# Dataset Distillation via Drifting

**George Cazenavette** **Angelina Quan** **Giannis Daras**
**Antonio Torralba** **Vincent Sitzmann**
Massachusetts Institute of Technology
drifting-distillation.github.io

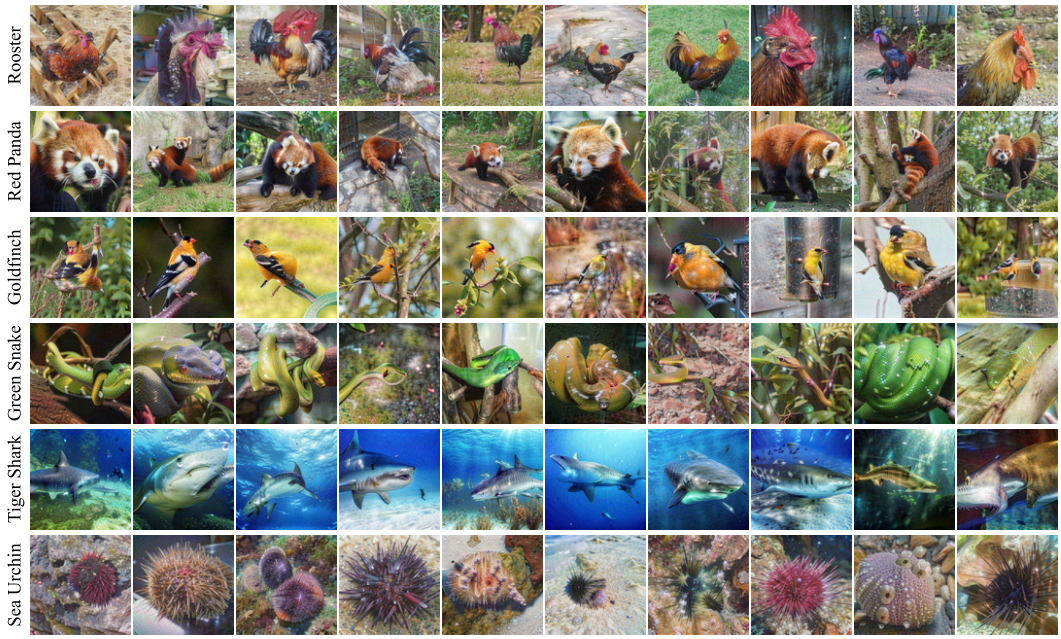


Figure 1: A selection of 6 classes from ImageNet-1k [12] distilled to 10 images per class via drifting with DINOv3 [53] features. Images seen here were initialized with *random noise*. Our method yields synthetic data that are high in both quality and diversity by virtue of the attractive and repulsive forces of our drifting loss.

## Abstract

The task of *dataset distillation* concerns the learning of a small set of synthetic samples that can serve as an effective replacement for a large real dataset when used for downstream tasks. While many solutions have been proposed since the problem's inception, most of them primarily seek to optimize the distilled data with respect to a classifier. In this work, we propose a new distillation method that learns a set of samples such that they directly approximate the per-class data distribution. Specifically, we "drift" our distilled data in the latent space of a feature extractor such that they move both toward the real data and away from each other without using a classification objective. We find that our drifted data cover the real distribution significantly better than existing methods while still serving as powerful training samples for classification. Furthermore, we showcase our method's use as a visualization tool for better interpreting models and extend it beyond images to other modalities. Our work advances the frontier of dataset distillation, offering a novel synthesis method and proposing distribution-based metrics as a complementary evaluation. More broadly, our findings help address the growing need to understand both data and models by uncovering more about how they interact.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

## 1 Introduction

Dataset distillation takes the classical problem of coreset selection [25] (choosing a representative subset of a large dataset) and extends it by allowing for the *learning* of a set of optimal samples rather than restricting to a selection of real data points. Since the task's proposal in the titular paper [65], dataset distillation solutions have focused on the optimization of synthetic samples for the purpose of training image classification models. Be it via gradient- [74, 76], trajectory- [4, 10, 24], or feature-matching [35, 62, 75] or directly maximizing a classifier's confidence [8, 9, 36, 49, 56, 71, 72, 78], distillation methods have traditionally indexed heavily toward the classification objective, often paying less attention to other potentially desirable attributes of a tiny synthetic dataset. Data distilled with respect to a decision boundary inherit the classifier's biases, possibly limiting the data's usability for other tasks or generalization to other models.

In this work, we aim to distill a dataset with the explicit goal of **best approximating the real data distribution**. Inspired by recent work in generative modeling [13], we learn our synthetic data via "drifting" such that each synthetic sample is attracted toward each real sample while simultaneously being repelled from every other synthetic sample. The competing positive and negative forces encourage the distilled data to spread across the real distribution in a way that promotes both coverage and diversity. As a handful of images alone cannot adequately approximate the real distribution, we instead drift many **augmentations** of each synthetic image at a time such that the distilled images, together with their augmentations, best cover the target distribution (Figure 2).

We find that this drifting method yields qualitatively more diverse and higher-quality images than existing methods (Figures 1 and 4). Despite being distilled without a classification objective, our images still serve as superior data for training such classifiers from scratch: the traditional dataset distillation benchmark (Sec. 4.1). Using MoCo-v3 [7] for the drifting feature space, our method reaches over 40% and 52% test accuracy when training a ResNet-18 [26] from scratch with fixed soft labels after distilling ImageNet-1k [12] down to just 10 and 50 images per class, respectively.

We further evaluate the distilled data on a set of feature-space metrics (coverage [41] along with nearest-neighbor and linear probe classification) where our method continues to match or outperform existing methods (Sec. 4.2). Our method also offers some model interpretability insights (Sec. 4.3), generalizes to other modalities (Sec. 4.4), and even serves as an avenue for art (Sec. 4.5).

Code and distilled datasets are available on our project page: drifting-distillation.github.io.

## 2 Related Work

**Dataset Distillation.** The concept of **coreset selection** [25], or finding a representative subset of a large dataset, has been thoroughly researched over many decades [2, 20, 37, 44, 46, 48, 57, 66, 69]. The idea of training a model to competitive performance using only a fraction of the data is attractive for many reasons, the most promising being the potential efficiency gains. More recently, **dataset distillation** [65], the *learning* of a small, representative dataset, has gained favor, as the direct optimization of a set of data points allows for more expressive and content-rich examples than being constrained to a selection of real samples.

For several years, *gradient*- and *trajectory*-matching methods [3, 4, 10, 24, 65, 74, 76] yielded the best performance on classification-based dataset distillation tasks. However, these expensive bi-level optimization methods fail to scale to realistic models (ResNet [26], ViT [16], etc.) and datasets (e.g., ImageNet-1k [12]). In contrast, the relatively cheap, often task-agnostic, *distribution*-matching methods [62, 75] scale computationally to large models and data but perform poorly at that scale.

Large-scale distillation began with the *Squeeze, Recover, and Relabel* [72] paradigm, which adopts the DeepInversion [70] objective: a DeepDream-style [39] model inversion that adds BatchNorm-statistic matching to the cross-entropy minimization with respect to a pre-trained classifier. Many follow-up works have since refined this method [9, 36, 49, 50, 71, 73]. These methods were the first to perform competitively on realistic models and datasets, but the distilled data are effective mainly for *knowledge distillation* [27] rather than from-scratch training. Some methods address the homogeneity issues that come with the classification objective via diversity-promoting regularizers [17, 38, 40, 51].

A related line of work instead selects (and then optionally further optimizes) real image patches by scoring them with a pre-trained classifier [8, 33, 56, 59, 64]. These methods are either restricted to,

or heavily biased toward, the empirical data distribution, severely limiting their application to other areas, such as privacy [15] or interpretability [3].

Another emerging area of dataset distillation focuses on leveraging pre-trained (most often diffusion-based) generative models to construct the synthetic data [23, 47, 55, 61, 77]. While previous works made use of generative priors [6] as a form of regularization, these methods leverage diffusion models to effectively draw information-dense samples from the real data manifold.

This work leverages newly proposed ideas from generative modeling to more effectively cover the data distribution. Notably, our method strictly focuses on approximating the real distribution and thus does not require a classification objective. Rather than requiring an extra loss term, diversity in our method arises naturally from the repulsive term of the drifting objective. Our method can also be initialized with random noise and does not make use of a pre-trained generative model.

**Drifting.** As a recently proposed approach to generative modeling [13], **drifting** upends the typical paradigm of diffusion [28, 29, 54] and flow-matching [34] models by moving the generated samples toward the target distribution at *training* time, removing the need for a costly denoising process at inference by natively enabling sampling in a single forward pass. In the simplest terms, each generated sample is simultaneously *pulled* toward every real sample and *pushed* away from every other generated sample (weighted by a softmax kernel) such that the distribution of generated samples eventually matches that of the real samples.

We apply drifting to **dataset distillation**. Rather than back-propagating the gradients of the drifting loss into the weights of a generative model, we *directly optimize* a discrete set of images such that these images, together with their augmentations, cover the target distribution. Aside from this change in parameterization, our only modification is to jointly drift many augmentations of each image at once, treating each augmentation as a unique image when computing negative distances. The rest of the drifting paradigm remains unchanged.

## 3 Method

In this section, we first formally motivate our distribution matching objective (Sec. 3.1) before defining our drifting loss (Sec. 3.2) and pyramid parameterization (Sec. 3.3).

### 3.1 Distribution Matching

The goal of the original drifting formulation [13] (or most generative modeling frameworks) is to learn a process that maps from one distribution ($p_\epsilon$) to another ($p_{\text{data}}$). Typically, $p_\epsilon$ is an infinitely sampleable, continuous noise distribution, and either a single- (drifting [13], GAN [22]) or multi-step (diffusion [28, 54], flow matching [34]) model $f$ is learned to map from $p_\epsilon$ to $p_{\text{data}}$ such that $p_{\text{data}} \approx f_\# p_\epsilon$, where "$f_\#$" denotes the pushforward induced by the model.

Rather than a generative model $f$, our goal is to *directly learn* a small set of synthetic samples $\mathbf{x} \in \mathcal{S}$ such that the distribution of these samples ($p_{\text{syn}}$) matches that of the real data:

$$p_{\text{syn}} \triangleq \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \delta_{\mathbf{x}} \approx p_{\text{data}} \tag{1}$$

where $\delta$ is the Dirac delta distribution. As described, this is not feasible: a handful of synthetic samples alone cannot adequately approximate an entire distribution of real data. Rather than attempting to optimize our synthetic data such that they alone cover the real distribution, we sample from a distribution of *augmentations* $A \sim P_A$ and optimize such that our samples, together with their augmentations, properly approximate the real distribution:

$$\mathbb{E}_{A \sim P_A}[A_\# p_{\text{syn}}] \approx p_{\text{data}} \tag{2}$$

In practice, $P_A$ consists of random resized crops, horizontal flips, and Gaussian noise.

### 3.2 Drifting for Dataset Distillation

To match distributions, we adopt the **drifting field** of [13]. Rather than the "evolution $\approx$ denoising" analogy, we view it as a "force" field wherein the synthetic samples are pulled and pushed in various

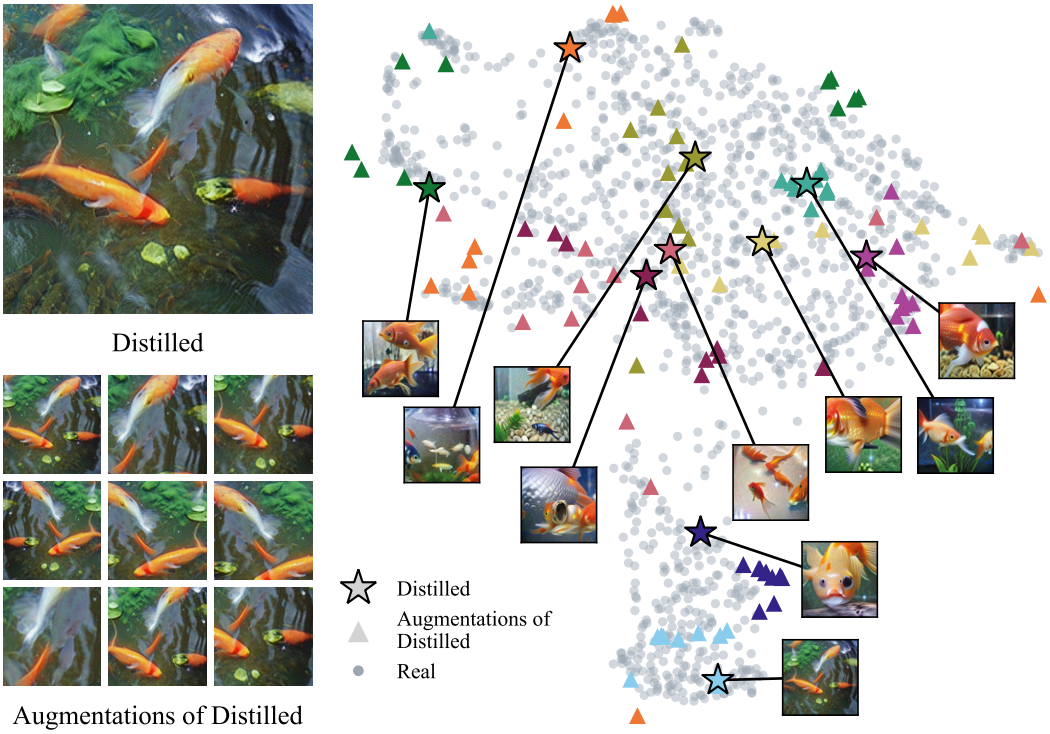


Figure 2: **Augmentations of Distilled Images Cover the Real Distribution.** At each distillation step, we take multiple augmentations of each synthetic sample **(left)**. The synthetic data, together with their augmentations, approximately cover the distribution of real data **(right)**. In this plot, the stars represent the distilled images, the triangles 10 augmentations of each distilled image, and the circles real data. Drifting multiple augmentations of each synthetic sample at each step leads to high *intra*-image diversity, allowing some distilled images (such as the orange star) to cover many parts of the real distribution at the same time.

directions. That is, for a given sample $\mathbf{x} \in \mathcal{S}$ and drifting field $\mathbf{V}$, $\mathbf{V}(\mathbf{x})$ gives the total "force" acting on $\mathbf{x}$. This leads to the following loss function where $\mathbf{x}$ is "pushed" toward its drifted target $\mathbf{x} + \mathbf{V}(\mathbf{x})$:

$$\mathcal{L} = \sum_{\mathbf{x}_i \in \mathcal{S}} \|\mathbf{x}_i - \texttt{stopgrad}\left(\mathbf{x}_i + \mathbf{V}(\mathbf{x}_i)\right)\|_2^2 . \tag{3}$$

The `stopgrad` operation provides a frozen target for the distilled samples. Rather than the weights of a generative model, this drifting loss is back-propagated **directly to our synthetic data**. Our drifting field $\mathbf{V}$ consists of two components: positive (attractive) and negative (repulsive). For a given synthetic sample $\mathbf{x}_i \in \mathcal{S}$, we have:

$$\begin{aligned} \mathbf{V}^+(\mathbf{x}_i) &\coloneqq \frac{1}{Z_{\text{data}}} \mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} \left[\kappa(\mathbf{x}_i, \mathbf{y})(\mathbf{y} - \mathbf{x}_i)\right] \\ \mathbf{V}^-(\mathbf{x}_i) &\coloneqq \frac{1}{Z_{\text{syn}}} \frac{1}{|\mathcal{S}|} \sum_{j, j \neq i} \kappa(\mathbf{x}_i, \mathbf{x}_j)(\mathbf{x}_j - \mathbf{x}_i) \end{aligned} \tag{4}$$

where $Z_{\text{data}}$ and $Z_{\text{syn}}$ are the normalization factors

$$Z_{\text{data}} \coloneqq \mathbb{E}_{y \sim p_{\text{data}}} \left[\kappa(\mathbf{x}_i, \mathbf{y})\right], \qquad Z_{\text{syn}} \coloneqq \frac{1}{|\mathcal{S}|} \sum_{j, j \neq i} \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

and $\kappa$ is the kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\tau}\|\mathbf{x} - \mathbf{y}\|_2\right) \tag{6}$$

with temperature $\tau$. The combined drifting field can be written as

$$\mathbf{V}(\mathbf{x}) \coloneqq \mathbf{V}^+(\mathbf{x}) - \mathbf{V}^-(\mathbf{x}) \tag{7}$$

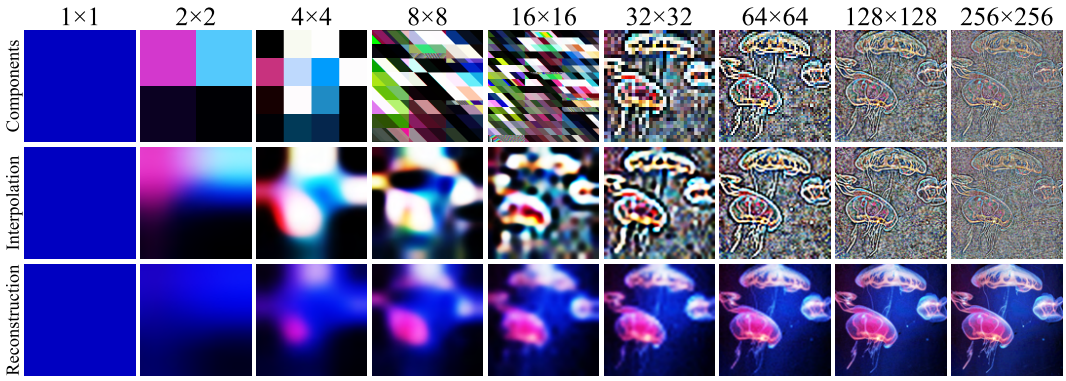


Figure 3: We represent distilled data as a Laplacian pyramid of cascading resolutions. To reconstruct a synthetic sample, each level (**top**) is bicubically interpolated to the full resolution (**middle**) before being added together **(bottom)**. This acts as a hyperparameter-free implicit regularizer on the distilled data, preventing overfitting to adversarial patterns often seen in model inversion tasks.

Intuitively, these positive and negative fields can be thought of as the attractive (or repulsive) force between a sample $\mathbf{x}_i$ and all other real (or synthetic) samples weighted by the distances between them via a softmax operation. In short, each synthetic sample is optimized to both be close to real samples and far from other synthetic samples. A given synthetic sample $\mathbf{x}_i$ will reach equilibrium when the total attractive and repulsive forces induced by the other real and synthetic samples cancel each other out, leading to $\mathbf{V}(\mathbf{x}_i) = \mathbf{0}$.

Rather than in pixel space, we compute the drifting field in the latent space of a pre-trained feature extractor, $\phi$. The choice of $\phi$ plays a crucial role, as it defines the space in which we want to match the real distribution; a stronger feature extractor may lead to better semantic alignment in image space. Combined with the augmentations $A \sim P_A$ described in Sec. 3.1, we have our full loss function[1]:

$$\mathcal{L} = \mathbb{E}_{A\sim P_A}\left[\sum_{\mathbf{x}\in\mathcal{S}} \left\|\phi\left(A(\mathbf{x})\right) - \texttt{stopgrad}\left(\phi\left(A(\mathbf{x})\right) + \mathbf{V}(\phi\left(A(\mathbf{x})\right))\right)\right\|_2^2\right] \tag{8}$$

In practice, multiple augmentations of the same synthetic sample are also considered *negative* samples with respect to each other, leading to increased intra-*sample* diversity as well as the inherent intra-class variety afforded by the drifting paradigm. We visualize our distilled distribution in Figure 2 and include pseudocode of the full process in Algorithms 1 and 2.

### 3.3 Implicit Regularization via Laplacian Pyramids

Inspired by recent work in dataset distillation [3] and model inversion [21], we choose to parameterize our distilled data as a *Laplacian pyramid* of different resolutions. Rather than naively optimizing the *pixels* of an image, we instead represent each sample as a cascade of images with progressively lower resolutions: $\rho = \{1\times 1, 2\times 2, 4\times 4, \ldots, R\times R\}$ where $R\times R$ is the full resolution. Together, these layers form a Laplacian pyramid $P = \{\mathbf{P}_r | r \in \rho\}$.

At each optimization step, the synthetic sample $\mathbf{x}$ is first built in the same manner an image is reconstructed from its Laplacian pyramid; each layer of the pyramid is bicubically upsampled to the canonical resolution before all layers are added together:

$$\mathbf{x} = \text{sigmoid}\left(\sum_{r\in\rho} \text{resize}_R\left(\mathbf{P}_r\right)\right) \tag{9}$$

The sigmoid operation acts as a soft, differentiable clamp to keep the range of the synthetic samples within that of the real data. As with prior work [3], we initialize our pyramid at low resolution and progressively add more layers throughout the distillation process. We also apply a fixed 3×3 decorrelation transform to the color channels such that the distilled images are biased toward natural colors [42].

Notably, this pyramid technique generalizes effortlessly to both 1D (audio) and 3D (video) data, as seen in Section 4.4. Please see Figure 3 for a visualization of one of our distilled image pyramids.

[1] Omitted here for readability: we additionally apply the same normalization terms as [13] (Appendix C).

| Method | Hard-Label | | Soft-Label | | Knowledge Distillation | |
|---|---|---|---|---|---|---|
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Noise init* | | | | | | |
| SRe$^2$L [72] | 8.1 ± 0.2 | 10.7 ± 0.3 | 17.5 ± 0.2 | 22.9 ± 0.3 | 40.6 ± 0.3 | 50.1 ± 0.1 |
| CDA [71] | 9.5 ± 0.1 | 16.1 ± 0.3 | 19.2 ± 0.2 | 28.8 ± 0.1 | 41.1 ± 0.2 | 50.8 ± 0.1 |
| G-VBSM$^\dagger$ [49] | 11.7 ± 0.3 | 24.9 ± 0.9 | 29.8 ± 0.2 | 44.9 ± 0.3 | 43.8 ± 0.3 | 53.0 ± 0.3 |
| WMDD [35] | 10.2 ± 0.3 | 11.6 ± 0.6 | 29.4 ± 0.5 | 40.6 ± 0.1 | 43.7 ± 0.1 | 53.2 ± 0.2 |
| SC-DD [78] | 6.1 ± 0.2 | 10.2 ± 0.1 | 19.1 ± 0.2 | 31.0 ± 0.3 | 40.8 ± 0.2 | 51.2 ± 0.2 |
| Ours (ResNet-18) | 17.6 ± 0.2 | 18.4 ± 0.5 | 30.5 ± 0.8 | 43.1 ± 0.3 | 44.8 ± 0.7 | 55.0 ± 0.3 |
| Ours (MoCo-v3) | **27.0** ± 0.2 | **33.5** ± 1.1 | **40.3** ± 0.1 | **52.8** ± 0.0 | **48.5** ± 0.1 | **57.2** ± 0.3 |
| Ours (DINOv3) | 21.3 ± 0.5 | 30.2 ± 0.2 | 35.9 ± 0.1 | 50.7 ± 0.1 | 45.0 ± 0.2 | 56.0 ± 0.2 |
| *Real init* | | | | | | |
| RDED [56] | 22.1 ± 0.1 | 33.8 ± 1.0 | 33.1 ± 0.2 | 44.0 ± 0.3 | 46.7 ± 0.3 | 56.1 ± 0.1 |
| CUDD [36] | 15.2 ± 0.2 | 19.8 ± 0.4 | 28.3 ± 0.1 | 41.1 ± 0.3 | 46.6 ± 0.1 | 55.4 ± 0.1 |
| FADRM [8] | 18.7 ± 0.1 | 25.7 ± 0.5 | 29.7 ± 0.1 | 40.0 ± 0.3 | 48.5 ± 0.3 | 56.4 ± 0.1 |
| FADRM+$^\dagger$ [8] | 20.5 ± 0.5 | 26.2 ± 0.6 | 32.5 ± 0.2 | 41.8 ± 0.3 | **49.2** ± 0.5 | 56.6 ± 0.1 |
| CV-DD$^\dagger$ [9] | 16.7 ± 0.0 | 21.9 ± 0.3 | 28.7 ± 0.3 | 38.3 ± 0.1 | 45.6 ± 0.1 | 54.2 ± 0.1 |
| Ours (ResNet-18) | 20.4 ± 0.4 | 22.0 ± 0.4 | 33.9 ± 0.2 | 45.4 ± 0.5 | 47.8 ± 0.2 | 56.4 ± 0.3 |
| Ours (MoCo-v3) | **28.9** ± 0.2 | 36.0 ± 0.7 | **41.7** ± 0.0 | 53.4 ± 0.1 | **49.6** ± 0.2 | 57.7 ± 0.2 |
| Ours (DINOv3) | 26.0 ± 0.2 | **41.1** ± 0.4 | 40.5 ± 0.0 | **54.3** ± 0.4 | **49.5** ± 0.3 | **58.1** ± 0.1 |

Table 1: **From-Scratch Training.** This table shows ImageNet-1k [12] classification performance when training a ResNet-18 [26] from scratch. We include results using one-hot hard labels (**left**), one-time soft labels (**middle**), and full knowledge distillation (**right**). Methods marked with a dagger$^\dagger$ distill using an ensemble of encoders. Across all three training paradigms, our distilled data lead to the best downstream performance.

# 4 Experiments

As the focus of this work is *large-scale* dataset distillation, our experimental results center on ImageNet-1k [12]. All distillation experiments are conducted one class at a time, allowing for easy parallelization across many GPUs. We primarily use three backbones for our image distillation experiments: ResNet-18 [26], DINOv3 [53], and MoCo-v3 [7] (ViT-B for both transformers). The specific features to which we apply the drifting loss are detailed in Appendix C. We compare against both noise-initialized methods (SRe$^2$L [72], CDA [71], G-VBSM [49], WMDD [35], and SC-DD [78]) and real-initialized methods (RDED [56], CUDD [36], FADRM and FADRM+ [8], and CV-DD [9]). Since much of the improvement among real-initialized methods seems to come from how the initial images are chosen [8, 36, 56], we report the two groups separately and run our own method under both initializations. Noise initialization remains of particular interest, as initializing with real images limits usefulness for some downstream tasks, such as those reliant on privacy [15]. We compare against diffusion-based methods [23, 47, 55, 61, 77] in Appendix Table B.2.

## 4.1 Dataset Distillation via Drifting Yields Superior Training Samples.

While many prior large-scale dataset distillation works bake the classification task into the data by directly minimizing a classifier's loss [8, 9, 36, 49, 56, 71, 72, 78], our drifting method is *solely* designed to match the per-class **distribution** of the real data. In fact, our method does not require a classification objective at all; drifting can occur in the space of any feature extractor, regardless of training objective. Nevertheless, we still find that our synthetic images serve as excellent data for training classifiers.

In Table 1, we train ResNet-18 [26] from scratch using the distilled data from each method in three settings: hard-label, soft-label, and knowledge distillation. Hard-label uses one-hot labels, while soft-label first computes a single fixed target vector for each training sample using a teacher model. Knowledge distillation (the evaluation method used in most prior large-scale distillation works) computes a new target vector at every step by passing the current augmented version of the training image through the teacher model; we include it for comparison with prior work, but note that it is effectively a teacher-student transfer task rather than from-scratch training (see Appendix C). To avoid confounding evaluation factors, metrics for all methods were recomputed using the exact same training recipes (Appendix D). We report the mean and standard deviation over 3 fixed seeds.

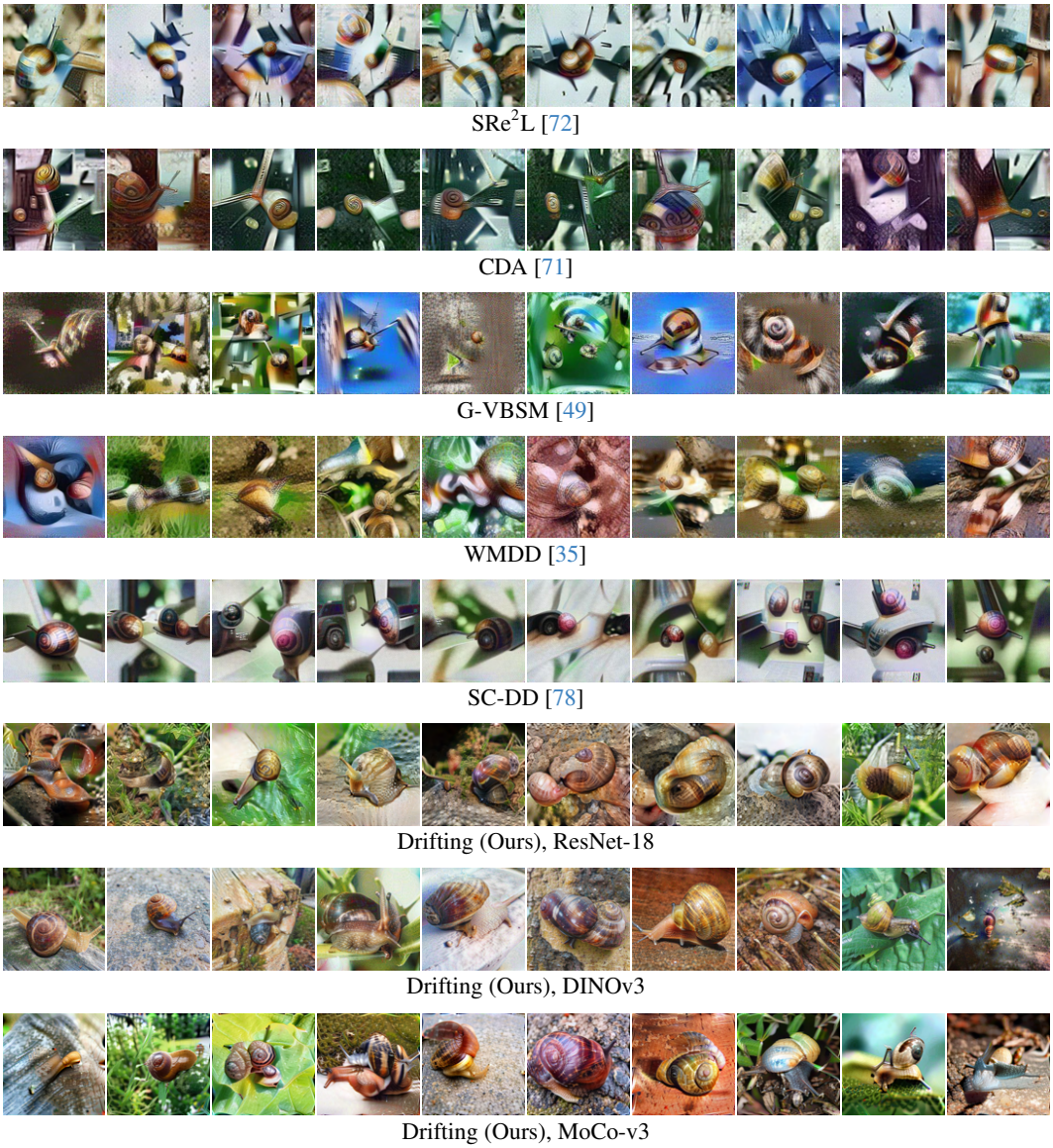


Figure 4: **Comparing Noise-Initialized Distillation Methods.** Even when using ResNet-18 as the distillation backbone, our method yields higher-quality and more diverse samples. When using a stronger backbone (DINOv3 [53]), the visual quality and diversity of our method increase further.

For the hard- and soft-label settings, our method consistently outperforms both the noise- and real-initialized baselines. In the knowledge distillation setting, we still match or exceed the baselines despite the smaller gap (due to the task's tendency to saturate). The performance difference is most dramatic in the hard-label setting, especially with noise initialization. At just 10 images per class, our best noise-initialized variant (MoCo-v3 [7]) reaches 27.0% test accuracy while the best noise-initialized baseline (G-VBSM [49]) achieves only 11.7%. With real initialization, our MoCo-v3 data reach 28.9% versus 22.1% for the best baseline (RDED [56]), and at 50 images per class the gaps remain large (33.5% vs. 24.9% with noise initialization and 41.1% vs. 33.8% with real initialization).

Notably, the best training data come from the self-supervised backbones (MoCo-v3 and DINOv3), which were never trained with labels, while the data distilled with ResNet-18 (a supervised classifier with the same architecture as the student) is the weakest of our three in every setting. Even so, our ResNet-18 data still outperforms the individual baselines in 49 of 60 pairwise comparisons (28 of 30 with noise initialization), and outperforms every noise-initialized baseline in four of the six settings. This suggests that the efficacy of our data stems from matching the real distribution in a rich feature space rather than from alignment with the student architecture or the classification task. We also note

| Method | Coverage | | | | k-NN | | | | Linear | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SigLIP2 | | MoCo-v3 | | SigLIP2 | | MoCo-v3 | | SigLIP2 | | MoCo-v3 | |
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Noise init* | | | | | | | | | | | | |
| SRe$^2$L [72] | 7.2 | 11.6 | 4.9 | 8.5 | 50.1 | 54.3 | 46.0 | 49.1 | 51.2 | 55.6 | 41.1 | 45.4 |
| CDA [71] | 6.5 | 11.1 | 4.6 | 8.7 | 44.5 | 51.7 | 44.8 | 49.5 | 51.0 | 57.7 | 43.2 | 50.9 |
| G-VBSM$^\dagger$ [49] | 5.7 | 8.6 | 5.3 | 8.4 | 41.3 | 45.9 | 39.7 | 42.5 | 62.3 | 70.5 | 47.1 | 61.8 |
| WMDD [35] | 9.5 | 17.5 | 9.4 | 19.0 | 49.6 | 56.0 | 45.8 | 50.7 | 65.2 | 66.4 | 45.7 | 58.7 |
| SC-DD [78] | 9.7 | 16.1 | 9.3 | 16.8 | 48.5 | 53.3 | 49.1 | 51.1 | 52.7 | 58.0 | 48.7 | 52.8 |
| Ours (ResNet-18) | 19.7 | 38.4 | 41.3 | 66.3 | 54.3 | 61.2 | 55.3 | 58.7 | 69.6 | 71.2 | 59.8 | 66.5 |
| Ours (MoCo-v3) | 61.2 | 86.9 | (91.9) | (97.9) | 64.3 | 68.4 | (60.2) | (62.5) | 74.6 | 74.6 | (66.6) | (69.9) |
| Ours (DINOv3) | **77.0** | **93.2** | **83.4** | **96.3** | **69.9** | **72.5** | **61.1** | **63.2** | **77.5** | **78.6** | **66.3** | **70.3** |
| *Real init* | | | | | | | | | | | | |
| RDED [56] | 23.3 | 34.6 | 31.8 | 49.2 | **70.2** | 71.6 | **61.0** | 62.1 | 74.4 | 77.2 | 63.5 | 66.6 |
| CUDD [36] | 11.2 | 15.1 | 10.5 | 17.1 | 56.3 | 58.0 | 52.1 | 53.9 | 63.2 | 63.5 | 53.9 | 58.9 |
| FADRM [8] | 28.5 | 43.0 | 30.3 | 52.3 | 62.8 | 66.9 | 57.8 | 61.3 | 67.6 | 71.5 | 61.1 | 64.9 |
| FADRM+$^\dagger$ [8] | 32.1 | 48.0 | 36.5 | 59.7 | 64.3 | 68.1 | 58.5 | 61.8 | 69.2 | 73.1 | 61.8 | 65.4 |
| CV-DD$^\dagger$ [9] | 10.0 | 16.6 | 8.4 | 15.8 | 55.7 | 61.1 | 51.8 | 54.9 | 62.7 | 66.6 | 52.7 | 57.4 |
| Ours (ResNet-18) | 30.0 | 57.5 | 54.9 | 82.0 | 62.1 | 68.3 | 58.0 | 61.3 | 72.8 | 74.5 | 63.1 | 67.8 |
| Ours (MoCo-v3) | 60.6 | 88.5 | (91.3) | (98.0) | 65.0 | 69.3 | (60.2) | (62.2) | 75.0 | 75.8 | (66.4) | (70.0) |
| Ours (DINOv3) | **79.9** | **95.5** | **85.1** | **97.5** | 69.9 | **72.8** | 60.3 | **63.0** | **77.0** | **78.5** | **65.7** | **70.2** |

Table 2: **Feature-Space Metrics.** Coverage (**left**), $k$-NN accuracy (**middle**), and linear probe accuracy (**right**) of distilled ImageNet-1k data in the SigLIP2 [60] and MoCo-v3 [7] latent spaces (see Appendix D). Parenthesized entries are self-evaluations and are not ranked. Methods marked with a dagger$^\dagger$ distill using an ensemble of encoders. Our method outperforms all baselines in every feature-space setting except for RDED [56] on IPC 10 $k$-NN in both latent spaces. The performance gap is especially large on the coverage metric.

that G-VBSM, FADRM+, and CV-DD (marked † in Tables 1 and 2) distill against an ensemble of backbones, whereas each of our variants uses a single encoder.

We include results for additional distillation backbones in Appendix Table B.1 and report from-scratch performance on the ImageNette and ImageWoof [19] datasets in Appendix Tables B.4 and B.5.

### 4.2 Drifted Synthetic Datasets Cover the Real Distribution.

While the traditional goal of dataset distillation methods has been the training of classifiers, our primary motivation is to learn a small set of synthetic images that best approximates the real distribution. To measure this directly, we evaluate the distilled data in the latent space of pre-trained encoders using three metrics: **coverage** [41], 1-nearest-neighbor ($k$-NN with $k = 1$) classification, and linear probe accuracy (details in Appendix D). Coverage measures the fraction of real validation points $\mathbf{y}_j \in \mathcal{T}_{\text{val}}$ whose neighborhoods, defined by their $k$-th nearest-neighbor distances $\text{NND}_k(\mathbf{y}_j)$ with respect to other real points, contain at least one synthetic point:

$$\text{Coverage} = \frac{1}{M}\sum_{j=1}^{M} \mathbb{1}\left[\exists\, i \text{ s.t. } d\big(\phi(\mathbf{x}_i), \phi(\mathbf{y}_j)\big) \leq \text{NND}_k(\mathbf{y}_j)\right] \tag{10}$$

where $d$ is the cosine distance. We use $k = 5$ and report the average coverage over all classes. We also add 10 random crops of each image to the pool to widen the synthetic distribution.

We evaluate in the latent spaces of SigLIP2 [60] (language-supervised) and MoCo-v3 [7] (self-supervised). Cells in which our MoCo-v3 variant is scored in its own distillation space are shown in parentheses and excluded from the rankings. Such self-evaluation seems to strongly inflate coverage (91.9% versus 83.4% for our DINOv3 variant at 10 images per class with noise initialization), but not $k$-NN or linear accuracy, where our MoCo-v3 variant is roughly on par with our DINOv3 variant.

In Table 2, our DINOv3 variant achieves the highest coverage in every setting (excluding self-evaluation); at 10 images per class, it covers 77.0% (SigLIP2) and 83.4% (MoCo-v3) of the real distribution, compared to at most 9.7% for any noise-initialized and 36.5% for any real-initialized baseline. Our ResNet-18 variant covers more of the distribution than every noise-initialized baseline




Figure 5: **Distilled images elucidate model representations.** Each of the models shown here induces a different style of distilled images to best cover its own distribution. CLIP [45] is a notable outlier, possibly due to its strong text-entanglement and the contrastive loss not used in more recent vision-text models, such as SigLIP2 [60].

in all settings, and more than every real-initialized baseline in all but one (SigLIP2 at 10 images per class, where FADRM+ reaches 32.1% versus our 30.0%). Our DINOv3 variant is also the best on $k$-NN and linear probe accuracy in every setting except two $k$-NN cells at 10 images per class, where RDED is ahead by less than one point. Notably, the gaps in coverage are far larger than those in $k$-NN or linear accuracy: classification metrics only require the distilled data to separate classes, whereas coverage rewards spanning the full extent of each class distribution, which is precisely what drifting optimizes for. Qualitatively, we see our distilled images are more diverse than those of the baseline methods, likely aiding in the distribution coverage (Figure 4).

Much of this distributional coverage comes from the distillation-time augmentations (Table A.1), which allow our method to explicitly optimize for intra-image diversity. We visualize this in Figure 2; while the distilled images themselves (stars) are relatively evenly spaced throughout the latent space, their augmentations (triangles) help cover the rest of the real distribution.

### 4.3 Distilled Data Elucidate Model Biases.

One recent work (Linear Gradient Matching [3]) focused specifically on distillation for pre-trained feature extractors. While this method highlighted the interactions between different models and their distilled data, it could not scale past 1 image per class due to massive resource requirements as a result of needing to jointly distill all classes together. We pick up where this work left off and distill 10 images per class with seven different backbones (ResNet-18 [26], ViT [16], CLIP [45], SigLIP2 [60], EVA-02 [18], MoCo-v3 [7], and DINOv3 [53]). In Figure 5, we see the monarch butterfly class distilled to 10 images per class for each of these models. As noted for 1 image per class in the linear gradient matching settings [3], each model yields images with its own particular bias despite using the same underlying distillation method. Most notably, we see that the CLIP images are quite murky and contain less-well-defined backgrounds than those of the other models, possibly due to the model's text entanglement. This is reflected quantitatively as well; when each set is evaluated in the other models' feature spaces (Appendix Table B.6), CLIP-distilled data cover the least of the real distribution (on average) while DINOv3-distilled data cover the most.



Figure 6: Four frames of a basketball dribbling video from HMDB51 [32] distilled via drifting with VideoMAE-v2 [63] into a 3D Laplacian pyramid.

| | Audio | | Video | |
|---|---|---|---|---|
| Method | Linear | $k$-NN | Linear | $k$-NN |
| Herding | 45.6 | 39.9 | 66.6 | 60.4 |
| Ours | **51.4** | **52.6** | **70.3** | **65.8** |

Table 3: Our drifting method generalizes to both audio and video, outperforming the coreset baseline.

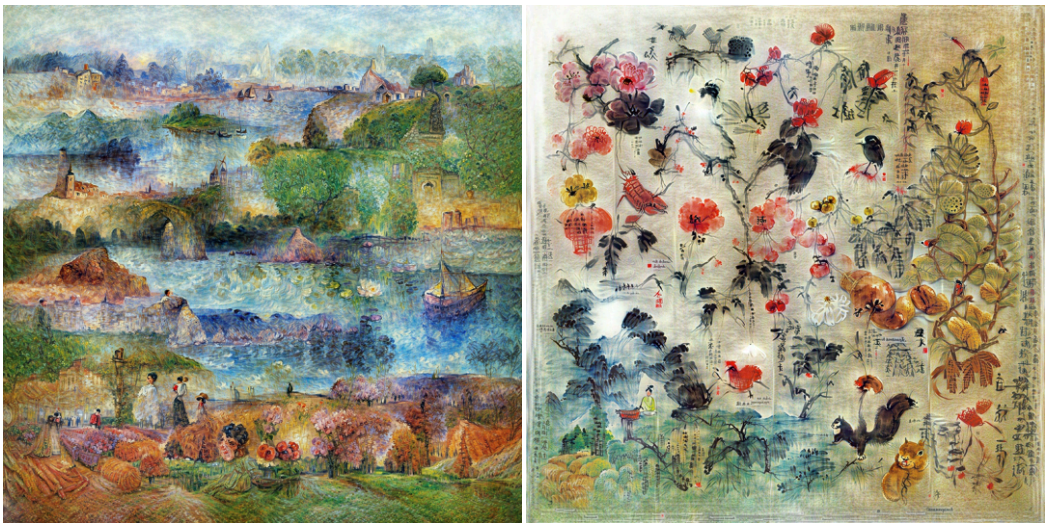

Figure 7: **Distilling Canvases.** We can distill a collection of images into a single large canvas via drifting by using a large number of random crops at each step. The above $2048 \times 2048$ pixel images were synthesized by drifting SigLIP2 [60] features to match the distribution of Claude Monet's (**left**) and Qi Baishi's (**right**) paintings.

### 4.4 Drifting Distillation Generalizes to Other Modalities.

We find that our drifting method and pyramid parameterization generalize to both **audio** and **video** data. For audio, we distill the FMA [11] small dataset of 8 music genres using the MusicFM-FMA [68] feature extraction model. For video, we distill the HMDB51 [32] human action recognition dataset of 51 action categories using the VideoMAE-v2 [63] feature extraction model. We distill 5 samples per class and evaluate with a linear probe and cosine 1-NN classification on mean-pooled features. For both modalities, we compare to and outperform the herding [66] coreset selection baseline (Table 3). Figure 6 shows frames from one of our distilled "dribbling" videos.

### 4.5 An Image is Worth a Thousand Paintings.

While dataset distillation's artistic value has been briefly explored in the past [5], we find that our drifting method excels at distilling the essence of an art style into a large canvas [52]. In Figure 7, we see the paintings of Claude Monet and Qi Baishi distilled into single images using SigLIP2 [60]. Remarkably, each distilled image clearly reflects its painter's individual style and contains common elements found throughout their work (e.g., Monet's water lilies and Qi's squirrels).

## 5 Conclusion

In this work, we introduced a new method of **Dataset Distillation via Drifting**, designed to explicitly learn a synthetic set of images that *best covers the target distribution*. In addition to best approximating the real data distribution when compared to the baseline methods, our distilled images also serve as superior training data for classification models from scratch.

Furthermore, our drifting method offers insights into model interpretability, generalizes to audio and video, and can be used to synthesize artwork. We hope our work inspires further dataset distillation research, especially in the direction of distribution coverage.

**Acknowledgments.** This work was supported by the National Science Foundation under Grant No. 2211259, by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) under 140D0423C0075, by the Amazon Science Hub, by the MIT-Google Program for Computing Innovation, by AMD via the MIT AI Hardware Program, by an ONR MURI grant 037564-00001, and by a 2025 MIT Office of Research Computing and Data Seed Grant. Giannis Daras is supported by a Jane Street Research Collaboration award, a TPU Research Cloud grant, and Lambda's Research Grant Program. We thank Mingyang Deng for sharing an early version of his drifting code with us and other members of the MIT community for providing valuable feedback on this work.

## References

[1] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *ASPLOS*, 2024. 25

[2] Z. Borsos, M. Mutny, and A. Krause. Coresets via bilevel optimization for continual learning and streaming. *NeurIPS*, 2020. 2

[3] G. Cazenavette, A. Torralba, and V. Sitzmann. Dataset distillation for pre-trained self-supervised vision models. *NeurIPS*, 2025. 2, 3, 5, 9

[4] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022. 2

[5] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Wearable ImageNet: Synthesizing tileable textures via dataset distillation. In *CVPR*, 2022. 10

[6] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Generalizing dataset distillation via deep generative prior. In *CVPR*, 2023. 3

[7] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021. 2, 6, 7, 8, 9, 18, 20, 25

[8] J. Cui, X. Bi, Y. Luo, X. Zhao, J. Liu, and Z. Shen. FADRM: Fast and accurate data residual matching for dataset distillation. In *NeurIPS*, 2025. 2, 6, 8, 19, 20

[9] J. Cui, Z. Li, X. Ma, X. Bi, Y. Luo, and Z. Shen. Dataset distillation via committee voting. *arXiv*, 2025. 2, 6, 8

[10] J. Cui, R. Wang, S. Si, and C.-J. Hsieh. Scaling up dataset distillation to ImageNet-1K with constant memory. In *ICML*, 2023. 2

[11] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson. FMA: A dataset for music analysis. In *ISMIR*, 2017. 10, 25

[12] 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. 1, 2, 6, 21, 25

[13] M. Deng, H. Li, T. Li, Y. Du, and K. He. Generative modeling via drifting. In *NeurIPS*, 2026. 2, 3, 5, 21, 24

[14] P. Dey, A. Sahdev, S. Bhati, K. R. Mopuri, and R. V. Babu. Rethinking dataset distillation: Hard truths about soft labels. In *CVPR*, 2026. 21

[15] T. Dong, B. Zhao, and L. Liu. Privacy for free: How does dataset condensation help privacy? In *ICML*, 2022. 3, 6

[16] 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. In *ICLR*, 2020. 2, 9, 20, 25

[17] J. Du, X. Zhang, J. Hu, W. Huang, and J. T. Zhou. Diversity-driven synthesis: Enhancing dataset distillation through directed weight adjustment. *NeurIPS*, 2024. 2

[18] Y. Fang, Q. Sun, X. Wang, T. Huang, X. Wang, and Y. Cao. EVA-02: A visual representation for neon genesis. *Image and Vision Computing*, 2024. 9, 20, 25

[19] Fastai. Fastai/imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french. 8, 19, 20, 21, 25

[20] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *STOC*, 2011. 2

[21] S. Fort and J. Whitaker. Direct ascent synthesis: Revealing hidden generative capabilities in discriminative models. *arXiv*, 2025. 5

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11), 2020. 3

[23] J. Gu, S. Vahidian, V. Kungurtsev, H. Wang, W. Jiang, Y. You, and Y. Chen. Efficient dataset distillation via minimax diffusion. In *CVPR*, 2024. 3, 6, 18, 19, 21

[24] Z. Guo, K. Wang, G. Cazenavette, H. Li, K. Zhang, and Y. You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. In *ICLR*, 2024. 2

[25] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, 2004. 2

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6, 9, 21, 25

[27] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2015. 2

[28] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3

[29] E. Hoogeboom, J. Heek, and T. Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, 2023. 3

[30] Hugging Face. Hugging Face: The AI community building the future. https://huggingface.co, 2016. 25

[31] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 21

[32] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *ICCV*, 2011. 10, 25

[33] M. Li, H. Gou, Y. Ma, R. Wang, K. Qin, and T. He. Fixed anchors are not enough: Dynamic retrieval and persistent homology for dataset distillation. In *CVPR*, 2026. 2

[34] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023. 3

[35] H. Liu, Y. Li, T. Xing, P. Wang, V. Dalal, L. Li, J. He, and H. Wang. Dataset distillation via the Wasserstein metric. In *ICCV*, 2025. 2, 6, 7, 8, 19, 20

[36] Z. Ma, A. Cao, F. Yang, Y. Gong, and X. Wei. Curriculum dataset distillation. *TIP*, 2025. 2, 6, 8

[37] B. Mirzasoleiman, K. Cao, and J. Leskovec. Coresets for robust training of deep neural networks against noisy labels. *NeurIPS*, 2020. 2

[38] S. Mohanty, A. Reddy, and K. R. Mopuri. DiRe: Diversity-promoting regularization for dataset condensation. In *WACV*, 2026. 2

[39] A. Mordvintsev, C. Olah, and M. Tyka. DeepDream – a code example for visualizing neural networks. Google Research Blog, jul 2015. 2

[40] B. B. Moser, S. Sarode, F. Raue, S. Frolov, K. Adamkiewicz, A. Shanbhag, J. Folz, T. C. Nauen, and A. Dengel. PRISM: Diversifying dataset distillation by decoupling architectural priors. *TMLR*, 2026. 2

[41] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. Reliable fidelity and diversity metrics for generative models. In *ICML*, 2020. 2, 8, 23

[42] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. 5

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 25

[44] M. Paul, S. Ganguli, and G. K. Dziugaite. Deep learning on a data diet: Finding important examples early in training. *NeurIPS*, 2021. 2

[45] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 9, 20, 25

[46] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. 2

[47] A. Roy, W.-Y. A. Lee, R. Chakraborty, and V. S. Lokhande. ManifoldGD: Training-free hierarchical manifold guidance for diffusion-based dataset distillation. In *CVPR*, 2026. 3, 6, 18, 19

[48] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. 2

[49] 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. 2, 6, 7, 8, 19, 20

[50] S. Shao, Z. Zhou, H. Chen, and Z. Shen. Elucidating the design space of dataset condensation. *NeurIPS*, 2024. 2, 22

[51] Z. Shen, A. Sherif, Z. Yin, and S. Shao. DELT: A simple diversity-driven EarlyLate training for dataset distillation. In *CVPR*, 2025. 2

[52] A. Shul, E. Horwitz, and Y. Hoshen. Distilling datasets into less than one image. *TMLR*, 2025. 10

[53] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski. DINOv3. *arXiv*, 2025. 1, 6, 7, 9, 18, 19, 20, 25

[54] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[55] D. Su, J. Hou, W. Gao, Y. Tian, and B. Tang. $D^4M$: Dataset distillation via disentangled diffusion model. In *CVPR*, 2024. 3, 6, 18, 19

[56] P. Sun, B. Shi, D. Yu, and T. Lin. On the diversity and realism of distilled dataset: An efficient dataset distillation paradigm. In *CVPR*, 2024. 2, 6, 7, 8, 19, 20, 21

[57] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019. 2

[58] TorchVision maintainers and contributors. TorchVision: PyTorch's computer vision library. https://github.com/pytorch/vision, 2016. 25

[59] M.-T. Tran, T. Le, X.-M. Le, T.-T. Do, and D. Phung. Enhancing dataset distillation via non-critical region refinement. In *CVPR*, 2025. 2

[60] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv*, 2025. 8, 9, 10, 20, 25

[61] H. Wang, Z. Zhao, J. Wu, Y. Shang, G. Liu, and Y. Yan. $CaO_2$: Rectifying inconsistencies in diffusion-based dataset distillation. In *ICCV*, 2025. 3, 6, 18, 19

[62] K. Wang, B. Zhao, X. Peng, Z. Zhu, S. Yang, S. Wang, G. Huang, H. Bilen, X. Wang, and Y. You. CAFE: Learning to condense dataset by aligning features. *CVPR*, 2022. 2

[63] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. In *CVPR*, 2023. 10, 25

[64] S. Wang, Y. Yang, G. Chen, P. Li, K. Li, Y. Zhou, Z. Chen, and L. Zhang. Grounding and enhancing informativeness and utility in dataset distillation. In *ICLR*, 2026. 2

[65] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. Dataset distillation. *arXiv*, 2018. 2

[66] M. Welling. Herding dynamical weights to learn. In *ICML*, 2009. 2, 10

[67] R. Wightman. PyTorch image models. `https://huggingface.co/timm`, 2019. 25

[68] M. Won, Y.-N. Hung, and D. Le. A foundation model for music informatics. In *ICASSP*, 2024. 10, 25

[69] S. Yang, Z. Cao, S. Guo, R. Zhang, P. Luo, S. Zhang, and L. Nie. Mind the boundary: Coreset selection via reconstructing the decision boundary. In *ICML*, 2024. 2

[70] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *CVPR*, 2020. 2

[71] Z. Yin and Z. Shen. Dataset distillation via curriculum data synthesis in large data era. *TMLR*, 2024. 2, 6, 7, 8, 19, 20

[72] Z. Yin, E. Xing, and Z. Shen. Squeeze, recover and relabel: Dataset condensation at ImageNet scale from a new perspective. *NeurIPS*, 2023. 2, 6, 7, 8, 19, 20

[73] X. Zhang, J. Du, P. Liu, and J. T. Zhou. Breaking class barriers: Efficient dataset distillation via inter-class feature compensator. In *ICLR*, 2025. 2

[74] B. Zhao and H. Bilen. Dataset condensation with differentiable Siamese augmentation. In *ICML*, 2021. 2

[75] B. Zhao and H. Bilen. Dataset condensation with distribution matching. *WACV*, 2023. 2

[76] B. Zhao, K. R. Mopuri, and H. Bilen. Dataset condensation with gradient matching. In *ICLR*, 2020. 2

[77] L. Zhao, Y. Wu, X. Jiang, J. Gu, Y. Wang, X. Xu, P. Zhao, and X. Lin. Taming diffusion for dataset distillation with high representativeness. In *ICML*, 2025. 3, 6, 18, 19

[78] M. Zhou, Z. Yin, S. Shao, and Z. Shen. Self-supervised dataset distillation: A good compression is all you need. *arXiv*, 2024. 2, 6, 7, 8

[79] C. Zhu, S. Suri, C. Jose, M. Oquab, M. Szafraniec, W. Wen, Y. Xiong, P. Labatut, P. Bojanowski, R. Krishnamoorthi, and V. Chandra. Efficient universal perception encoder. *arXiv*, 2026. 20, 25

# Appendix

## A Hyperparameter Sweeps

Each study below changes a single setting of our default configuration (DINOv3 backbone, noise initialization, ImageNet-1k, IPC 10) and reports the same from-scratch and feature-space metrics as Tables 1 and 2; defaults are marked †. Each table is paired with a figure showing all 10 distilled images of a single class.

**Augmentations per step.** Drifting augmented views of each image is critical. Without augmentation, every metric collapses (e.g., 0.2% hard-label accuracy and 1.0% coverage). Additionally, every metric improves monotonically as more augmented views are drifted per step, with 10 views performing best throughout (Table A.1, Figure A.1).




Figure A.1: **Effects of augmentations.** Here, we visualize the effect the number of augmentations per step has on the distilled data. As we increase the number of augmentations, the images appear to (qualitatively) become richer in semantic information. Images shown here were initialized with noise and distilled with DINOv3 to match ImageNet-1k's "orange" class. For quantitative results, please see Table A.1.

| | ResNet-18 from scratch | | | Coverage | | k-NN | | Linear | |
|---|---|---|---|---|---|---|---|---|---|
| | Hard | Soft | KD | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 |
| None | 0.2 ± 0.0 | 0.2 ± 0.0 | 1.9 ± 0.1 | 1.0 | 0.6 | 2.7 | 5.8 | 6.5 | 4.9 |
| 1 | 7.5 ± 0.3 | 27.4 ± 0.3 | 39.3 ± 0.3 | 35.8 | 28.6 | 56.0 | 50.2 | 69.5 | 59.2 |
| 2 | 15.5 ± 0.4 | 34.4 ± 0.2 | 42.4 ± 0.6 | 65.5 | 66.0 | 66.5 | 57.4 | 76.6 | 64.9 |
| 5 | 19.1 ± 0.4 | 35.4 ± 0.3 | 44.2 ± 0.4 | 75.1 | 79.9 | 69.4 | 60.3 | 77.3 | 66.1 |
| 10† | **21.3** ± 0.5 | **35.9** ± 0.1 | **45.0** ± 0.2 | **77.0** | **83.4** | **69.9** | **61.1** | **77.5** | **66.3** |

Table A.1: **Augmented copies per step.** The number of augmented views of each distilled image used at every optimization step. *None* is a single deterministic resized view. Performance monotonically increases with the number of augmentations, with diminishing returns as we approach 10.

**Backbone size.** A larger backbone does not necessarily lead to better data. DINOv3 ViT-B is best or tied on every metric, while ViT-L matches it only in the knowledge distillation setting and ViT-S covers notably less of the distribution (Table A.2, Figure A.2). However, the ViT-L variant could possibly perform better with more hyperparameter tuning.




Figure A.2: **Effect of backbone size.** All 10 distilled images of the ImageNet-1k "lorikeet" class using DINOv3 ViT-S, ViT-B (default), and ViT-L. For quantitative results, see Table A.2.

| | ResNet-18 from scratch | | | Coverage | | k-NN | | Linear | |
|---|---|---|---|---|---|---|---|---|---|
| | Hard | Soft | KD | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 |
| Small | 18.5 ± 0.5 | 31.9 ± 0.5 | 41.9 ± 0.5 | 61.8 | 77.4 | 67.6 | 60.9 | 77.2 | 65.3 |
| Base† | **21.3** ± 0.5 | **35.9** ± 0.1 | **45.0** ± 0.2 | **77.0** | **83.4** | **69.9** | **61.1** | **77.5** | **66.3** |
| Large | 18.3 ± 0.3 | 34.0 ± 0.4 | **44.6** ± 0.4 | 75.6 | 78.1 | 69.8 | 60.2 | 76.9 | 65.6 |

Table A.2: **Backbone model size.** Base size performs the best with our settings, but the larger model might improve with some tuning.

**Kernel temperature.** A small temperature ($\tau = 0.05$) slightly improves $k$-NN accuracy but substantially lowers soft-label accuracy and coverage, while a large temperature ($\tau = 5$) matches the default in the soft-label and knowledge distillation settings but lowers hard-label accuracy and coverage. Our default ($\tau = 0.5$) gives the best balance (Table A.3, Figure A.3).




Figure A.3: **Effect of kernel temperature.** All 10 distilled images of the ImageNet-1k "terrapin" class for three drifting kernel temperatures $\tau$ (default $\tau = 0.5$). For quantitative results, see Table A.3.

| | ResNet-18 from scratch | | | Coverage | | k-NN | | Linear | |
|---|---|---|---|---|---|---|---|---|---|
| | Hard | Soft | KD | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 |
| $\tau = 0.05$ | 17.9 ± 0.2 | 26.4 ± 0.1 | 39.6 ± 0.2 | 71.6 | 66.6 | **72.7** | **62.7** | 76.9 | 64.4 |
| $\tau = 0.5$† | **21.3** ± 0.5 | **35.9** ± 0.1 | **45.0** ± 0.2 | **77.0** | **83.4** | 69.9 | 61.1 | **77.5** | **66.3** |
| $\tau = 5$ | 17.7 ± 0.1 | **35.9** ± 0.4 | **45.1** ± 0.5 | 64.7 | 70.7 | 66.2 | 57.8 | 76.1 | 65.0 |

Table A.3: **Drifting distance kernel temperature.** The method remains decently robust to a higher temperature, but a lower one significantly hurts performance.

**Drifted features.** Drifting the CLS token alone already yields usable data, but adding the mean patch token and, even more so, *every* patch token steadily improves from-scratch accuracy (11.8% $\rightarrow$ 15.0% $\rightarrow$ 21.3% hard-label) and coverage, while $k$-NN and linear accuracy remain nearly unchanged (Table A.4, Figure A.4).




Figure A.4: **Effect of the drifted features.** All 10 distilled images of the ImageNet-1k "golden retriever" class when the drifting loss is applied to the CLS token only, the CLS token plus the mean patch token, or the CLS token plus every patch token (default). For quantitative results, see Table A.4.

| | ResNet-18 from scratch | | | Coverage | | k-NN | | Linear | |
|---|---|---|---|---|---|---|---|---|---|
| | Hard | Soft | KD | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 |
| CLS | 11.8 $\pm$ 0.2 | 26.4 $\pm$ 0.0 | 41.3 $\pm$ 0.3 | 70.2 | 72.5 | 69.4 | **61.8** | 76.7 | 66.0 |
| +mean | 15.0 $\pm$ 0.3 | 31.0 $\pm$ 0.3 | 43.6 $\pm$ 0.6 | 74.2 | 80.0 | 69.3 | 61.3 | 77.4 | **66.6** |
| +grid† | **21.3** $\pm$ 0.5 | **35.9** $\pm$ 0.1 | **45.0** $\pm$ 0.2 | **77.0** | **83.4** | **69.9** | 61.1 | **77.5** | 66.3 |

Table A.4: **Drifted features.** The DINOv3 tokens to which the drifting loss is applied: the CLS token only, the CLS token plus the mean patch token (+mean), or the CLS token plus every patch token (+grid). Drifting the CLS token with the full grid performs best.

**Parameterization.** Both components of our parameterization matter. Removing color decorrelation lowers hard-label accuracy from 21.3% to 16.5% and SigLIP2 coverage from 77.0% to 62.0%, and replacing the Laplacian pyramid with raw pixels collapses from-scratch accuracy (3.3% hard-label) and coverage (41.9%). Notably, $k$-NN and linear accuracy barely change in either case, again showing that these metrics are much less sensitive to the quality of the distilled data than from-scratch training or coverage (Table A.5, Figure A.5).




Figure A.5: **Effect of the parameterization.** All 10 distilled images of the ImageNet-1k "flamingo" class with our full parameterization, without color decorrelation (−color), and with raw pixels instead of a Laplacian pyramid (−pyramid). For quantitative results, see Table A.5.

| | ResNet-18 from scratch | | | Coverage | | k-NN | | Linear | |
|---|---|---|---|---|---|---|---|---|---|
| | Hard | Soft | KD | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 | SigLIP2 | MoCo-v3 |
| Full† | **21.3** $\pm$ 0.5 | **35.9** $\pm$ 0.1 | **45.0** $\pm$ 0.2 | **77.0** | **83.4** | **69.9** | **61.1** | **77.5** | **66.3** |
| −color | 16.5 $\pm$ 0.5 | 31.4 $\pm$ 0.8 | 38.1 $\pm$ 0.8 | 62.0 | 76.5 | 69.7 | 60.6 | 77.4 | 65.7 |
| −pyramid | 3.3 $\pm$ 0.3 | 11.0 $\pm$ 1.0 | 24.7 $\pm$ 1.0 | 41.9 | 46.4 | 69.2 | 59.7 | 77.2 | 62.1 |

Table A.5: **Parameterization.** Our full parameterization, without color decorrelation (−color), and with raw pixels in place of the Laplacian pyramid (−pyramid). The full parameterization performs best.

| Method | Hard-Label | | Soft-Label | | Knowledge Distillation | |
|---|---|---|---|---|---|---|
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Noise init* | | | | | | |
| ResNet-18 | 17.6 ± 0.2 | 18.4 ± 0.5 | 30.5 ± 0.8 | 43.1 ± 0.3 | 44.8 ± 0.7 | 55.0 ± 0.3 |
| ViT | 20.7 ± 0.4 | 28.4 ± 0.6 | 34.0 ± 0.6 | 48.7 ± 0.1 | 44.2 ± 0.2 | 54.9 ± 0.2 |
| CLIP | 6.1 ± 0.2 | 8.4 ± 0.3 | 18.5 ± 0.2 | 29.2 ± 0.5 | 32.9 ± 0.3 | 44.3 ± 0.4 |
| SigLIP2 | 14.9 ± 0.2 | 21.0 ± 0.0 | 30.7 ± 0.5 | 44.5 ± 0.7 | 40.8 ± 0.3 | 52.2 ± 0.6 |
| EVA-02 | 16.8 ± 0.3 | 24.7 ± 0.2 | 31.5 ± 0.3 | 45.9 ± 0.5 | 42.8 ± 0.9 | 53.8 ± 0.1 |
| MoCo-v3 | **27.0** ± 0.2 | **33.5** ± 1.1 | **40.3** ± 0.1 | **52.8** ± 0.0 | **48.5** ± 0.1 | **57.2** ± 0.3 |
| DINOv3 | 21.3 ± 0.5 | 30.2 ± 0.2 | 35.9 ± 0.1 | 50.7 ± 0.1 | 45.0 ± 0.2 | 56.0 ± 0.2 |
| *Real init* | | | | | | |
| ResNet-18 | 20.4 ± 0.4 | 22.0 ± 0.4 | 33.9 ± 0.2 | 45.4 ± 0.5 | 47.8 ± 0.2 | 56.4 ± 0.3 |
| ViT | 25.6 ± 0.2 | 37.6 ± 0.4 | 39.8 ± 0.5 | 52.9 ± 0.2 | 48.8 ± 0.3 | 57.6 ± 0.1 |
| CLIP | 20.9 ± 0.2 | 36.5 ± 0.5 | 35.5 ± 0.2 | 49.1 ± 0.2 | 47.3 ± 0.2 | 55.9 ± 0.2 |
| SigLIP2 | 21.9 ± 0.2 | 32.3 ± 0.3 | 37.1 ± 0.3 | 50.0 ± 0.2 | 47.6 ± 0.1 | 56.4 ± 0.2 |
| EVA-02 | 24.1 ± 0.3 | 37.7 ± 0.2 | 38.1 ± 0.1 | 51.8 ± 0.4 | 48.4 ± 0.1 | 57.1 ± 0.3 |
| MoCo-v3 | **28.9** ± 0.2 | 36.0 ± 0.7 | **41.7** ± 0.0 | 53.4 ± 0.1 | **49.6** ± 0.2 | 57.7 ± 0.2 |
| DINOv3 | 26.0 ± 0.2 | **41.1** ± 0.4 | 40.5 ± 0.0 | **54.3** ± 0.4 | **49.5** ± 0.3 | **58.1** ± 0.1 |

Table B.1: **From-scratch training across distillation backbones.** Most backbones yield distilled data that can effectively train a model from scratch, with MoCo-v3 [7] and DINOv3 [53] eking out a slight edge over the others.

| Method | Hard-Label | | Soft-Label | | Knowledge Distillation | |
|---|---|---|---|---|---|---|
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Diffusion* | | | | | | |
| Minimax [23] | 22.2 ± 0.2 | 36.1 ± 0.3 | 35.9 ± 0.2 | 48.8 ± 0.2 | 49.7 ± 0.1 | 58.1 ± 0.1 |
| D4M [55] | 17.0 ± 0.1 | 30.5 ± 0.2 | 35.8 ± 0.2 | 50.3 ± 0.1 | 49.6 ± 0.3 | 58.5 ± 0.1 |
| $D^3HR$ [77] | 23.7 ± 0.2 | 38.0 ± 0.4 | 35.7 ± 0.1 | 50.2 ± 0.2 | 48.1 ± 0.3 | 57.7 ± 0.1 |
| $CaO_2$ [61] | **28.8** ± 0.2 | **44.3** ± 0.1 | 40.7 ± 0.2 | 54.1 ± 0.1 | **51.4** ± 0.3 | 59.5 ± 0.1 |
| ManifoldGD [47] | 27.6 ± 0.3 | **43.8** ± 0.4 | 41.2 ± 0.2 | **54.6** ± 0.0 | **51.5** ± 0.1 | **59.8** ± 0.1 |
| *Ours, noise init* | | | | | | |
| ResNet-18 | 17.6 ± 0.2 | 18.4 ± 0.5 | 30.5 ± 0.8 | 43.1 ± 0.3 | 44.8 ± 0.7 | 55.0 ± 0.3 |
| MoCo-v3 | 27.0 ± 0.2 | 33.5 ± 1.1 | 40.3 ± 0.1 | 52.8 ± 0.0 | 48.5 ± 0.1 | 57.2 ± 0.3 |
| DINOv3 | 21.3 ± 0.5 | 30.2 ± 0.2 | 35.9 ± 0.1 | 50.7 ± 0.1 | 45.0 ± 0.2 | 56.0 ± 0.2 |
| *Ours, real init* | | | | | | |
| ResNet-18 | 20.4 ± 0.4 | 22.0 ± 0.4 | 33.9 ± 0.2 | 45.4 ± 0.5 | 47.8 ± 0.2 | 56.4 ± 0.3 |
| MoCo-v3 | **28.9** ± 0.2 | 36.0 ± 0.7 | **41.7** ± 0.0 | 53.4 ± 0.1 | 49.6 ± 0.2 | 57.7 ± 0.2 |
| DINOv3 | 26.0 ± 0.2 | 41.1 ± 0.4 | 40.5 ± 0.0 | **54.3** ± 0.4 | 49.5 ± 0.3 | 58.1 ± 0.1 |

Table B.2: **Comparison with Diffusion-Based Methods.** Our method matches the performance of the *best* diffusion-based methods in the soft-label setting while falling slightly behind on hard-label and knowledge distillation. Despite not leveraging a pre-trained generative model, our distilled data remain competitive.

# B More Datasets and Backbones

Table B.1 extends Table 1 to additional distillation backbones. Table B.2 compares against diffusion-based methods, which we exclude from the main text because they rely on a generative model pre-trained on the target data; Table B.3 makes the same comparison in feature space. Tables B.4 and B.5 report from-scratch results on ImageNette and ImageWoof using their official splits. Finally, Table B.6 scores each of our distilled sets in every other backbone's latent space.

| Method | Coverage | | | | $k$-NN | | | | Linear | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SigLIP2 | | MoCo-v3 | | SigLIP2 | | MoCo-v3 | | SigLIP2 | | MoCo-v3 | |
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Diffusion* | | | | | | | | | | | | |
| Minimax [23] | 60.6 | 82.3 | 67.6 | 88.2 | 67.2 | 70.3 | 57.9 | 61.4 | 71.9 | 74.4 | 61.2 | 64.4 |
| D4M [55] | 37.1 | 60.3 | 47.1 | 76.3 | 57.8 | 62.3 | 48.1 | 53.5 | 65.6 | 67.7 | 55.2 | 62.2 |
| $D^3HR$ [77] | 55.0 | 78.9 | 65.0 | 88.5 | 67.9 | 71.0 | 59.4 | 62.7 | 72.7 | 75.1 | 62.4 | 66.2 |
| $CaO_2$ [61] | 72.9 | 92.3 | 71.2 | 94.1 | 67.3 | 70.2 | 58.5 | 61.1 | 73.1 | 75.9 | 62.8 | 66.9 |
| ManifoldGD [47] | 77.4 | 94.7 | 75.1 | 94.9 | 68.5 | 72.1 | 59.4 | 62.9 | 74.5 | 77.2 | 63.9 | 68.1 |
| *Ours, noise init* | | | | | | | | | | | | |
| ResNet-18 | 19.7 | 38.4 | 41.3 | 66.3 | 54.3 | 61.2 | 55.3 | 58.7 | 69.6 | 71.2 | 59.8 | 66.5 |
| MoCo-v3 | 61.2 | 86.9 | (91.9) | (97.9) | 64.3 | 68.4 | (60.2) | (62.5) | 74.6 | 74.6 | (66.6) | (69.9) |
| DINOv3 | 77.0 | 93.2 | 83.4 | 96.3 | 69.9 | 72.5 | **61.1** | **63.2** | **77.5** | **78.6** | **66.3** | **70.3** |
| *Ours, real init* | | | | | | | | | | | | |
| ResNet-18 | 30.0 | 57.5 | 54.9 | 82.0 | 62.1 | 68.3 | 58.0 | 61.3 | 72.8 | 74.5 | 63.1 | 67.8 |
| MoCo-v3 | 60.6 | 88.5 | (91.3) | (98.0) | 65.0 | 69.3 | (60.2) | (62.2) | 75.0 | 75.8 | (66.4) | (70.0) |
| DINOv3 | **79.9** | **95.5** | **85.1** | **97.5** | **69.9** | **72.8** | 60.3 | 63.0 | 77.0 | 78.5 | 65.7 | 70.2 |

Table B.3: **Feature-Space Comparison with Diffusion-Based Methods.** Unlike in the from-scratch regime, our method (DINOv3 [53] variant) consistently outperforms the diffusion-based methods across all feature-space metrics. This is quite remarkable given that our method does not use a generative model to constrain the distilled data to the real image manifold.

| Method | Hard-Label | | Knowledge Distillation | |
|---|---|---|---|---|
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Noise init* | | | | |
| $SRe^2L$ [72] | 10.6 ± 0.3 | 15.9 ± 0.7 | 77.6 ± 1.0 | 91.8 ± 0.3 |
| CDA [71] | 17.3 ± 0.8 | 32.7 ± 0.3 | 75.7 ± 2.2 | 91.0 ± 0.3 |
| G-VBSM [49] | 23.4 ± 1.2 | 21.5 ± 0.5 | **82.8** ± 0.4 | **93.6** ± 0.3 |
| WMDD [35] | 42.4 ± 1.2 | 55.7 ± 0.1 | 81.2 ± 0.7 | 92.4 ± 0.1 |
| Ours (ResNet-18) | **56.2** ± 0.4 | 70.7 ± 0.7 | **82.7** ± 0.6 | 92.4 ± 0.3 |
| Ours (DINOv3) | **56.4** ± 0.8 | **71.9** ± 0.3 | 79.9 ± 0.3 | 92.2 ± 0.3 |
| *Real init* | | | | |
| RDED [56] | 42.8 ± 0.8 | 64.6 ± 0.7 | 78.7 ± 0.2 | **92.8** ± 0.2 |
| FADRM [8] | 50.4 ± 1.2 | 72.6 ± 0.5 | **83.9** ± 0.6 | **93.2** ± 0.3 |
| Ours (ResNet-18) | **60.9** ± 0.6 | **74.7** ± 1.1 | **84.5** ± 0.9 | **92.9** ± 0.4 |
| Ours (DINOv3) | **61.3** ± 0.6 | **75.9** ± 0.3 | **83.4** ± 1.0 | **93.1** ± 0.4 |

Table B.4: **ImageNette From-Scratch Training.** On the "easy" 10-class ImageNette [19] dataset, we outperform the baselines in the hard-label setting and match or beat them in the knowledge distillation regime.

| Method | Hard-Label | | Knowledge Distillation | |
|---|---|---|---|---|
| | IPC 10 | IPC 50 | IPC 10 | IPC 50 |
| *Noise init* | | | | |
| SRe$^2$L [72] | 15.3 ± 0.3 | 25.6 ± 0.5 | 56.7 ± 0.6 | 80.6 ± 0.3 |
| CDA [71] | 15.8 ± 0.1 | 24.2 ± 0.6 | 57.1 ± 0.6 | 80.8 ± 0.2 |
| G-VBSM [49] | 18.7 ± 0.8 | 26.8 ± 0.5 | 62.3 ± 0.2 | **84.1** ± 0.3 |
| WMDD [35] | 16.7 ± 0.2 | 20.7 ± 0.9 | 33.3 ± 0.9 | 65.9 ± 1.4 |
| Ours (ResNet-18) | 34.2 ± 1.0 | 53.2 ± 0.2 | **65.6** ± 0.7 | 81.7 ± 0.6 |
| Ours (DINOv3) | **37.7** ± 0.4 | **57.8** ± 0.8 | **66.5** ± 0.9 | **83.3** ± 0.6 |
| *Real init* | | | | |
| RDED [56] | 14.6 ± 0.9 | 27.2 ± 1.2 | 60.4 ± 0.7 | **83.7** ± 0.5 |
| FADRM [8] | 25.0 ± 0.8 | 41.6 ± 0.6 | 61.2 ± 2.2 | 81.9 ± 0.6 |
| Ours (ResNet-18) | **40.1** ± 0.8 | 56.6 ± 1.5 | **66.3** ± 1.2 | 82.1 ± 0.4 |
| Ours (DINOv3) | **40.3** ± 0.7 | **59.5** ± 1.0 | **68.4** ± 1.4 | **84.3** ± 0.3 |

Table B.5: **ImageWoof From-Scratch Training.** On the "hard" 10-class ImageWoof [19] dataset, we also outperform the baselines in the hard-label setting and match or beat them in the knowledge distillation regime.

| Distilled set | Feature space | | | | | | |
|---|---|---|---|---|---|---|---|
| | ViT | CLIP | SigLIP2 | EVA-02 | MoCo-v3 | EuPE | DINOv3 |
| ViT [16] | (94.6) | 59.9 | 50.6 | 57.2 | 69.9 | 41.7 | 39.6 |
| CLIP [45] | 29.0 | (77.2) | 50.4 | 59.1 | 20.4 | 41.3 | 28.6 |
| SigLIP2 [60] | 73.5 | 84.7 | (78.0) | 84.8 | 56.9 | 71.5 | 56.1 |
| EVA-02 [18] | 76.2 | 86.2 | 70.5 | (91.8) | 58.7 | 76.5 | 64.6 |
| MoCo-v3 [7] | 89.5 | 72.2 | 61.2 | 75.7 | (91.9) | 52.3 | 48.2 |
| EuPE [79] | 82.1 | 85.5 | 75.8 | 90.4 | 78.2 | (81.8) | 67.3 |
| DINOv3 [53] | 82.8 | 86.1 | 77.0 | 89.6 | 83.4 | 84.1 | (95.1) |

Table B.6: **Inter-Model Coverage.** Coverage of our noise-initialized ImageNet-1k distilled sets (IPC 10), distilled with the backbone in each row and evaluated in the feature space of the backbone in each column. Some encoders yield data that generalize very well (e.g., DINOv3) while data yielded by other encoders seem very overfit to the source model (e.g., CLIP). Same-model performance is indicated by (parentheses).

# C More Method Details

**Note on from-scratch evaluation:** As pointed out in recent work [14], the most popular large-scale dataset distillation evaluation method effectively changes the task to *knowledge distillation* rather than training from scratch. While knowledge distillation is an interesting task deserving of its own attention, we wish to focus on from-scratch training, the original goal of dataset distillation. For our soft-label evaluations, rather than generating new soft labels at every step, we use a teacher model to generate exactly one soft label per image before training by averaging its temperature-softened predictions over 10 random crops. We re-evaluate prior works under this training paradigm as well.

### Distillation

For our real-initialized variants, we initialize the distilled images with those of RDED [56] and distill for 5000 iterations. Our noise-initialized variants start from random noise and are distilled for 10000 iterations. We use all real data as positive samples at every step (this is feasible even on small GPUs with $\sim$1.3k real embeddings per class).

To optimize our distilled data, we use Adam [31] with a learning rate of 0.001.

Images are distilled at $256 \times 256$ resolution. Image pyramids are initialized with the first 4 layers (up to $8 \times 8$) as standard Gaussian noise. A new layer is added every 200 iterations (as standard Gaussian noise) until the full resolution is reached.

Videos are distilled at $256 \times 256$ spatial resolution and 16 frames. Video pyramids are initialized with the first 4 layers and 1 frame as standard Gaussian noise. A new layer with twice the spatial and temporal resolution is added every 200 iterations (as standard Gaussian noise) until the full resolution is reached.

Audio pyramids are initialized as just the first layer. A new layer is added every 100 steps until the final resolution of 524,288 samples is reached.

**Drifting field implementation.** For clarity, Section 3.2 presents the drifting field in its simplest form. In practice, we follow the full implementation of [13]: the kernel is additionally normalized over the synthetic-sample axis (their Algorithm 2), and we apply the feature and drift normalizations of their Appendix A.6. Concretely, each feature $\phi_j$ is rescaled by a stop-gradient scale $S_j$ such that the average pairwise distance is approximately $\sqrt{C_j}$ (where $C_j$ is the feature dimension), the kernel temperature is correspondingly scaled to $\tau\sqrt{C_j}$, and the resulting field $\mathbf{V}_j$ is divided by its root-mean-square per-dimension magnitude. A single normalization scale is shared across all spatial locations of a given feature map, and the total loss is the sum of the per-feature losses.

**Features.** The feature extractor $\phi$ yields several features $\phi_j$ per sample. For ViT backbones, we use the CLS token together with all patch tokens. For ResNet-18, we use the final global-average-pooled vector together with the per-location feature vectors of the final two convolutional layers. For both audio and video methods, we drift the mean token at each layer.

# D Evaluation Protocols

All from-scratch evaluations train a ResNet-18 [26] student and report top-1 accuracy of the final-epoch model on the ImageNet-1k validation set (single $224 \times 224$ center crop after resizing to 256). All numbers are the mean and standard deviation over 3 training seeds. Every method, including ours, is re-evaluated with identical recipes.

## D.1 Hard-Label Training

On ImageNet-1k [12], hard-label training uses the soft-label recipe below with $\alpha = 0$ (i.e., only the label-smoothed cross-entropy on the one-hot labels), so the two settings differ only in the targets.

On ImageNette and ImageWoof [19], we use the evaluation protocol from Minimax Diffusion [23]:

- `net_type = resnet`
- `depth = 18`

- `norm_type = instance`
- `optimizer = SGD`
- `lr = 0.01`
- `momentum = 0.9`
- `weight_decay = 5e-4`
- `epochs (E)`
  - `IPC = 1  → E = 3000`
  - `IPC = 10 → E = 2000`
  - `IPC = 50 → E = 1500`
- `lr_schedule = MultiStepLR`
  - `milestones = [2·E/3, 5·E/6]`
  - `gamma = 0.2`
- `batch_size = 64`

### D.2 Soft-Label Training

We implement our own recipe for soft-label training and apply it equally to our method and all baselines. Before training, a pre-trained ResNet-18 teacher (torchvision weights) computes **exactly one soft label per distilled image**: the average of its temperature-scaled softmax outputs ($T = 10$) over 10 random-resized crops (scale $[0.3, 1]$) with horizontal flips. The student is trained with

$$\mathcal{L} = \alpha\, T^2\, \mathrm{CE}(\mathrm{softmax}(z/T), y_{\text{soft}}) + (1-\alpha)\, \mathrm{CE}_{\text{ls}=0.1}(z, y_{\text{hard}})$$

where $z$ are the student logits and $\alpha = 0.9$. Other hyperparameters are as follows:

- `student = resnet18 (batch norm)`
- `optimizer = AdamW`
- `lr = 5e-4`
- `lr_schedule = 20-epoch linear warmup, then cosine to 1e-5`
- `weight_decay = 0.5 → 0.1 (cosine)`
- `epochs = 1000`
- `batch_size = 50`
- `ema_decay = 0.9995`
- `augmentation`
  - `RandomResizedCrop(min_scale=0.3)`
  - `RandomHorizontalFlip()`
  - `TrivialAugmentWide()`
  - `MixUp(alpha=0.2, p=0.5)`

### D.3 Knowledge Distillation

Rather than from-scratch training, most existing works on large-scale dataset distillation (i.e., ImageNet-1k for ResNet-18) have evaluated using a **knowledge distillation** protocol. Rather than using a (hard or soft) fixed label for each distilled image, the knowledge distillation protocol uses a teacher model to recompute a new label at every training step using a newly augmented version of the image. For our knowledge distillation evaluations, we use the recipe defined in EDC [50].

The method uses a set of $M$ teacher models $f_m$ to compute a soft target for each augmented sample $x$ as the average of each model's raw output logits:

$$y_{\text{soft}} = \frac{1}{M} \sum_m f_m(x)$$

The total knowledge distillation loss is then

$$\mathcal{L}_{\text{KD}} = \text{MSE}(z, y_{\text{soft}}) + \lambda_{\text{CE}}\text{CE}(z, y_{\text{hard}})$$

where $z$ are the output logits of the student model, MSE is the mean-squared-error loss, CE is the cross-entropy loss, $y_{\text{hard}}$ is the class-specified one-hot label, and $\lambda_{\text{CE}}$ is the cross-entropy weight.

Other hyperparameters are as follows:

- `student = resnet18`
- `teachers`
  - `ImageNet-1k → 4-model EDC ensemble`
    - `resnet18`
    - `mobilenet_v2`
    - `shufflenet_v2_x0_5`
    - `alexnet`
  - `ImageNette/ImageWoof → resnet18`
- `ce_weight = 0.025`
- `optimizer = AdamW`
- `lr = 1e-3`
- `weight_decay = 0.01`
- `lr_schedule = cosine`
- `lr_schedule_deceleration = 2`
- `ema_decay = 0.99`
- `augmentation`
  - `RandomResizedCrop(min_scale=0.5)`
  - `RandomHorizontalFlip()`
  - `RandAugment(n=2, m=9)  (ImageNette and ImageWoof only)`
  - `CutMix(alpha=1.0)`
- `batch_size`
  - `ImageNet-1k → 100`
  - `ImageNette/ImageWoof → 50`
- `epochs (E)`
  - `ImageNet-1k → E = 300`
  - `ImageNette/ImageWoof → E = 1000`

### D.4 Coverage

For a set of distilled points, we compute the coverage [41] by measuring the fraction of real **test set** points that have at least one distilled point within the ball defined by their $k$-th nearest real neighbor in the latent space of some encoder. We use cosine distance and $k = 5$ by default. We compute the coverage per class and report the average. The pool of distilled points includes each distilled image's center crop together with 10 random-resized crops (scale $[0.08, 1]$) per image.

### D.5 $k$-Nearest Neighbors

We compute the $k$-Nearest Neighbor ($k$-NN) performance of a distilled dataset in the typical fashion; for each real test point, we compute the $k$ nearest distilled points (in the space of some encoder) and predict the most common label among those $k$ neighbors. By default, we use cosine distance and $k = 1$. As with coverage, the pool of distilled points includes each distilled image's center crop together with 10 random-resized crops (scale $[0.08, 1]$) per image.

### D.6 Linear Classification

We train a linear classifier on frozen, unnormalized features of an augmented bank: each distilled image's center crop together with 10 random-resized crops (scale $[0.7, 1]$), sampled uniformly. We use the following parameters:

- `optimizer = AdamW`
- `lr = 1e-3`
- `lr_schedule = cosine annealing to 0 over all steps (stepped per step)`
- `weight_decay = 0`
- `batch_size = 256 (sampled uniformly with replacement)`
- `steps: IPC = 10 → 4000, IPC = 50 → 19600`

## E Algorithms

**Algorithm 1** Distillation loss for one class (one optimization step)

**Require:** Pyramids of the synthetic samples $\{\mathbf{x}_s\}_{s=1}^{|\mathcal{S}|}$, real images of the class $\{\mathbf{y}_k\}_{k=1}^{M}$, features $\{\phi_j\}$, augmentation distribution $P_A$, augmentations per step $n$

Decode each $\mathbf{x}_s$ from its pyramid ▷ Eq. 9
Sample $A_1, \dots, A_n \sim P_A$
**for** each feature $\phi_j$ **do**
  $\tilde{\mathbf{x}}_{a,s} \leftarrow \phi_j(A_a(\mathbf{x}_s))$ for all $a, s$ ▷ $N = n|\mathcal{S}|$ embeddings
  $\tilde{\mathbf{y}}_k \leftarrow \phi_j(\mathbf{y}_k)$ ▷ un-augmented; precomputed
  $\mathbf{V}, S \leftarrow \textsc{DriftField}\big(\texttt{stopgrad}(\tilde{\mathbf{x}}), \tilde{\mathbf{y}}\big)$ ▷ Algorithm 2
  $\mathcal{L}_j \leftarrow \mathrm{MSE}\left(\tilde{\mathbf{x}}/S,\ \texttt{stopgrad}(\tilde{\mathbf{x}}/S + \mathbf{V})\right)$
**end for**
**return** $\mathcal{L} = \sum_j \mathcal{L}_j$ ▷ back-propagated to the pyramids

**Algorithm 2** DRIFTFIELD: the drifting field for one feature $\phi_j$, following Alg. 2 and App. A.6 of [13]. All steps are applied independently at each spatial location of $\phi_j$; only $S$ and $\lambda_\tau$ are computed jointly over all locations.

**Require:** Synthetic embeddings $\{\tilde{\mathbf{x}}_i\}_{i=1}^{N}$ (detached), real embeddings $\{\tilde{\mathbf{y}}_k\}_{k=1}^{M}$, feature dimension $C$, temperatures $\mathcal{T}$
**Ensure:** Normalized drifting field $\mathbf{V}(\tilde{\mathbf{x}}_i)$ for $i = 1, \dots, N$, feature scale $S$

$d^+_{ik} \leftarrow \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_k\|_2, \quad d^-_{il} \leftarrow \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_l\|_2$
$S \leftarrow \texttt{stopgrad}\big(\,\mathrm{mean}(\{d^+_{ik}\} \cup \{d^-_{il}\})\big)/\sqrt{C}$ ▷ feature normalization
Divide all $d$, $\tilde{\mathbf{x}}$, and $\tilde{\mathbf{y}}$ by $S$
$d^-_{ii} \leftarrow \infty$ ▷ mask self-pairs
$\mathbf{V}(\tilde{\mathbf{x}}_i) \leftarrow \mathbf{0}$
**for** $\tau \in \mathcal{T}$ **do**
  $\ell_{i,\cdot} \leftarrow -\big[\, d^+_{i,1:M} \,\|\, d^-_{i,1:N} \,\big]/(\tau\sqrt{C})$ ▷ logits over all targets
  $A \leftarrow \sqrt{\mathrm{softmax}_{\text{targets}}(\ell) \odot \mathrm{softmax}_i(\ell)}$ ▷ normalize over both axes
  Split $A = [A^+ \,\|\, A^-]$ into positive and negative columns
  $W^+_{ik} \leftarrow A^+_{ik} \sum_l A^-_{il}, \quad W^-_{il} \leftarrow A^-_{il} \sum_k A^+_{ik}$
  $\mathbf{V}_\tau(\tilde{\mathbf{x}}_i) \leftarrow \sum_k W^+_{ik}\, \tilde{\mathbf{y}}_k - \sum_l W^-_{il}\, \tilde{\mathbf{x}}_l$
  $\lambda_\tau \leftarrow \sqrt{\mathrm{mean}_i\, \|\mathbf{V}_\tau(\tilde{\mathbf{x}}_i)\|_2^2 / C}$ ▷ drift normalization
  $\mathbf{V}(\tilde{\mathbf{x}}_i) \leftarrow \mathbf{V}(\tilde{\mathbf{x}}_i) + \mathbf{V}_\tau(\tilde{\mathbf{x}}_i)/\lambda_\tau$
**end for**
**return** $\{\mathbf{V}(\tilde{\mathbf{x}}_i)\}_{i=1}^{N}, S$

# F Limitations, Compute, Broader Impacts, and Licenses

### Limitations

Our largest limitation is the time needed to distill the data. For 10 images per class, each class takes $\sim$0.25 GPU hours (more or less depending on GPU model), corresponding to roughly 250 GPU hours for ImageNet-1k before accounting for hyperparameter sweeps, multiple backbones, and larger IPC settings. Although distilling each class separately makes the process easy to parallelize, the total compute cost is significant.

### Compute Resources

We ran our experiments on a shared cluster using whatever GPUs we could get. Used GPUs include RTX 3090s, RTX 4090s, L40Ss, V100s, A100s, H100s, and H200s. All image experiments could comfortably fit on a 24 GB GPU. Audio and video experiments required an A100, H100, or H200.

### Broader Impacts

Our method does not come with any readily apparent negative effects. Advances in dataset distillation may, in fact, lead to more efficient training, thereby shrinking the AI industry's carbon footprint.

### Licenses

**Datasets.** For ImageNet [12] we build on the Torchvision [58] implementation as part of the PyTorch [1, 43] ecosystem. ImageNet has a unique non-commercial license, to which we have agreed. The remaining datasets are used as follows:

- **ImageNette and ImageWoof** [19]: the official `imagenette2` and `imagewoof2` releases (Apache-2.0); both are subsets of ImageNet and are also subject to the ImageNet terms above.
- **FMA** [11]: the `fma_small` subset. Its metadata is released under CC BY 4.0, and each audio track is distributed under the Creative Commons license chosen by its artist; the dataset is intended for research use.
- **HMDB51** [32]: released under CC BY 4.0.
- **Paintings**: the Claude Monet and Qi Baishi paintings in Section 4.5 were collected from WikiArt, the Art Institute of Chicago, and the Metropolitan Museum of Art. Images from the Art Institute of Chicago and the Met are public-domain works released under CC0; WikiArt presents public-domain works and, under a fair-use rationale, copyrighted works. Monet's paintings are in the public domain. We use all paintings solely for non-commercial research.

**Models.** All backbones are used with their publicly released pre-trained weights and are kept frozen.

- **ResNet-18** [26] and **ViT-B/16** [16]: Torchvision [58] `IMAGENET1K_V1` weights (BSD-3-Clause). The ResNet-18, MobileNetV2, ShuffleNetV2, and AlexNet teachers used for soft labels and knowledge distillation also come from Torchvision (BSD-3-Clause).
- **CLIP** [45]: `openai/clip-vit-base-patch16` on Hugging Face [30], released by OpenAI under the MIT license.
- **SigLIP2** [60]: `google/siglip2-base-patch16-224` (Apache-2.0).
- **EVA-02** [18]: `eva02_base_patch14_224.mim_in22k` from PyTorch Image Models [67] (MIT).
- **MoCo-v3** [7]: ViT-B weights loaded from `nyu-visionx/moco-v3-vit-b`; the official MoCo-v3 release is licensed under CC BY-NC 4.0.
- **EuPE** [79]: `facebook/EUPE-ViT-B` (FAIR Noncommercial Research License).
- **DINOv3** [53]: `facebook/dinov3-vitb16-pretrain-lvd1689m`, along with the ViT-S/16 and ViT-L/16 variants for the model-size ablation (DINOv3 License).
- **VideoMAE-v2** [63]: `OpenGVLab/VideoMAEv2-Base` (CC BY-NC 4.0).
- **MusicFM** [68]: the FMA checkpoint from `minzwon/MusicFM` (MIT).

## 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: We support all claims made in the abstract and introduction.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: There is a "Limitations" section in the appendix.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: Our paper contains no theoretical results.

Guidelines:

- The answer [N/A] 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: We describe the method and experiments in the main text and provide further details in the appendix.

   Guidelines:

   - The answer [N/A] 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: We include a link to the code on our project page.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Details are given in the appendix.

   Guidelines:

   - The answer [N/A] 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: We include $\pm$ stdev where available and applicable.

   Guidelines:

   - The answer [N/A] 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: This information is given in the appendix.

   Guidelines:

   - The answer [N/A] 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 abide by the code of ethics.

   Guidelines:

   - The answer [N/A] 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 discuss this in the appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work poses no such risk.

    Guidelines:

    - The answer [N/A] 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 include license information in the appendix.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: We do not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [No]

Justification: We do not use an LLM in this research.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.