starting from noise $\mathbf{z}_T$, conditioned on an embedding $\mathbf{c}$ (typically derived from class labels) [38, 26]. Each step $t$ predicts a less noisy state $\mathbf{z}_{t-1}$ based on the current state $\mathbf{z}_t$ and condition $\mathbf{c}$. The standard update rule [18, 29] calculates the subsequent state $\mathbf{z}_{t-1}$ using the predicted mean $\mu_\theta$ and variance $\sigma_t^2$:

$$\mu_\theta(\mathbf{c}, \mathbf{z}_t, \gamma_t) = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta\left(\mathbf{c}, \mathbf{z}_t, \gamma_t\right) \right) \tag{4}$$

$$\mathbf{z}_{t-1} \leftarrow \mu_\theta(\mathbf{c}, \mathbf{z}_t, \gamma_t) + \sigma_t^2 \varepsilon_t, \tag{5}$$

where $f_\theta$ is the LDM's pre-trained noise prediction network (usually a U-Net), $\alpha_t$ and $\gamma_t$ relate to the noise schedule, and $\varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is random noise added at step $t$.

**Algorithm 1** Latent Dataset Distillation with Diffusion Models (LD3M)

**Input:** randomly selected collection $X_s$, pre-trained encoder $\mathcal{E}$, pre-trained decoder $\mathcal{D}$, pre-trained denoiser $\mu_\theta$ with frozen parameters $\theta$, noise levels $\sigma_t$.

$\mathcal{Z} = \mathcal{E}\left(X_s\right)$
$\mathbf{z}_T \sim q(\mathbf{z}_T \mid \mathcal{Z})$
**for** $t = T, \ldots, 1$ **do**
  $\varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_{t-1} \leftarrow \left((1 - \frac{t}{T}) \cdot \mu_\theta(\mathbf{c}, \mathbf{z}_t, \gamma_t) + \frac{t}{T} \cdot \mathbf{z}_T\right) + \sigma_t^2 \varepsilon_t$
**end for**
$X_{\text{syn}} \leftarrow \mathcal{D}\left(\mathbf{z}_0\right)$
**Return:** $X_{\text{syn}}$

For dataset distillation, our goal is to learn the optimal initial latent representations $\mathcal{Z}$ and conditioning codes $\mathbf{c}$ that minimize the distillation loss $\mathcal{L}$ between the synthetic images and the target dataset:

$$\mathcal{Z}^*, \mathbf{c}^* = \arg\min_{\mathcal{Z}, \mathbf{c}} \mathcal{L}(\mathcal{D}[p_\theta(\mathbf{z}_0|\mathbf{z}_T, \mathbf{c})], \mathcal{T}), \quad \text{where } \mathbf{z}_T \sim q(\mathbf{z}_T|\mathcal{Z}). \tag{6}$$

Here, $p_\theta(\mathbf{z}_0|\mathbf{z}_T, \mathbf{c})$ denotes the final state $\mathbf{z}_0$ resulting from the $T$-step reverse process starting from $\mathbf{z}_T$, which itself is a noised version of the learnable $\mathcal{Z}$ obtained via the forward process $q$. We want to highlight that the sampling of $\varepsilon$ introduces stochasticity during inference. We fix, however, for one sampling phase, the residual variable $\mathbf{z}_T$ to be constantly the same.

To effectively minimize distillation losses like gradient or trajectory matching (Equation 6), which rely on fine-grained alignment between synthetic and real data processing, requires guiding the generative process. While conditioning $\mathbf{c}$ provides class guidance, optimizing the initial latent $\mathcal{Z}$ offers the necessary degrees of freedom to shape the generated sample $\mathcal{D}\left(\mathbf{z}_0\right)$ precisely. Relying solely on fixed $\mathcal{Z}$ or only optimizing $\mathbf{c}$ proved insufficient empirically, yielding suboptimal results resembling simple LAION-5B [31] data retrieval similar to D4M [34].

Vanishing gradients inherent in backpropagating through the $T$ steps of Equation 5 present the primary obstacle to optimizing Equation 6 [19]. To overcome this, we introduce a simple yet effective modification to the reverse step, injecting a residual connection from the initial state $\mathbf{z}_T$:

$$\mathbf{z}_{t-1} \leftarrow \underbrace{\left((1 - \tfrac{t}{T}) \cdot \mu_\theta(\mathbf{c}, \mathbf{z}_t, \gamma_t) + \tfrac{t}{T} \cdot \mathbf{z}_T\right)}_{\text{Modified Mean}} + \sigma_t^2 \varepsilon_t. \tag{7}$$

Crucially, while this modification deviates from standard sampling aimed at matching the distribution of real images, its purpose here is to enable gradient flow for distillation optimization, a task where downstream utility, not photorealism [42, 4], is paramount (see supplementary material for detailed discussion). In other words, our method breaks the sampler's fidelity to the original data distribution in order to better achieve the distillation objective of matching feature distributions.

In conclusion, the modification replaces the standard predicted mean $\mu_\theta$ with a weighted average of $\mu_\theta$ and the initial state $\mathbf{z}_T$. The weight of $\mathbf{z}_T$ decreases linearly from 1 (at $t = T$) to nearly 0 (as $t \to 0$). This creates a direct pathway for gradients from the loss $\mathcal{L}$ (computed using $\mathbf{z}_0$) back to $\mathbf{z}_T$,

and thus to the learnable $\mathcal{Z}$, bypassing the long chain of Jacobian products that causes gradients to vanish. The enhanced gradient flow to $\mathcal{Z}$ can be conceptually represented as:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{Z}} = \sum_{t=1}^{T} \underbrace{\left(1 - \frac{T-1}{T}\right) \cdot \left[\frac{\partial \mathcal{L}}{\partial \mathbf{z}_t} \cdot \frac{\partial \mathbf{z}_t}{\partial \mathbf{z}_{t-1}} \cdot \ldots \cdot \frac{\partial \mathbf{z}_0}{\partial \mathcal{Z}}\right]}_{\text{Original (Decaying) Path}} + \underbrace{\left(\tfrac{t}{T}\right) \cdot \left[\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{t-1}} \frac{\partial \mathbf{z}_{t-1}}{\partial \mathbf{z}_T} \frac{\partial \mathbf{z}_T}{\partial \mathcal{Z}}\right]}_{\text{Enhanced Path via Skip Connection}}. \tag{8}$$

A comprehensive description can be found in Algorithm 1.

**Notes on Markovian Property:** $\mathbf{z}_{t-1}$ depends on $\mathbf{z}_t$ and $\mathbf{z}_T$, but not on any earlier states such as $\mathbf{z}_{t+1}$, $\mathbf{z}_{t+2}$, and so on. Therefore, the probability distribution for $\mathbf{z}_{t-1}$ only depends on $\mathbf{z}_t$ and the fixed initial state $\mathbf{z}_T$, which is constant throughout the diffusion process. Thus, we have: $p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_{t+1}, \ldots, \mathbf{z}_T) = p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_T)$. This confirms that LD3M remains Markovian.

**Notes on Generalisability:** Our gradient enhancement technique (Equation 7) applies to various diffusion model architectures beyond LDMs, suggesting broader potential for future work. In this study, we focused on LDMs primarily as a proof of concept, leveraging readily available pre-trained models and LDM's foundational role in latent-space diffusion.

### 4.2 Efficient Latent Code Initialization

Standard practice in dataset distillation initializes synthetic data using real images from the target classes [21]. GAN-based methods like GLaD face a significant challenge here, requiring complex and costly GAN inversion techniques to find latent codes $\mathcal{Z}$ that reconstruct target real images $\mathbf{x}$ [36].

LD3M benefits immensely from the autoencoder structure inherent in LDMs. We initialize the learnable latent codes $\mathcal{Z}_{\text{init}}$ simply by encoding a small, randomly selected set of real images $X_s$ using the pre-trained image encoder. Similarly, the initial conditioning codes $\mathbf{c}_{\text{init}}$ are obtained using the pre-trained class embedding network.

### 4.3 Memory Efficiency via Gradient Checkpointing

Optimizing through the $T$ steps of the reverse diffusion process, even with our modification, can be memory-intensive. Following GLaD [5], we employ gradient checkpointing [6] to manage VRAM usage. The procedure involves:

1. Perform the forward pass $\mathcal{S} = \mathcal{D}(p_\theta(\mathbf{z}_0|\mathbf{z}_T, \mathbf{c}))$ *without* storing intermediate activations.
2. Calculate the distillation loss $\mathcal{L}(\mathcal{S}, \mathcal{T})$ and the gradient with respect to the output, $\partial\mathcal{L}/\partial\mathcal{S}$.
3. To compute the gradient $\partial\mathcal{L}/\partial\mathcal{Z}$ (and $\partial\mathcal{L}/\partial\mathbf{c}$), recompute the necessary segments of the forward pass through the diffusion process and decoder $\mathcal{D}$, storing only the activations needed for the immediate backward pass segment.

This avoids storing the full computation graph, which GLaD also exploits for a single generator pass.

## 5 Experiments

We evaluate LD3M against relevant baselines, primarily the state-of-the-art latent generative prior method GLaD [5], following its experimental setup for fair comparison. We conduct extensive experiments on 10 diverse 10-class subsets of ImageNet-1k [8] at $128 \times 128$ (IPC=1, IPC=10) and $256 \times 256$ (IPC=1) resolutions, as well as CIFAR-10. Key implementation details are in §5.1; full hyperparameters and setup details are in the supplementary material.

### 5.1 Setup Details

**Datasets & Evaluation:** We use ImageNet subsets (ImNet-A to E, ImNette, ImWoof, Birds, Fruits, Cats) from [5, 20, 4] and CIFAR-10. Following standard protocol, we distill datasets using DC [42], DM [41], or MTT [4] and evaluate by training unseen architectures (AlexNet [22], VGG-11 [33], ResNet-18 [17], ViT [11]) from scratch on the distilled set, reporting mean test accuracy ($\pm$ std. dev.) over 5 runs.

Table 2: CIFAR-10 comparisons with GLaD on IPC=1 and SRe2L/D4M on IPC=50.

(a) IPC=1

| Dist. | Method | AlexNet | ResNet18 | VGG11 | ViT | Average |
|---|---|---|---|---|---|---|
| DC | pixel space | 25.9±0.2 | 27.3±0.5 | 28.0±0.5 | 22.9±0.3 | 26.0±0.4 |
| | GLaD (rand G) | **30.1**±0.5 | 27.3±0.2 | 28.0±0.9 | 21.2±0.6 | 26.6±0.5 |
| | GLaD (trained G) | 26.0±0.7 | **27.6**±0.6 | 28.2±0.6 | 23.4±0.2 | 26.3±0.5 |
| | **LD3M** (trained G) | 27.2±0.8 | 26.6±0.9 | **31.5**±0.3 | **29.0**±0.2 | **28.6**±0.6 |
| DM | pixel space | 22.9±0.2 | 22.2±0.7 | 23.8±0.5 | 21.3±0.5 | 22.6±0.5 |
| | GLaD (rand G) | 23.7±0.3 | 21.7±1.0 | 24.3±0.8 | 21.4±0.2 | 22.8±0.6 |
| | GLaD (trained G) | 25.1±0.5 | 22.5±0.7 | 24.8±0.8 | 23.0±0.1 | 23.8±0.5 |
| | **LD3M** (trained G) | **27.2**±0.4 | **23.0**±0.7 | **25.4**±0.4 | **23.8**±0.3 | **24.9**±0.5 |

(b) IPC=50

| Method | ConvNet |
|---|---|
| SRe2L [40] | 60.2† |
| D4M [34] | 72.8* |
| **LD3M (Ours)** | **73.2** |

† Reported [40] at IPC=1K; Decoupled method.
* Reported [34]; Sampling-based Diffusion.

Table 3: Cross-architecture performance (%) with 1 IPC on ImageNet subsets (128 × 128). LD3M (**bold**) generally achieves the best results within each algorithm and overall best (blue) compared to Pixel Space and GLaD. For instance, LD3M improves DC, MTT, and DM by +3.76%, +5.68%, and +16.34%, respectively.

| Distil. Space | Alg. | All | ImNet-A | ImNet-B | ImNet-C | ImNet-D | ImNet-E | ImNette | ImWoof | ImNet-Birds | ImNet-Fruits | ImNet-Cats |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pixel space | MTT | 25.4±2.3 | 33.4±1.5 | 34.0±3.4 | 31.4±3.4 | 27.7±2.7 | 24.9±1.8 | 24.1±1.8 | 16.0±1.2 | 25.5±3.0 | 18.3±2.3 | 18.7±1.5 |
| | DC | 27.9±1.6 | 38.7±4.2 | 38.7±1.0 | 33.3±1.9 | 26.4±1.1 | 27.4±0.9 | 28.2±1.4 | 17.4±1.2 | 28.5±1.4 | 20.4±1.5 | 19.8±0.9 |
| | DM | 19.2±1.0 | 27.2±1.2 | 24.4±1.1 | 23.0±1.4 | 18.4±0.7 | 17.7±0.9 | 20.6±0.7 | 14.5±0.9 | 17.8±0.8 | 14.5±1.1 | 14.0±1.1 |
| GLaD | MTT | 29.0±1.2 | 39.9±1.2 | 39.4±1.3 | **34.9±1.1** | 30.4±1.5 | 29.0±1.1 | 30.4±1.5 | 17.1±1.1 | 28.2±1.1 | 21.1±1.2 | 19.6±1.2 |
| | DC | 29.8±1.3 | 41.8±1.7 | **42.1±1.2** | 35.8±1.4 | 28.0±0.8 | 29.3±1.3 | 31.0±1.6 | 17.8±1.1 | 29.1±1.0 | 22.3±1.6 | 21.2±1.4 |
| | DM | 22.4±1.4 | 31.6±1.4 | 31.3±3.9 | 26.9±1.2 | 21.5±1.0 | 20.4±0.8 | 21.9±1.1 | 15.2±0.9 | 18.2±1.0 | 20.4±1.6 | 16.1±0.7 |
| **LD3M** | MTT | **30.4**±1.3 | **40.9**±1.1 | **41.6**±1.7 | 34.1±1.7 | **31.5**±1.2 | **30.1**±1.3 | **32.0**±1.3 | **19.9**±1.2 | **30.4**±1.5 | **21.4**±1.1 | **22.1**±1.0 |
| | DC | **30.9**±1.2 | **42.3**±1.3 | 42.0±1.1 | **37.1**±1.8 | **29.7**±1.3 | **31.4**±1.1 | **32.9**±1.2 | **18.9**±0.6 | **30.2**±1.4 | **22.6**±1.3 | **21.7**±0.8 |
| | DM | **25.9**±1.2 | **35.8**±1.1 | **34.1**±1.0 | **30.3**±1.2 | **24.7**±1.0 | **24.5**±0.9 | **26.8**±1.7 | **18.1**±0.7 | **23.0**±1.8 | **24.5**±1.9 | **17.0**±1.1 |

Table 4: Cross-architecture performance (%) with 10 IPC on ImageNet A-E (128 × 128). LD3M (**bold**, blue) consistently outperforms Pixel Space and GLaD with, for instance, an improvement of +2.52% and +3.46% with DC and DM, respectively.

| Distil. Space | Alg. | All | ImNet-A | ImNet-B | ImNet-C | ImNet-D | ImNet-E |
|---|---|---|---|---|---|---|---|
| pixel space | DC | 42.3±3.5 | 52.3±0.7 | 45.1±8.3 | 40.1±7.6 | 36.1±0.4 | 38.1±0.4 |
| | DM | 44.4±0.5 | 52.6±0.4 | 50.6±0.5 | 47.5±0.7 | 35.4±0.4 | 36.0±0.5 |
| GLaD | DC | 45.9±1.0 | 53.1±1.4 | 50.1±0.6 | 48.9±1.1 | 38.9±1.0 | 38.4±0.7 |
| | DM | 45.8±0.6 | 52.8±1.0 | 51.3±0.6 | **49.7±0.4** | 36.4±0.4 | 38.6±0.7 |
| **LD3M** | DC | **47.1±1.2** | **55.2±1.0** | **51.8±1.4** | **49.9±1.3** | **39.5±1.0** | **39.0±1.3** |
| | DM | **47.3±2.1** | **57.0±1.3** | **52.3±1.1** | 48.2±4.9 | **39.5±1.5** | **39.4±1.8** |

**Models:** We use ConvNet-5/ConvNet-6 [15] for distillation. As in GLaD [5], we use AlexNet [22], VGG-11 [33], ResNet-18 [17], and a Vision Transformer [11] for evaluating the distilled dataset quality for unseen architectures. For LD3M, we use the public ImageNet pre-trained LDM [28] with its 2× compression autoencoder, without fine-tuning. Default diffusion steps $T = 10$ ($128^2$) or $T = 20$ ($256^2$) are used unless specified. For GLaD comparison, the ImageNet pre-trained StyleGAN-XL [30] is used.

### 5.2 Results

**Main Results: LD3M Outperforms State-of-the-Art.** We first establish LD3M's effectiveness against the primary baseline GLaD and other state-of-the-art methods on CIFAR-10 (Table 2). At IPC=1, LD3M matches or exceeds GLaD using DC/DM distillation. The sampling-based D4M [34] delivered ≈10% accuracy for all tested models. At IPC=50, LD3M surpasses reported results from D4M and decoupled SRe2L [40] with MTT (though optimization and IPC differ, see caption). We subsequently focus our main analysis on GLaD [5] as the leading state-of-the-art baseline for latent generative prior gradient-matching distillation.

Moving to the more challenging ImageNet subsets, LD3M consistently outperforms GLaD across resolutions and IPC values. For IPC=1 at 128 × 128 (Table 3), LD3M achieves the best overall

Table 5: Cross-architecture performance (%) for $256 \times 256$ images (DC, IPC=1) using different generator initializations (ImageNet, FFHQ, Random). LD3M outperforms GLaD across initializations. Best results marked **bold**, top 3 blue. For example, LD3M improves by roughly +2 p.p. on average, whereas it improves the performance by roughly +7 p.p. compared to pixel space.

| Distil. Space | All | ImNet-A | ImNet-B | ImNet-C | ImNet-D | ImNet-E |
|---|---|---|---|---|---|---|
| pixel space | 29.5±3.1 | 38.3±4.7 | 32.8±4.1 | 27.6±3.3 | 25.5±1.2 | 23.5±2.4 |
| GLaD (ImageNet) | 34.4±2.6 | 37.4±5.5 | 41.5±1.2 | 35.7±4.0 | 27.9±1.0 | 29.3±1.2 |
| GLaD (Random) | 34.5±1.6 | 39.3±2.0 | 40.3±1.7 | 35.0±1.7 | 27.9±1.4 | 29.8±1.4 |
| GLaD (FFHQ) | 34.0±2.1 | 38.3±5.2 | 40.2±1.1 | 34.9±1.1 | 27.2±0.9 | 29.4±2.1 |
| **LD3M** (ImageNet) | 36.3±1.6 | **42.1**±2.2 | **42.1**±1.5 | 35.7±1.7 | 30.5±1.4 | 30.9±1.2 |
| **LD3M** (Random) | **36.5**±1.6 | 42.0±2.0 | 41.9±1.7 | **37.1**±1.4 | 30.5±1.5 | **31.1**±1.4 |
| **LD3M** (FFHQ) | 36.3±1.5 | 42.0±1.6 | 41.9±1.6 | 36.5±2.2 | **30.5**±0.9 | 30.6±1.1 |

Table 6: Ablation: Impact of Diffusion (DC, IPC=1). Comparing LD3M (w/ diffusion) against GLaD and LD3M using only the autoencoder (w/o diffusion). Full LD3M consistently outperforms both.

| Method | All | ImNet-A | ImNet-B | ImNet-C | ImNet-D | ImNet-E |
|---|---|---|---|---|---|---|
| GLaD | 35.4±1.3 | 41.8±1.7 | **42.1**±1.2 | 35.8±1.4 | 28.0±0.8 | 29.3±1.3 |
| LD3M (w/o diffusion) | 35.3±1.3 | 40.6±1.9 | 41.9±1.1 | 35.3±1.0 | 29.4±1.4 | 29.5±1.3 |
| LD3M (w/ diffusion) | **36.5**±1.3 | **42.3**±1.3 | 42.0±1.1 | **37.1**±1.8 | **29.7**±1.3 | **31.4**±1.1 |
| | +1.2+0.0 | +1.7-0.6 | +0.1+0.0 | +1.8+0.8 | +0.3-0.1 | +1.9-0.2 |

average accuracy in 9/10 subsets, significantly boosting performance particularly for DM (+16.3% avg.). This advantage persists at IPC=10 (Table 4), where LD3M improves average accuracy over GLaD by +2.5% (DC) and +3.5% (DM), reaching over 47% overall. The performance gains extend to $256 \times 256$ resolution (Table 5), where LD3M surpasses GLaD by +6% on average, even when comparing differently pre-trained generators (ImageNet, FFHQ, random), highlighting the robustness of leveraging the LDM prior via LD3M.

**Ablation Studies: Why LD3M Works.** The diffusion process itself is crucial for LD3M's improved performance over autoencoder-only approaches. Table 6 provides empirical evidence: removing the diffusion steps ("w/o diffusion", akin to *Duan et al.* [12]) results in performance comparable to GLaD (35.3% avg.), which is significantly lower than full LD3M utilizing the diffusion process (36.5% avg.). This +1.2 percentage points highlight that simply using the LDM's autoencoder is insufficient; successfully optimizing *through* the reverse process is necessary to unlock the performance benefits observed, validating our core approach over simpler AE-only methods. Naturally, we can expect similar limitations with few- or one-step diffusion models *in the context of dataset distillation*.

Furthermore, both learnable latents and our proposed gradient enhancement drive LD3M's effectiveness. Table 7 dissects these contributions: optimizing only conditioning $\mathbf{c}$ yields poor results (15.8% avg.), but making the latent $\mathcal{Z}$ learnable provides a substantial boost (22.3% avg.), confirming the necessity of latent optimization motivated in §4.1. Critically, however, this alone does not surpass GLaD; only by subsequently adding our enhanced gradient flow (Equation 7) does LD3M achieve its SOTA performance (28.1% avg., vs. 26.6% for GLaD). This clearly demonstrates that while learnable latents offer vital degrees of freedom, they cannot be effectively optimized through standard diffusion backpropagation; the enhanced gradient flow enabled by our residual connection (Equation 7) is the key component that makes end-to-end optimization through diffusion truly effective for this task.

**Robustness and Practical Considerations.** LD3M exhibits robustness in various aspects. For instance, standard initialization using real images encoded via the LDM's encoder significantly benefits DC and DM performance compared to Gaussian noise initialization (Table 8), while being vastly simpler than GLaD's GAN inversion (§4.2). Visually (Figure 2 and Figure 3), LD3M generates abstract but class-informative images that appear more consistent across different LDM pre-training datasets (ImageNet, FFHQ, Random) compared to GLaD. Our analysis of diffusion steps $T$ (Figure 4) reveals an optimal performance/runtime trade-off around $T = 10 - 40$.

Table 7: Ablation: Different LDM variations (MTT on ImageNette and IPC=1). Tested was the distillation by making only the conditioning learnable ($\mathbf{c}$), one with also learnable latent representation ($\mathcal{Z}$), and lastly, one which incorporates our modified reverse process (Equation 7).

| Method | All | AlexNet | VGG-11 | ResNet-18 | ViT |
|---|---|---|---|---|---|
| GLaD | 26.6±1.6 | 28.7±0.3 | **29.2±1.2** | **30.8±2.9** | 17.8±1.5 |
| LDM learnable conditioning ($\mathbf{c}$) | 15.8±1.5 | 14.2±2.6 | 15.1 ±1.6 | 16.5±4.9 | 16.8±4.0 |
| + learnable latent code ($\mathcal{Z}$) | 22.3±2.0 | 22.8±2.0 | 26.3±0.9 | 23.4±3.2 | 17.5±2.0 |
| + enhanced gradient flow (Eq. 7) | **28.1±3.3** | **29.2±1.9** | **29.2±1.2** | 30.6±1.3 | **25.1±1.7** |

Table 8: Ablation: Impact of Initialization (ImageNette, IPC=1, 5K iter.). Compares initializing $\mathcal{Z}$ and $\mathbf{c}$ from Gaussian noise vs. encoded real images across different distillation algorithms.

| Dist. Method | Dist. Space | All | AlexNet | ResNet18 | VGG11 | ViT |
|---|---|---|---|---|---|---|
| MTT | Gauss. noise | 31.0±1.4 | 28.7±1.6 | 34.1±1.5 | 32.2±0.6 | 29.1±1.8 |
| | random image | **32.0±1.3** | **30.1±1.4** | **35.6±1.4** | **32.2±0.3** | **30.0±1.2** |
| DC | Gauss. noise | 13.1±2.1 | 13.1±1.5 | 11.6±1.8 | 13.8±2.2 | 13.7±2.7 |
| | random image | **32.9±2.1** | **31.6±1.3** | **30.4±0.6** | **31.8±1.2** | **37.7±1.5** |
| DM | Gauss. noise | 13.4±1.8 | 13.4±2.0 | 12.4±1.8 | 13.4±1.4 | 14.4±2.1 |
| | random image | **26.8±1.7** | **31.9±1.3** | **23.2±2.2** | **25.9±2.0** | **26.1±1.4** |

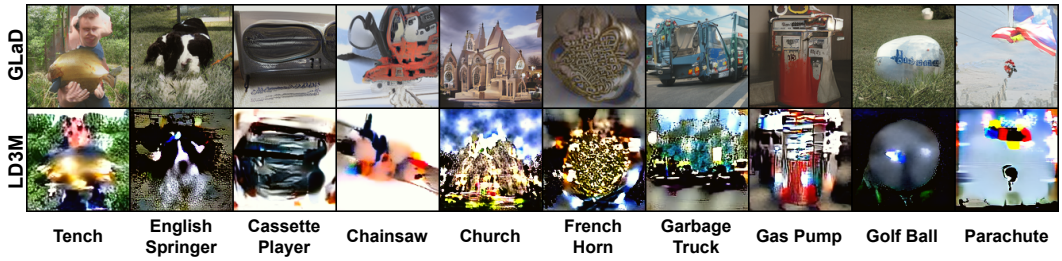


Figure 2: Visual comparison of LD3M versus GLaD (MTT, ImageNette, 1K iter.). GLaD outputs tend toward smooth, photo-realistic textures but can blur class-defining details, whereas LD3M produces bolder, higher-contrast shapes that highlight key discriminative features (*e.g.*, wing contours, beak outline). This abstraction trade-off suggests LD3M prioritizes core class signals over pixel-perfect fidelity, which empirically enhances downstream model generalization, contrasting claims made by sampling-based methods like D4M [34].

LD3M also offers computational flexibility. Our analysis indicates an optimal balance between performance and distillation time occurs with around $T = 10 - 40$ diffusion steps (Figure 4). Using $T = 20$, LD3M requires slightly less peak GPU memory on an A100-40GB compared to GLaD (29.4GB vs 31.2GB) and completes the distillation process faster (574 min vs 693 min). This computational efficiency, coupled with the inherent flexibility to adjust the number of diffusion steps ($T$) based on available resources, enhances LD3M's practical appeal.

## 6 Conclusion

We addressed the critical challenge preventing end-to-end dataset distillation through powerful diffusion models: vanishing gradients across the long denoising chain. We introduced **LD3M**, which unlocks optimization through diffusion priors via a simple yet effective modification: injecting linearly decaying residual connections from the initial noisy state into each reverse step. This novel approach enhances gradient flow sufficiently to effectively learn both latent codes ($\mathcal{Z}$) and conditioning ($\mathbf{c}$) without altering pre-trained model weights. This direct optimization contrasts sharply with previous diffusion distillation methods that circumvented the gradient challenge by relying solely on sampling fixed representations [34, 16].

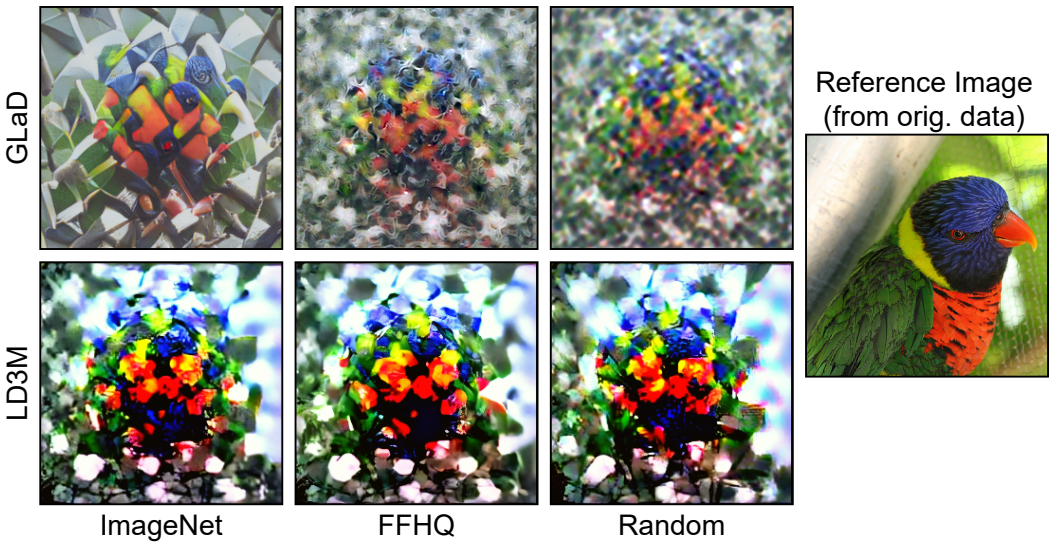


Figure 3: Example $256 \times 256$ images of a distilled class (ImageNet-B: Lorikeet) with differently initialized generators GLaD and LD3M. The various initializations, *i.e.*, which dataset was used for training the generators, are denoted at the bottom. We used DC as distillation algorithm.

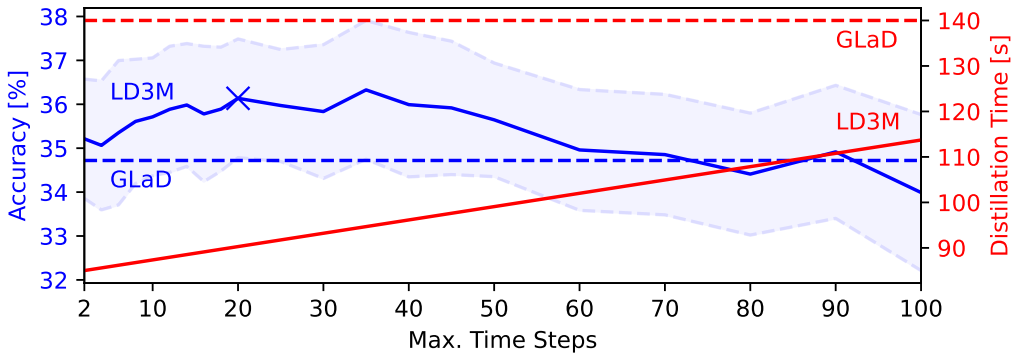


Figure 4: Accuracy vs. Distillation Time Trade-off with Diffusion Steps $T$ (ImageNet A-E avg., MTT, IPC=1). LD3M performance (blue line, mean $\pm$ std) peaks around $T = 35$. GLaD baseline (dashed lines) and optimal trade-off (X) shown for reference.

Our experiments (§5) demonstrate LD3M's clear advantages. It significantly outperforms the state-of-the-art GAN-prior method, GLaD, on diverse ImageNet subsets at $128 \times 128$ and $256 \times 256$ resolutions, improving cross-architecture generalization by up to 4.8 percentage points (IPC=1) and 4.2 points (IPC=10). Ablation studies confirmed that both leveraging the diffusion process and our specific gradient enhancement are crucial for this success. Furthermore, LD3M offers practical benefits: vastly simpler initialization than GAN inversion (§4.2) and faster overall distillation times compared to GLaD (§5).

## 7 Future Work

By successfully enabling gradient-based optimization through diffusion models, LD3M paves the way for leveraging state-of-the-art generative diffusion models for creating highly effective and compact distilled datasets. Future work should explore alternative residual formulations and integration with fast samplers like DPM-Solver [24]. Scaling and evaluation on larger benchmarks like ImageNet-1K also remain important next steps, contingent on computational resources.

## Acknowledgments

This work was supported by the EU project SustainML (Grant 101070408) and by the BMBF Project Albatross (Grant 01IW24002). All compute was done thanks to the Pegasus cluster at DFKI.

## References

[1] O. Bar-Tal, L. Yariv, Y. Lipman, and T. Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. 2023.

[2] S. Bengio, K. Dembczynski, T. Joachims, M. Kloft, and M. Varma. Extreme classification (dagstuhl seminar 18291). In *Dagstuhl Reports*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *ICLR*, 2017.

[4] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022.

[5] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Generalizing dataset distillation via deep generative prior. In *CVPR*, pages 3739–3748, 2023.

[6] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint*, 2016.

[7] Y. Chen, W. Huang, and L. Weng. Provable and efficient dataset distillation for kernel ridge regression. *NeurIPS*, 37:88739–88771, 2024.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021.

[10] T. Dong, B. Zhao, and L. Lyu. Privacy for free: How does dataset condensation help privacy? In *ICML*, pages 5378–5396. PMLR, 2022.

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

[12] Y. Duan, J. Zhang, and L. Zhang. Dataset distillation in latent space. *arXiv preprint*, 2023.

[13] S. Frolov, B. B. Moser, and A. Dengel. Spotdiffusion: A fast approach for seamless panorama generation over time. *arXiv preprint*, 2024.

[14] D. Ganguli, D. Hernandez, L. Lovitt, A. Askell, Y. Bai, A. Chen, T. Conerly, N. Dassarma, D. Drain, N. Elhage, et al. Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022.

[15] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

[16] J. Gu, S. Vahidian, V. Kungurtsev, H. Wang, W. Jiang, Y. You, and Y. Chen. Efficient dataset distillation via minimax diffusion. *arXiv preprint*, 2023.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[19] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998.

[20] J. Howard. A smaller subset of 10 easily classified classes from imagenet, and a little more french. *URL https://github.com/fastai/imagenette*, 2019.

[21] J.-H. Kim, J. Kim, S. J. Oh, S. Yun, H. Song, J. Jeong, J.-W. Ha, and H. O. Song. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*. PMLR, 2022.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25, 2012.

[23] Q. Liu, A. Kortylewski, Y. Bai, S. Bai, and A. Yuille. Discovering failure modes of text-guided diffusion models via adversarial search. *arXiv preprint*, 2023.

[24] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 2022.

[25] B. Moser, F. Raue, J. Hees, and A. Dengel. Less is more: Proxy datasets in nas approaches. In *CVPR*, 2022.

[26] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio, and A. Dengel. Diffusion models, image super-resolution and everything: A survey. *arXiv preprint*, 2024.

[27] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint*, 2022.

[28] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[29] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. Image super-resolution via iterative refinement. *IEEE TPAMI*, 2022.

[30] A. Sauer, K. Schwarz, and A. Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[31] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 35:25278–25294, 2022.

[32] S. Shao, Z. Yin, M. Zhou, X. Zhang, and Z. Shen. Generalized large-scale data condensation via various backbone and statistical matching. In *CVPR*, 2024.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2014.

[34] D. Su, J. Hou, W. Gao, Y. Tian, and B. Tang. Dˆ 4: Dataset distillation via disentangled diffusion model. In *CVPR*, pages 5809–5818, 2024.

[35] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. Dataset distillation. *arXiv preprint*, 2018.

[36] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang. Gan inversion: A survey. *IEEE TPAMI*, 2022.

[37] E. Xue, Y. Li, H. Liu, P. Wang, Y. Shen, and H. Wang. Towards adversarially robust dataset distillation by curvature regularization. In *AAAI*, volume 39, pages 9041–9049, 2025.

[38] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[39] Z. Yin and Z. Shen. Dataset distillation in large data era. *arXiv preprint*, 2023.

[40] Z. Yin, E. Xing, and Z. Shen. Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. *NeurIPS*, 2023.

[41] B. Zhao and H. Bilen. Dataset condensation with distribution matching. In *WACV*, 2023.

[42] B. Zhao, K. R. Mopuri, and H. Bilen. Dataset condensation with gradient matching. *ICLR*, 2020.

[43] X. Zhong, H. Fang, B. Chen, X. Gu, T. Dai, M. Qiu, and S.-T. Xia. Hierarchical features matter: A deep exploration of gan priors for improved dataset distillation. *arXiv preprint*, 2024.

[44] Z. Zhou, W. Feng, S. Lyu, G. Cheng, X. Huang, and Q. Zhao. Beard: Benchmarking the adversarial robustness for dataset distillation. *arXiv preprint arXiv:2411.09265*, 2024.

## NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The abstract and the Introduction section described the task of distilled dataset and our presented solution based on pre-trained Diffusion Models. Our model is evaluated on several standard datasets for Image Classification.
   Guidelines:
   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: We include a discussion on the limitations of our method in the appendix.
   Guidelines:
   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: We provide derive theoretical motivation for the sampling process in section 4.
   Guidelines:
   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**
   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
   Answer: [Yes]
   Justification: All hyperparameters are described in our paper. The presented model is clearly described for reproducibility and the algorithm is summarized in Algorithm 1.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
   Answer: [Yes]
   Justification: The datasets are standard and well-known benchmarks for Image Classification. The code will be also released after the publication of our paper.
   Guidelines:
   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not

including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?
   Answer: [Yes]
   Justification: Our experimental section described all required information such as models and datasets. We follow the same experimental setting as previous work in Distilled Dataset (GLaD). Also, the code repository will contain the training scripts for reproducing the results.
   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [Yes]
   Justification: our experiments were run five times, and it is clearly mentioned that we are reporting mean test accuracy and standard deviation.
   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
Answer: [Yes]
Justification: In the supplementary materials, we include information on the required compute resources.
Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**
Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
Answer: [Yes]
Justification: We comply with the code of ethics.
Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**
Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
Answer: [Yes]
Justification: We provide societal impact in the sections 1 and 3.
Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**
Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?
Answer: [NA]

Justification: Our method does not produce new risks of misuse, because we re-use existing models and datasets in a compressed manner.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We credit all owners of the assets that we use.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets are released.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper did not require any input for crowdsourcing experiments. We evaluated our presented model using standard benchmarks for Image Classification.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We conduct no user studies.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs as part of our method.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# Supplementary Material

## A Definitions of DC, DM, and MTT

**Dataset Condensation (DC)** ensures alignment by deriving the gradients via a classification error [42]. It calculates the loss on real ($\ell^{\mathcal{T}}$) and the respective synthetic data ($\ell^{\mathcal{S}}$). Next, it minimizes the distance between the gradients of both network instances. More concretely,

$$\mathcal{L}_{DC} = 1 - \frac{\nabla_\theta \ell^{\mathcal{S}}(\theta) \cdot \nabla_\theta \ell^{\mathcal{T}}(\theta)}{\|\nabla_\theta \ell^{\mathcal{S}}(\theta)\| \, \|\nabla_\theta \ell^{\mathcal{T}}(\theta)\|}. \tag{9}$$

**Distribution Matching (DM)** obtains gradients by minimizing the logits on the real and synthetic datasets. It enforces the feature extractor (ConvNet) to produce similar features for real and synthetic images [41]. The distribution matching loss is

$$\mathcal{L}_{DM} = \sum_c \left\| \frac{1}{|\mathcal{T}_c|} \sum_{\mathbf{x} \in \mathcal{T}_c} \psi(\mathbf{x}) - \frac{1}{|\mathcal{S}_c|} \sum_{\mathbf{s} \in \mathcal{S}_c} \psi(\mathbf{s}) \right\|^2, \tag{10}$$

where $\mathcal{T}_c, \mathcal{S}_c$ are the real and synthetic images for a class $c$.

**Matching Training Trajectories (MTT)** concentrates on the trajectory of network parameters [4]. In more detail, MTT exploits several trained instances of a model, called experts, and stores the training trajectory of parameters $\{\theta_t^*\}_0^T$ at predetermined intervals, called expert trajectories. For dataset distillation, MTT samples a random set of parameters $\theta_t^*$ from the trajectory at a given timestamp. Next, it trains a new network, $\hat{\theta}_{t+N}$, initialized with the parameters on the respective synthetic images (for $N$ iterations). Finally, the distance between the trajectory on the real dataset, $\theta_{t+M}^*$ with $M$ steps, and the trajectory on the synthetic one, $\hat{\theta}_{t+N}$, is minimized. As a result, MTT tries to mimic the original dataset's training path (trajectory of parameters) with the synthetic images:

$$\mathcal{L}_{MTT} = \frac{\|\hat{\theta}_{t+N} - \theta_{t+M}^*\|^2}{\|\theta_t^* - \theta_{t+M}^*\|^2}. \tag{11}$$

## B Justification for Modified Reverse Process in Distillation

Our core modification to the reverse diffusion step is crucial for improving the gradient flow back to the initial latent $\mathcal{Z}$, allowing end-to-end optimization. A natural question arises regarding the validity of this modified process, as it intentionally deviates from standard diffusion sampling procedures designed for high-fidelity image generation that perfectly match the learned data distribution.

**Different Objectives: Distillation vs. Faithful Sampling.** The key insight is that the objective of dataset distillation differs fundamentally from standard image generation. Distillation aims to synthesize a small set of maximally informative samples that enable efficient training of downstream models, prioritizing the encoding of essential class-discriminative features over photorealism [35, 4, 42]. Pixel-perfect adherence to the original data distribution is not necessarily required or even optimal; abstract or stylized images often yield excellent distillation performance if they capture core class characteristics effectively [5].

**Impact of Modification.** Our modification introduces a direct dependency on the initial noisy state $\mathbf{z}_T$ throughout the reverse process. While preserving the Markov property (as shown in the main paper) and the representational power of the pre-trained denoiser $f_\theta$, this change means the resulting process $p_\theta(\mathbf{z}_0|\mathbf{z}_T, \mathbf{c})$ no longer guarantees sampling exactly from the original distribution $\mu$ learned by the LDM. If applied without the corrective feedback loop of distillation optimization (e.g., for unconditional generation), this modification can lead to more abstract outputs that deviate from the expected style, as illustrated with an FFHQ-trained model in Figure 5. This deviation is expected, as the process is no longer constrained solely by the standard denoising objective. We also observe a slight reduction in sample diversity (measured by average LPIPS between generated samples: 0.386 with modification vs. 0.420 without, on ImageNette samples), likely due to the persistent influence of the fixed $\mathbf{z}_T$.

Figure 5: Influence of our modified reverse process in a classical image generation setting (unconditional FFHQ). It shows that the residual connections alter the generation process significantly, leading to abstract artifacts and the loss of coherence expected in a facial dataset: **(top)** with modification and **(bottom)** without modification.

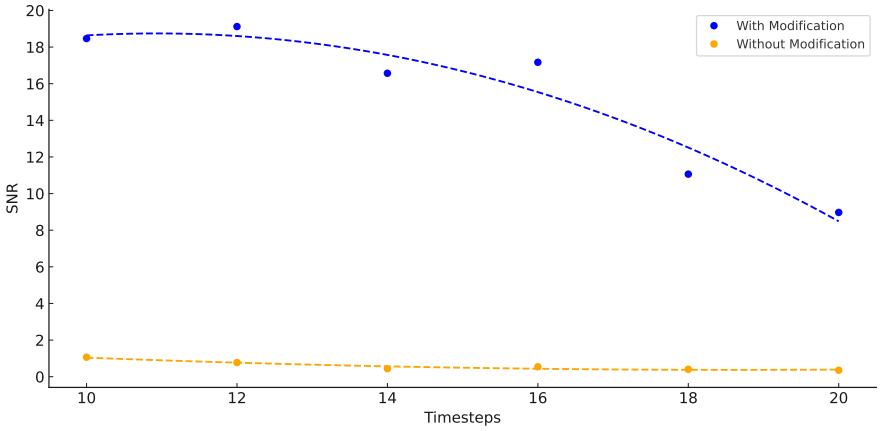


Figure 6: Gradient flow analysis comparing the Signal-to-Noise Ratio (SNR) of gradient norms for LD3M with and without our modification. Diffusion demonstrates a more stable gradient flow, indicating enhanced optimization dynamics. Dashed lines show a polyfit plot to highlight the trends.

**Suitability for Distillation.** Crucially, within the dataset distillation framework, the latent codes $\mathcal{Z}$ (and conditioning $\mathbf{c}$) are continuously optimized to minimize the distillation loss $\mathcal{L}$. This optimization process actively counteracts potential adverse effects of the modified sampling path by guiding the generation towards producing images (even if abstract) that are highly effective for the downstream task defined by $\mathcal{L}$. The essential properties needed are: (1) sufficient generative capacity to create diverse, class-relevant features, which the LDM provides; and (2) strong gradient flow for optimization, which our modification enables (as also shown in Figure 6). The empirical success of LD3M - significantly outperforming GLaD and AE-only baselines, and performing robustly even with randomly initialized LDMs - demonstrates that this trade-off (sacrificing perfect distribution matching for tractable optimization) is highly beneficial for the specific goal of dataset distillation. The resulting "abstract" representations effectively encode class information for robust generalization. Therefore, while distinct from standard sampling, our modified reverse process is a well-justified and necessary component for unlocking diffusion models for effective, end-to-end dataset distillation.

Table 9: Common hyperparameters for training the distillation algorithms used in this work.

| Parameter | Value |
|---|---|
| DSA Augmentations | Color / Crop / Cutout / Flip / Scale / Rotate |
| Iteration (Distillation) | 5,000 ($128 \times 128$) / 10,000 ($256 \times 256$) |
| Momentum | 0.5 |
| Batch Real | 256 |
| Batch Train | 256 |
| Batch Test | 128 |

# C Hyper-Parameters for Distillation Algorithms

**LDM.** For all our LDM experiments, we set the unconditional guidance scale to the default value of 3. For $128 \times 128$ images, we used max. time steps of 10, and for $256 \times 256$ images, we used 20.

**DC.** We utilize a learning rate of $10^{-3}$ throughout our DC experiments to update the latent code representation and the conditioning information.

**DM.** In every DM experiment, we adopt a learning rate of $10^{-2}$, applying it to updates of the latent code representation alongside the conditioning information.

**MTT.** For MTT experiments, a uniform learning rate of $10$ is applied to update the latent code representation and the conditioning information. We buffered 100 trajectories for expert training, each with 15 training epochs. We used ConvNet-5 and InstanceNorm. During dataset distillation, we used three expert epochs, max. start epoch of 5 and 20 synthetic steps.

# D Large Scale Datasets

Although LD3M is compatible with various distillation algorithms -including DC, DM, and MTT -our current experiments focus on baseline variants that do not leverage inter-class relationships during optimization. This is an essential avenue for further improvement: incorporating inter-class information (e.g., through contrastive losses or hierarchical label structures) may enhance the discriminative quality of the synthetic data. Future work will explore how LD3M's expressive latent trajectories can be used to facilitate such structured, cross-class-aware distillation.

# E Limitations

While LD3M improves dataset distillation compared to GLaD, it is essential to acknowledge certain limitations. A primary concern arises from the linear addition in the diffusion process, which may not sufficiently combat the vanishing gradient problem for larger time steps, as observed in our experiments [19]. Further alternative strategies for integrating the initial state $\mathbf{z}_T$ in the diffusion process should be evaluated to address this issue, *e.g.*, non-linear progress towards 0 as $t$ approaches 0. These alternative approaches could offer more nuanced and dynamic ways to manage the influence of $\mathbf{z}_T$ across different stages of the diffusion, potentially mitigating the problem of vanishing gradients and enhancing the overall efficacy of the distillation process.

# F Hardware and Software

All experiments were run on a workstation equipped with an NVIDIA RTX A6000 GPU (48 GB VRAM). Our implementation uses PyTorch 1.10.1 with torchvision 0.11.2, and we build upon the GLaD library for dataset distillation with a generative prior.

Table 10: Class listings for our ImageNet subsets.

| Dataset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| ImageNet-A | Leonberg | Probiscis Monkey | Rapeseed | Three-Toed Sloth | Cliff Dwelling | Yellow Lady's Slipper | Hamster | Gondola | Orca | Limpkin |
| ImageNet-B | Spoonbill | Website | Lorikeet | Hyena | Earthstar | Trollybus | Echidna | Pomeranian | Odometer | Ruddy Turnstone |
| ImageNet-C | Freight Car | Hummingbird | Fireboat | Disk Brake | Bee Eater | Rock Beauty | Lion | European Gallinule | Cabbage Butterfly | Goldfinch |
| ImageNet-D | Ostrich | Samoyed | Snowbird | Brabancon Griffon | Chickadee | Sorrel | Admiral | Great Gray Owl | Hornbill | Ringlet |
| ImageNet-E | Spindle | Toucan | Black Swan | King Penguin | Potter's Wheel | Photocopier | Screw | Tarantula | Sscilloscope | Lycaenid |
| ImageNette | Tench | English Springer | Cassette Player | Chainsaw | Church | French Horn | Garbage Truck | Gas Pump | Golf Ball | Parachute |
| ImageWoof | Australian Terrier | Border Terrier | Samoyed | Beagle | Shih-Tzu | English Foxhound | Rhodesian Ridgeback | Dingo | Golden Retriever | English Sheepdog |
| ImageNet-Birds | Peacock | Flamingo | Macaw | Pelican | King Penguin | Bald Eagle | Toucan | Ostrich | Black Swan | Cockatoo |
| ImageNet-Fruits | Pineapple | Banana | Strawberry | Orange | Lemon | Pomegranate | Fig | Bell Pepper | Cucumber | Granny Smith Apple |
| ImageNet-Cats | Tabby Cat | Bengal Cat | Persian Cat | Siamese Cat | Egyptian Cat | Lion | Tiger | Jaguar | Snow Leopard | Lynx |

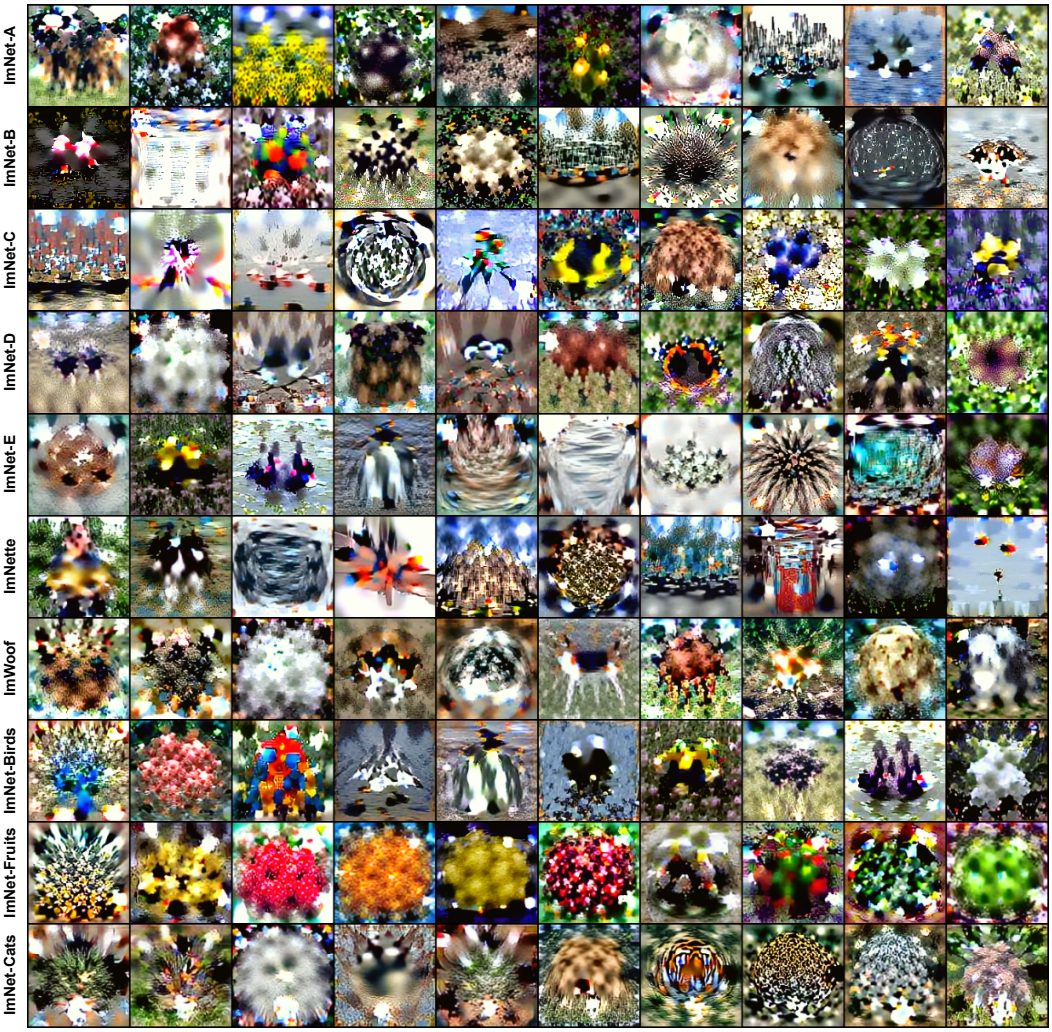


Figure 7: Images distilled by MTT in LD3M for IPC=1.


Figure 8: Images distilled by DC in LD3M for IPC=1.

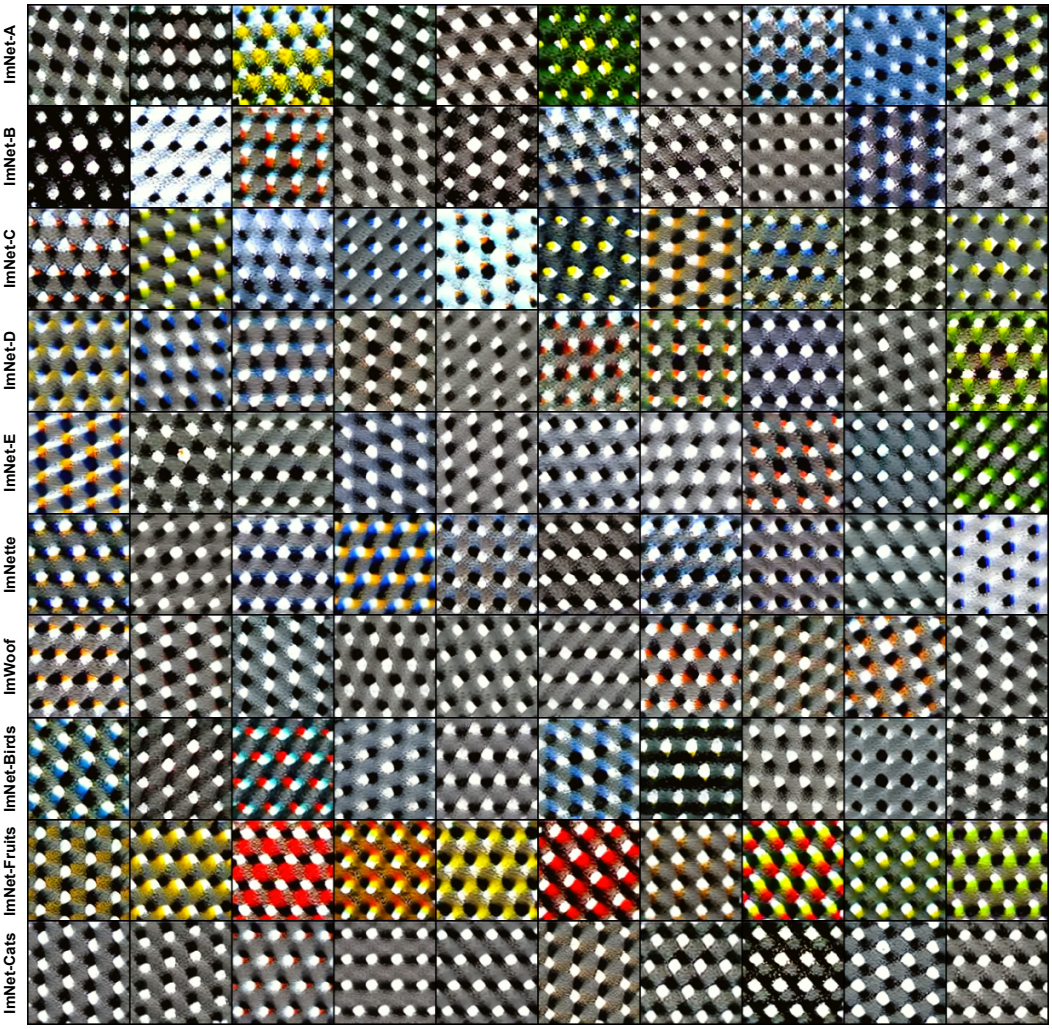


Figure 9: Images distilled by DM in LD3M for IPC=1.

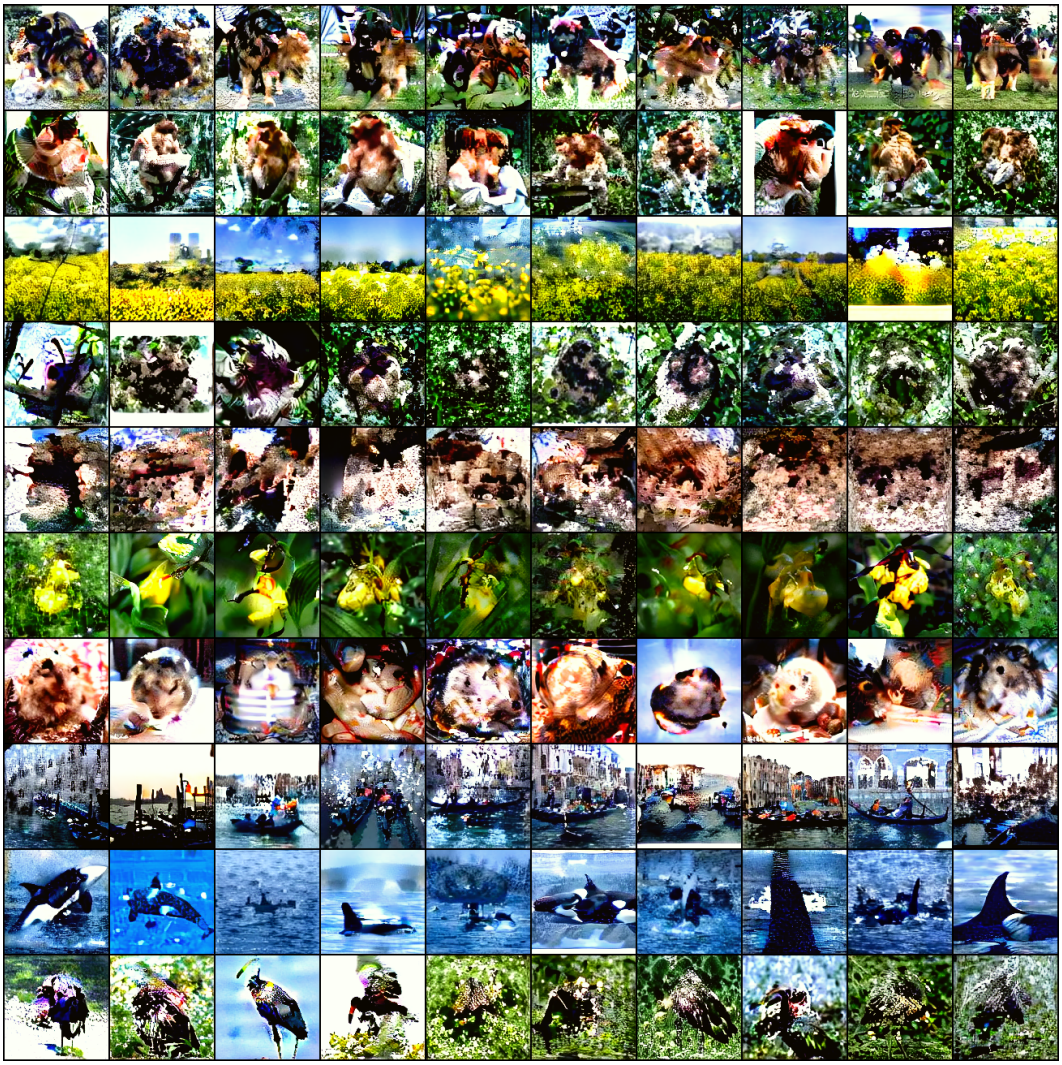

Figure 10: Images distilled by DC in LD3M for IPC=10 and ImageNet-A.

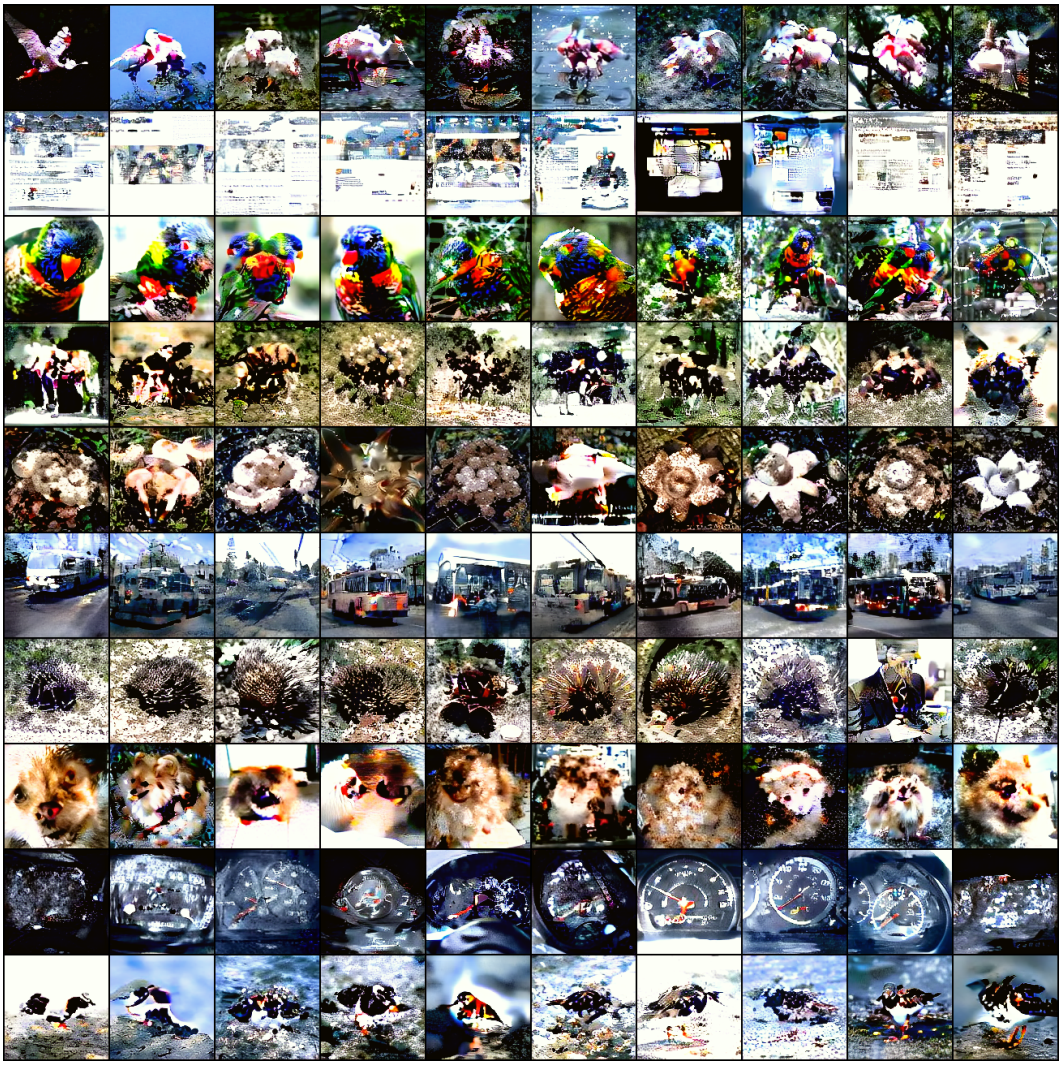

Figure 11: Images distilled by DC in LD3M for IPC=10 and ImageNet-B.

Figure 12: Images distilled by DC in LD3M for IPC=10 and ImageNet-C.

Figure 13: Images distilled by DC in LD3M for IPC=10 and ImageNet-D.

Figure 14: Images distilled by DC in LD3M for IPC=10 and ImageNet-E.

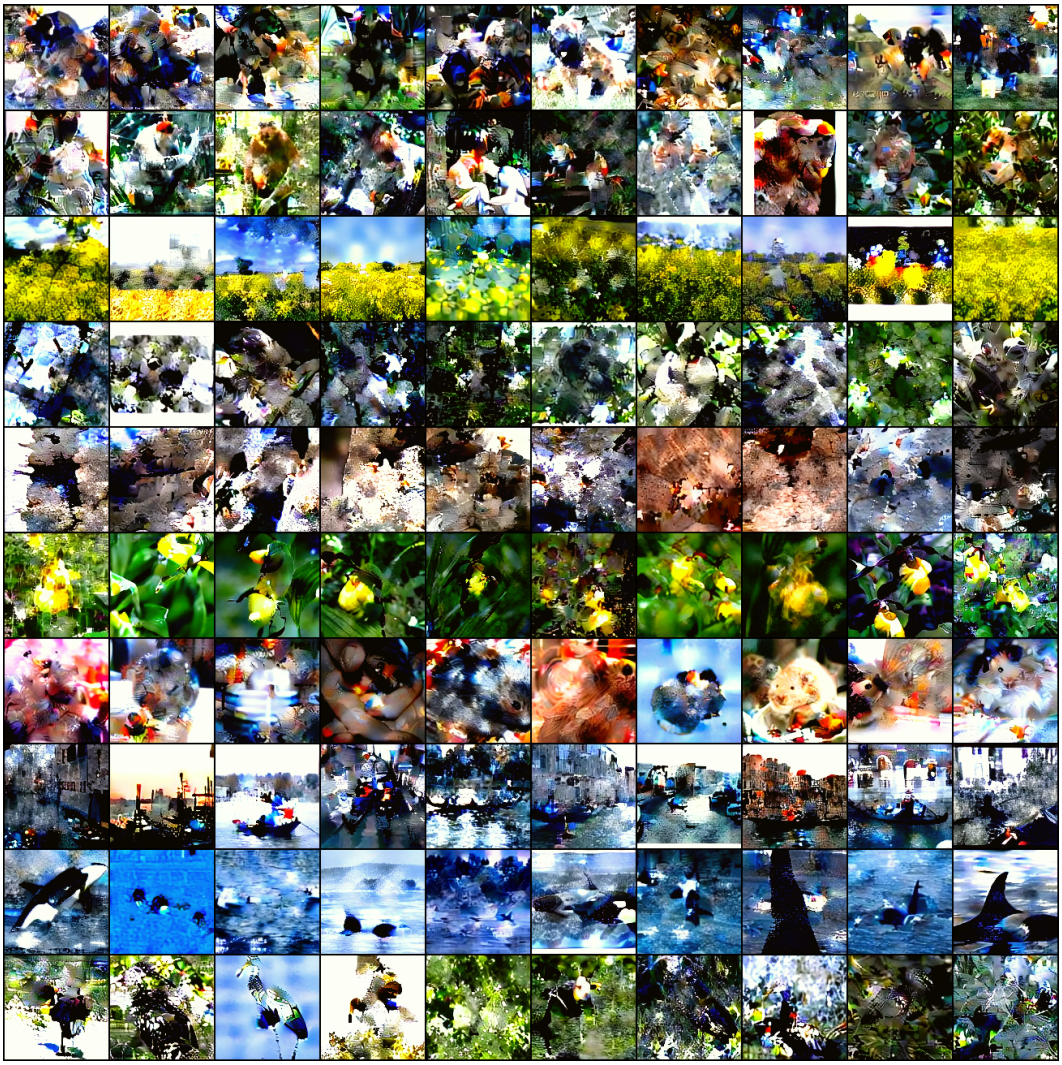

Figure 15: Images distilled by DM in LD3M for IPC=10 and ImageNet-A.

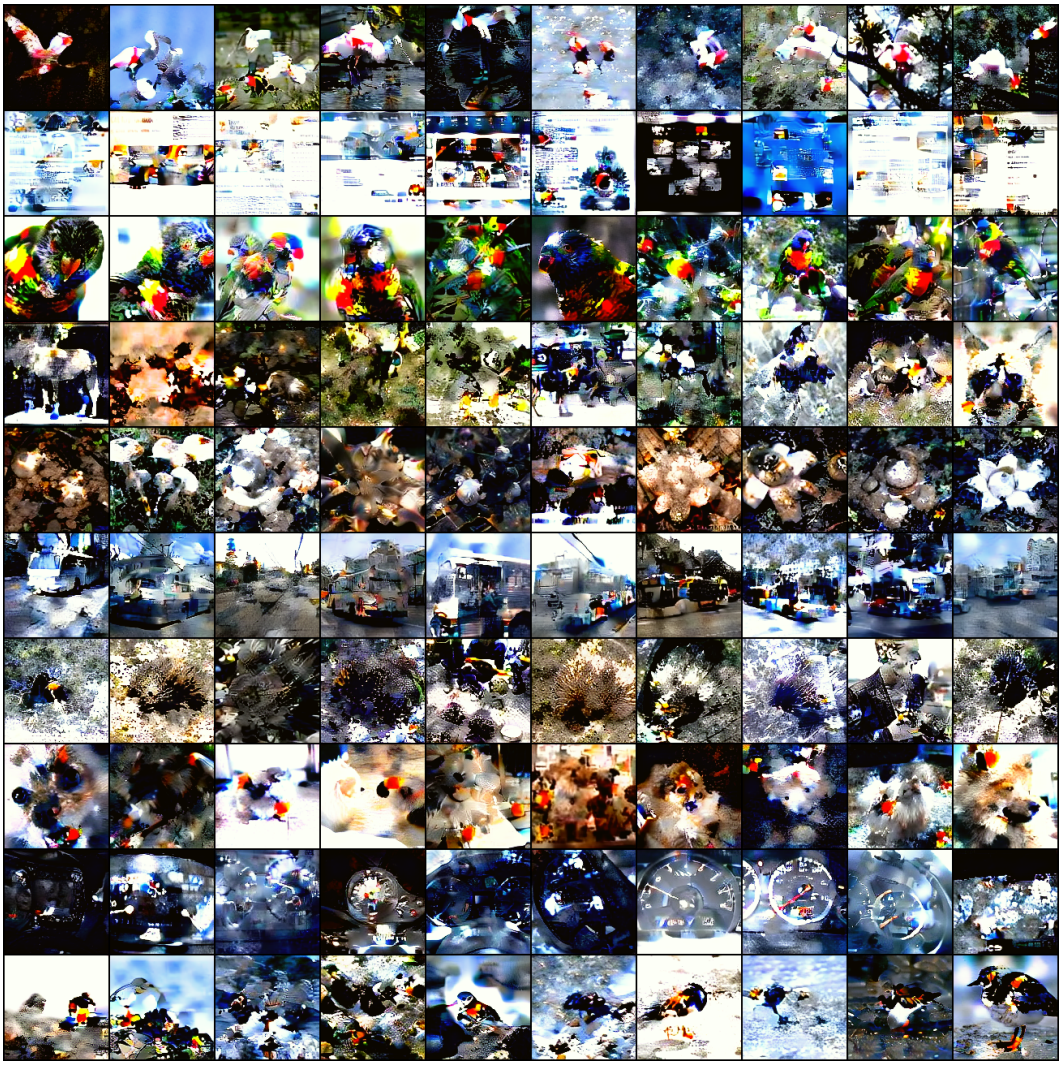

Figure 16: Images distilled by DM in LD3M for IPC=10 and ImageNet-B.

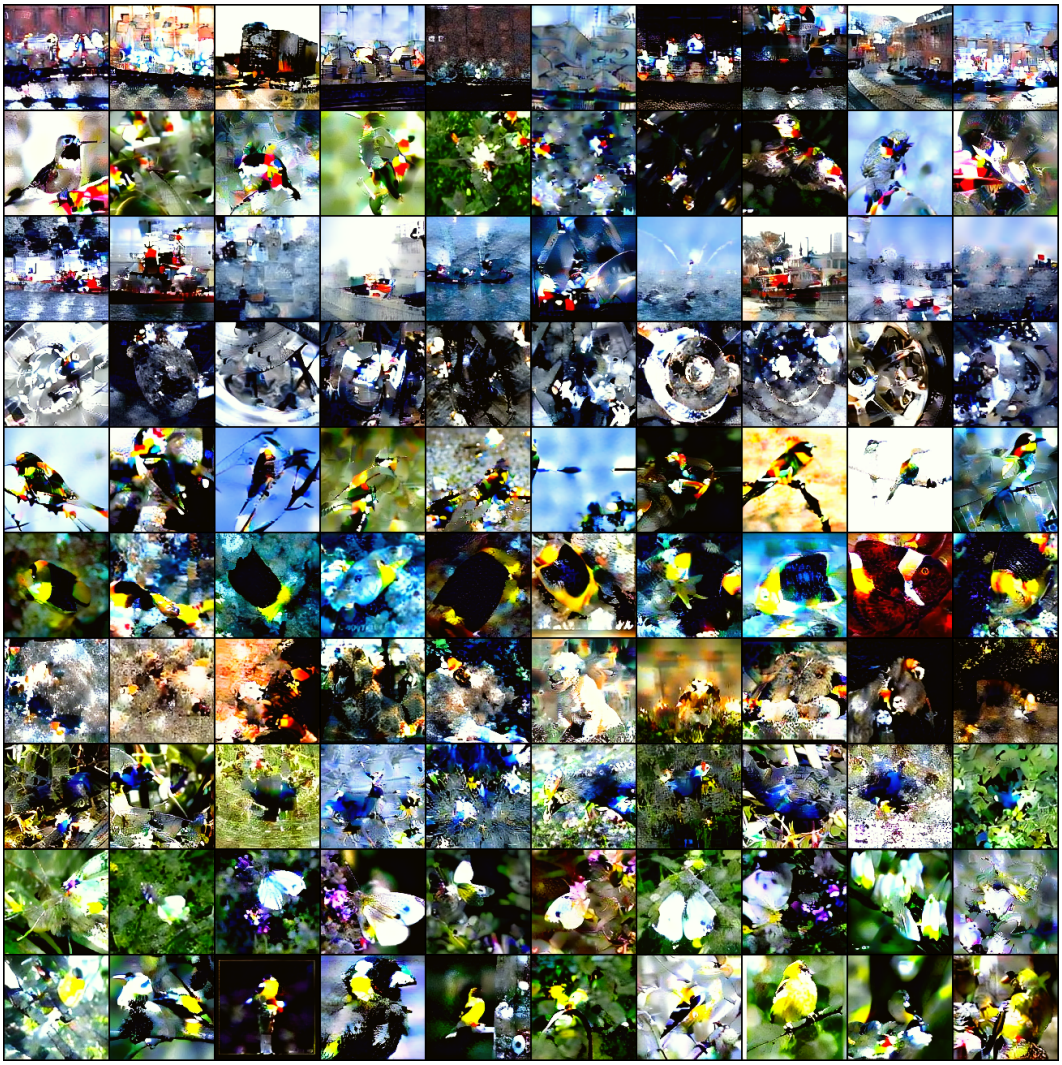

Figure 17: Images distilled by DM in LD3M for IPC=10 and ImageNet-C.

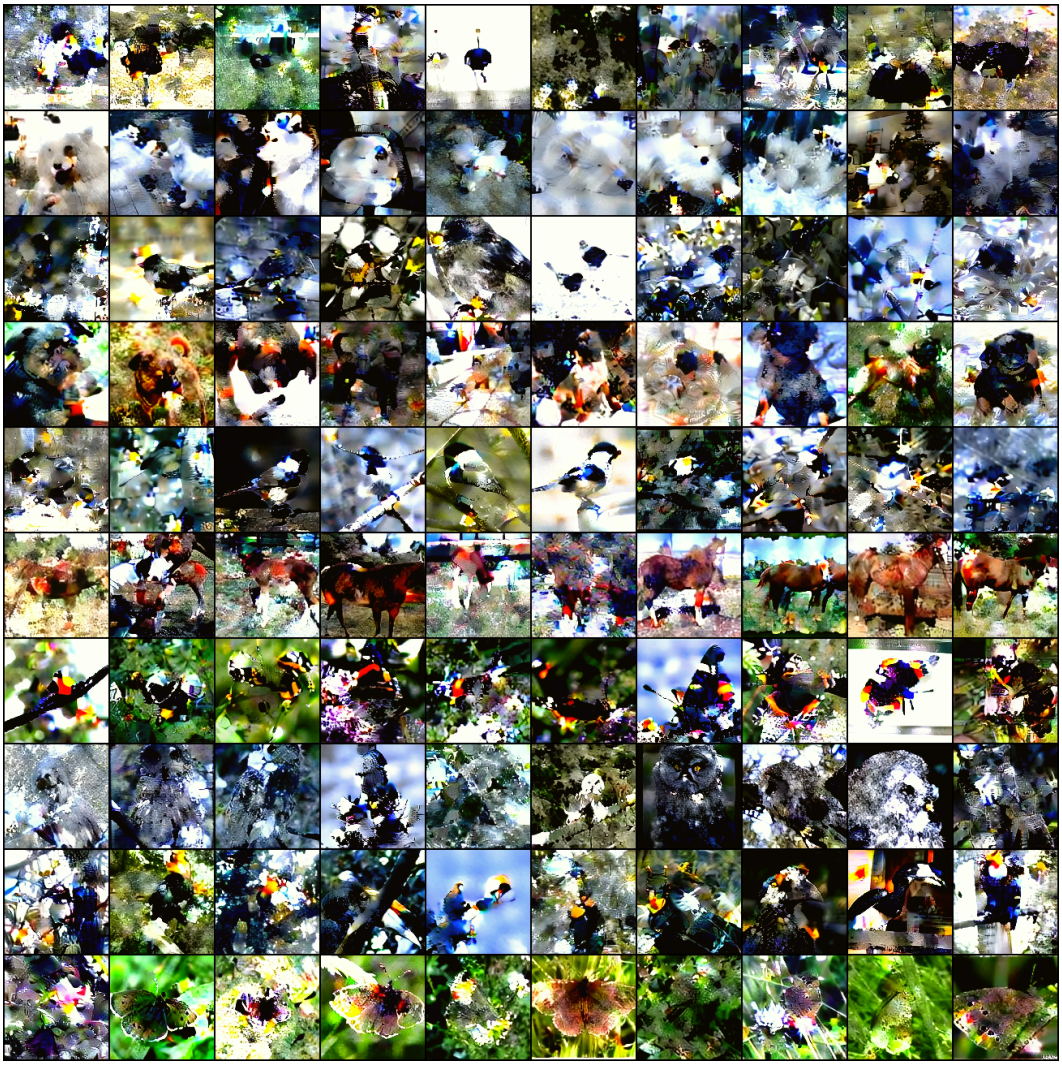

Figure 18: Images distilled by DM in LD3M for IPC=10 and ImageNet-D.