Large Scale Dataset Distillation with Domain Shift

Noel Loo [1 2] Alaa Maalouf [1] Ramin Hasani [1 2] Mathias Lechner [1 2] Alexander Amini [1 2] Daniel Rus [1]

Abstract

Dataset Distillation seeks to summarize a large dataset by generating a reduced set of synthetic samples. While there has been much success at distilling small datasets such as CIFAR-10 on smaller neural architectures, Dataset Distillation methods fail to scale to larger high-resolution datasets and architectures. In this work, we introduce **D**ataset **D**istillation with **D**omain **S**hift (**D3S**), a scalable distillation algorithm, made by reframing the dataset distillation problem as a *domain shift* one. In doing so, we derive a universal bound on the distillation loss, and provide a method for efficiently approximately optimizing it. We achieve state-of-the-art results on Tiny-ImageNet, ImageNet-1k, and ImageNet-21K over a variety of recently proposed baselines, including high cross-architecture generalization. Additionally, our ablation studies provide lessons on the importance of validation-time hyperparameters on distillation performance, motivating the need for standardization.

1. Introduction

Dataset Distillation (Wang et al., 2018) is the task aiming to condense a large dataset into a smaller set of synthetic samples. The primary objective is to ensure that models trained with these synthetic samples can deliver competitive performance in comparison to models trained on the complete dataset (Loo et al., 2023b; Zhao & Bilen, 2021; Zhou et al., 2022; Maalouf et al., 2023b; Nguyen et al., 2021c; Zhao et al., 2021). Diverging from conventional core-sets or subset selection methods (Tukan et al., 2020; Borsos et al., 2020), dataset distillation methodology entails the generation of synthetic samples in a continuous space rather than their selection from the original dataset. This approach often leads to improved performance, particularly, in scenarios with higher compression rates. Recognizing the crucial importance of compressing large datasets into more manageable sizes, recent years have seen the introduction of several algorithms, such as gradient or distribution matching (Zhao et al., 2021; Zhao & Bilen, 2023), kernel-induced points (Nguyen et al., 2021b;c), feature alignment(Wang et al., 2022), and matching training trajectories (Cazenavette et al., 2022).

Scaling dataset distillation to large datasets. While dataset distillation techniques have shown impressive performance across diverse datasets and neural networks, they fall short when it comes to scaling to large datasets. This may stem from their significant GPU memory demands, inability to generate an informative synthetic dataset as the distilled set sizes increase, and difficulty in adapting to large, complex models; see (Yu et al., 2023) for more details.

To this end, in this work, we present **D**ataset **D**istillation with **D**omain **S**hift (**D3S**), a scalable distillation algorithm based on framing the dataset distillation problem as *Domain Shift* one. The problem of domain shift/adaptation deals with the test-time distribution differing from the training distribution, resembling the purpose of data distillation in some sense. While domain shift is well-studied independently of Dataset Distillation (Ben-David et al., 2010; Li et al., 2018), this work is the first to draw connections to Dataset Distillation. In doing so, we achieve state-of-the-art performance on large-scale datasets such as ImageNet-1K. Specifically, we contribute the following:

1. We formalize the problem of dataset distillation as one of Domain Shift, leading to a universal upper bound on the distillation loss.
2. We introduce an efficient method of approximating and minimizing this bound, leading to the D3S algorithm.
3. We verify D3S on large scale datasets such as ImageNet-1k, surpassing the previous state-of-the-art by up to 17.8%. [1]

2. Related Work

Coresets (Borsos et al., 2020; Chen, 2009; Maalouf et al., 2022a) are weighted subsets, selected from a larger train-

[*]Equal contribution [1]MIT CSAIL [2]Liquid AI. Correspondence to: Noel Loo <loo@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Code available at `https://github.com/yolky/d3s_distillation`

ing dataset. Used in training, they yield outcomes similar to the full dataset, expediting the training process significantly. Coresets have been developed for diverse machine-learning problems, such as k-means and k-median clustering (Maalouf et al., 2023a; Braverman et al., 2016; Huang & Vishnoi, 2020; Jubran et al., 2020; Cohen-Addad et al., 2022), regression (Maalouf et al., 2019; Meyer et al., 2022; Maalouf et al., 2022b), and low-rank approximation (Cohen et al., 2017; Braverman et al., 2020; Maalouf et al., 2020). Specifically designed for neural networks, recent strategies focus on choosing coresets before each training epoch to align their gradients with those of the entire dataset (Mirzasoleiman et al., 2020a;b; Tukan et al., 2023), then, the model undergoes training on the chosen coreset. However, despite theoretical support, these methods encounter limitations when attempting to compute a coreset (once) for an entire training procedure in practice.

Dataset distillation (Wang et al., 2018), akin to coresets, involves generating synthetic samples instead of selecting subsets from the training data. Here, synthetic samples are freely learned in continuous space rather than selected from the original dataset, and often excel in high compression rate scenarios, yielding superior performance. Similar to coresets, training on these synthetic samples aims to enhance speed and improve model performance (Zhao et al., 2021; Zhao & Bilen, 2021; Loo et al., 2022b; Nguyen et al., 2020; Loo et al., 2023b; Bohdal et al., 2020). Thus, dataset distillation holds promise in diverse applications such as continual learning (Sangermano et al., 2022; Zhou et al., 2022) and neural architecture search (Such et al., 2019). Dataset distillation methods vary, including approximate matching of training trajectories and gradient with the full dataset (Cazenavette et al., 2022; Zhao et al., 2021; Zhao & Bilen, 2023) and direct unrolling of the model training computation graph (Wang et al., 2018). Due to the high memory and computation demands of unrolling, recent approaches aim to approximate the unrolled computation (Nguyen et al., 2021b;c; Loo et al., 2022b; 2023b; Zhou et al., 2022). Recent work (Yin et al., 2023; Yin & Shen, 2023) propose separating the task of image recovery from the task of extracting information from the full dataset, leading to algorithms that scale to full-sized ImageNet-1K (Deng et al., 2009).

2.1. Domain Shift

The Domain Shift problem deals with the scenario where the training-time dataset differs in distribution from the test-time dataset (Ben-David et al., 2010; Mansour et al., 2009). This happens in many practical applications, either due to non-stationary data causing domain drift, biased datasets, or incorrect assumptions such as i.i.d. data. To tackle this problem, many methods have been employed, such as *domain adaptation* methods, which aim to quickly modify models to perform in novel unseen distributions (Zhao et al., 2019; Nguyen et al., 2021a), or *domain generalization* problems which aim to train a model on a source domain to transfer to a target domain (Li et al., 2018).

A general probabilistic framework for domain adaptation treats the source domain as a data distribution $p_S(x)$, with marginal label distribution $p_S(y|x)$, and the target domain analogously with $p_T(x)$ and $p_T(y|x)$ (Ben-David et al., 2010). Typically, $p_T(y|x)$ is not known (otherwise one may just train on the target domain directly), or there exist limited samples from it, but $p_T(x)$ is known. A common method to achieve domain invariance is to train a network θ to have latent representations $p_\theta(z|x)$ such that the marginal distribution of $p_S(z)$ and $p_T(z)$ are similar, which can be done via adversarial methods (Li et al., 2018), the Wasserstein distance (Shen et al., 2018), the KL-divergence (Nguyen et al., 2021a), or other methods (Zhao et al., 2019; Azizzadenesheli et al., 2019; Johansson et al., 2019). The problem is well studied theoretically (Ben-David et al., 2010; Mansour et al., 2009), leading to generalization bounds for the general domain shift problem as well as special cases such as covariate shift (Cortes et al., 2010; Johansson et al., 2019) or label shift (Azizzadenesheli et al., 2019). In this work, we use these bounds to motivate a general-purpose Dataset Distillation algorithm.

3. Reframing Dataset Distillation as Domain Shift

In this section, we reconsider the problem of dataset distillation, one which is typically viewed as a bilevel optimization problem (Wang et al., 2018), we frame it as a *domain shift* one. The key insight is to treat our distilled dataset distribution X_{Support} as our source domain distribution $p_S(x)$ and our full training set X_{Train} as the target distribution $p_T(x)$, and define $p_S(x, y)$ and $p_T(x, y)$ as the corresponding joint image-label distribution. In doing so, we will show that distributional similarity in both the image distribution and the conditional label distribution is necessary for a good distilled dataset. While much prior work (Zhao & Bilen, 2023; Yin et al., 2023) has used the notion of distributional similarity between the distilled and full dataset to motivation distillation algorithms, they lack theoretical justification, which we provide here.

A key difference in the formulation of domain shift for dataset distillation compared to standard domain shift is that typically, $p_S(x, y)$ and $p_T(x, y)$ are fixed, and the training algorithm which produces $p_\theta(y|x)$ is modified, i.e. we aim to develop a training procedure that creates networks that generalizes between domains. In our setting, **the training algorithm which produces $p_\theta(y|x)$ is fixed** as standard SGD, and **and we have control over** $p_S(x, y)$. This means typical distribution shift methods cannot be directly applied, and we must start from the fundamentals.

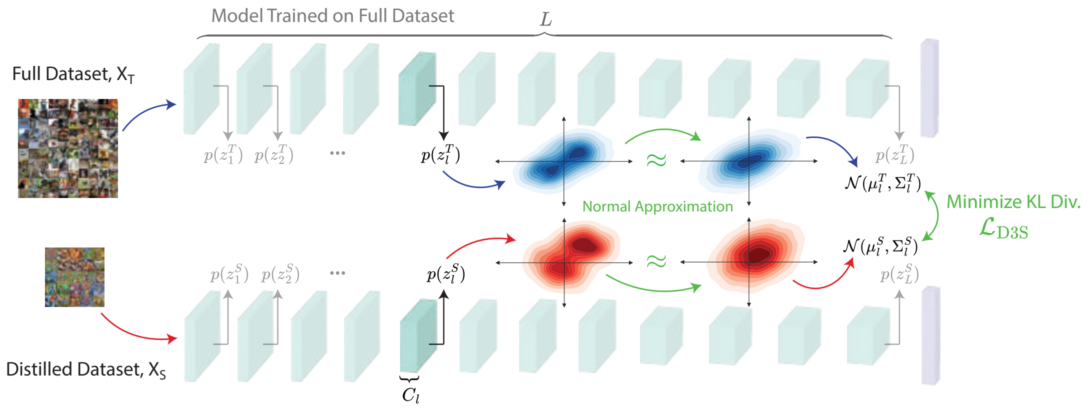
Model Trained on Full Dataset
L
Full Dataset, X_T
$p(z_1^T)$ $p(z_2^T)$ ... $p(z_l^T)$
$p(z_L^T)$
$\mathcal{N}(\mu_l^T, \Sigma_l^T)$
Normal Approximation
Minimize KL Div.
$\mathcal{L}_{D3S}$
$\mathcal{N}(\mu_l^S, \Sigma_l^S)$
$p(z_1^S)$ $p(z_2^S)$ ... $p(z_l^S)$
$p(z_L^S)$
Distilled Dataset, X_S
C_l

Figure 1. A schematic of the D3S algorithm. D3S works by approximating the domain shift bound in Theorem 3.1 using Normal approximations of intermediate representations of the full and distilled training set, on a network trained on the full dataset. Minimizing the KL-divergence between these Gaussian distributions leads to the D3S loss.

Let $\hat{p}(y|x)$ be the predictive distribution of any classifier, which typically in our case is the one given by a trained network on $p_S(x, y)$. With $l_S = \mathbb{E}_{p_S}[-\log \hat{p}(y|x)]$ and $l_T = \mathbb{E}_{p_T}[-\log \hat{p}(y|x)]$, i.e. the cross-entropy loss associated with the classifier $\hat{p}$ evaluated on the source and target distribution, respectively, we have the following bound on the difference between the two losses:

Theorem 3.1. *If $-\log \hat{p}(y|x)$ is bounded by positive constant C, we have:*

$$l_T \leq l_S + \frac{C}{2\sqrt{2}}\sqrt{D_{KL}\left(p_T(x,y)||p_S(x,y)\right)}$$
$$= l_S + \frac{C}{2\sqrt{2}}\sqrt{D_{KL}\left(p_T(x)||p_S(x)\right) + D_{KL}\left(p_T(y|x)||p_S(y|x)\right)}$$

Proof. See Appendix B

Where D_{KL} refers to the KL-divergence. The proof of this follows closely with that of proposition 2 Nguyen et al. (2021a), however, note that we modify direct the distributions $p_T(x)$, as opposed to that of a latent representation $p_T(z|x)$. Like Nguyen et al. (2021a), we can ensure that $-\log \hat{p}(y|x)$ is bounded by C by padding all labels with a small positive value to avoid non-zeros, but because we are interested in the classification accuracy in distillation as opposed to the loss directly, we omit such padding. Additionally, for the case when $D_{KL}(p_T||p_S) \geq 2$, we have a tighter bound in Theorem B.1.

4. D3S

We now present Dataset Distillation with Domain Shift (D3S), a method of efficiently optimizing the bound in Theorem 3.1. We can split this bound into two parts: optimizing $D_{KL}\left(p_T(x)||p_S(x)\right)$, which is the raw image distribution, which we discuss in Section 4.1, and optimizing $D_{KL}\left(p_T(y|x)||p_S(y|x)\right)$, which is the conditional distribution of the labels given a datapoint, which we discuss in Section 4.3.

4.1. Optimizing the Image Distribution

The estimation and optimization of KL divergences between two arbitrary continuous distributions is a longstanding challenge. Typical methods include the use of low-dimensional projections (Goldfeld & Greenewald, 2021), adversarial methods such as GANs (Goodfellow et al., 2014; Nowozin et al., 2016), or proxy metrics such as the MMD (Gretton et al., 2012; Chen et al., 2016; Arjovsky et al., 2017). These methods all require jointly iterating over samples from $p_T(x, y)$ and $p_S(x, y)$, which for large datasets can be very slow. Therefore, we aim for a method that can compute this KL divergence using only a set of summary statistics. Multivariate normal distributions are fully defined by their mean μ and covariance Σ, and therefore are a good choice.

Therefore, we approximate $p_T(x)$ and $p_S(x)$ as multivariate normal distributions over the distribution of latent representations of a network trained on $p_T(x)$. Specifically, let θ_T be a network of L layers trained on $p_T(x, y)$. Let $p_{S,\theta}(z_l) = \int h_\theta^l(x) p_S(x) dx$, where $h_\theta^l(x) = z_l$ is function which outputs the intermediate representations of θ at layer l. Specifically, at the output of each convolution layer, we approximate that distribution by C-dimensional multivariate normal, with $\mu_l^T \in R^{C_l}$, and $\Sigma_l^T \in R^{C_l \times C_l}$, where C_l is the number of convolutional channels of that layer[2]. This leads to a set of L μ_l^T and Σ_l^T statistics which summarize our full dataset, each corresponds to a layer of the network, with T indicating that it is on the training dataset, and l indexing the layer. These are precomputed in a single forward pass through the trained network. We do the same multivariate-normal approximation for each layer in our distilled set batch, leading to μ_l^S and Σ_l^S. We then optimize the KL-divergence formula for multivariate normals, plus a

[2]We use $^{\mathsf{T}}$ for tranpose and ^{T} to denote the training/target set.

Algorithm 1 Dataset Distillation with Distribution Shift Image Synthesis (D3S)

```
Input: A set of $M$ trained teacher models: $\{f_{\theta_m}(x)\}_{m=1}^{M}$
       Precomputed full-dataset statistics for L layers of M models: $\{\mu_{m,l}^{T}\}_{m,l=1}^{M,L}$, $\{\Sigma_{m,l}^{T}\}_{m,l=1}^{M,L}$
       Randomly initialized initial distilled dataset $X_S$ of size $|S|$ and target one-hot labels $y_S$
       Scalar label loss coefficient $\alpha$, learning rate $\eta$, iterations per batch $K$, batch size $|B|$
Output: Distilled dataset images $X_S$
Initialize: running $\{\tilde{\mu}_{m,l}^{S}\}_{m,l=1}^{M,L}$ and $\{\tilde{\Sigma}_{m,l}^{S}\}_{m,l=1}^{M,L}$ at 0 for ISU
for batch index $b$ in $\{1, \cdots, \lceil \frac{|S|}{|B|} \rceil\}$ do
  Select new batch $X_b \subset X_S$ and labels $y_b \subset y_S$ of size $|B|$ and model index $m \leftarrow (b \mod M)$
  for Iteration $t$ in $\{1, \cdots, K\}$ do {Main optimization loop}
    Pass $X_b$ through $f_{\theta_m}$, to obtain $\hat{y}_b = f_{\theta_m}(X_b)$ and batch feature means and covariances $\{\mu_{m,l}^{S,b}\}_{l=1}^{L}$, $\{\Sigma_{m,l}^{S,b}\}_{l=1}^{L}$
    Update: $\{\hat{\mu}_{m,l}^{S,b}\}_{l=1}^{L}$ and $\{\hat{\Sigma}_{m,l}^{S,b}\}_{l=1}^{L}$ with Equation (2)
    Compute: $\mathcal{L}_{D3S}$ via equation Equation (1) using $\{\hat{\mu}_{m,l}^{S,b}\}_{l=1}^{L}$ and $\{\hat{\Sigma}_{m,l}^{S,b}\}_{l=1}^{L}$    ▷ See Algorithm 3 for more details
    Update: $X_b \leftarrow X_b - \eta \frac{\partial \mathcal{L}_{D3S}}{\partial X_b}$
  end for
  Update: $\tilde{\mu}_{m,l}^{S}$ and $\tilde{\Sigma}_{m,l}^{S}$ for every model $m$ and layer $l$ with Equation (2)
end for
Return: $X_S$    ▷ See Algorithm 2 for the labelling procedure
```

small label alignment term from a cross-entropy loss on the target labels $\mathcal{L}_{\text{x-ent}}$, leading to the D3S loss:

$$\mathcal{L}_{\text{D3S}} = \frac{1}{2}\sum_{l=1}^{L}\Bigg(\log\frac{|\Sigma_l^T|}{|\Sigma_l^S|} + \text{Tr}((\Sigma_l^T)^{-1}\Sigma_l^S) + (\mu_l^T - \mu_l^S)^{\intercal}(\Sigma_l^T)^{-1}(\mu_l^T - \mu_l^S) - C_l\Bigg) + \alpha\mathcal{L}_{\text{x-ent}}(y, f_\theta(x_S)) \tag{1}$$

Where y is the target labels for the distilled dataset x_S, and $f_\theta(x_S)$ is the output of the trained network on that set. α is a scalar value which we fix at 20.0. Note that we can cache the expensive computation of Σ_l^{T-1} and $\log|\Sigma_l^T|$, since these are constant. This use of cached statistics has also been in used prior work in domain adaptation (Adachi et al., 2022).

Note that by doing this multivariate-normal approximation, we actually are no longer directly optimizing the bound in Theorem 3.1. Nonetheless, we hope that this is an accurate proxy for it. The use of proxies for the hard-to-compute KL-divergence has been used in prior literature (Chen et al., 2016). Future work can study better methods for calculating and optimizing the bound in Theorem 3.1, or for more expressive approximations than the one we use in D3S.

4.2. Batch Incremental Statistic Updating (ISU)

Optimizing Equation (1) requires optimization of statistics μ_l^S and Σ_l^S, which are functions of the **whole** distilled dataset. For larger distilled sets, loading this into memory is unfeasible, so we must batch the computation into batch sizes $|B|$. However, optimizing $\mu_l^{S,b}$ and $\Sigma_l^{S,b}$ independently for each batch does not take into account the contributions from other batches. We propose a simple approximation that mitigates this problem, which we call *incremental statistic updating* (ISU). Specifically, we initialize running counters of $\tilde{\mu}_l^S = 0$ and $\tilde{\Sigma}_l^S = 0$. For the bth batch, we optimize Equation (1) for T iterations, using $\hat{\mu}_l^{S,b}$ and $\hat{\Sigma}_l^{S,b}$ given by:

$$\begin{aligned}\hat{\mu}_l^{S,b} &= \frac{1}{b}\mu_l^{S,b} + (1-\frac{1}{b})\tilde{\mu}_l^S \\ \hat{\Sigma}_l^{S,b} &= \frac{1}{b}\Sigma_l^{S,b} + (1-\frac{1}{b})\tilde{\Sigma}_l^S\end{aligned} \tag{2}$$

We then update $\tilde{\mu}_l^S \leftarrow \hat{\mu}_l^{S,b}$ and $\tilde{\Sigma}_l^S \leftarrow \hat{\Sigma}_l^{S,b}$ after optimizing batch b. That is, $\tilde{\mu}_l^S$ and $\tilde{\Sigma}_l^S$ are the running statistics, and batch b only aims to fill the "missing" part that previous batches did not contain.

4.3. Optimizing the Conditional Label Distribution

Optimizing $D_{KL}(p_T(y|x)||p_S(y|x))$ is much more straightforward. Assuming that the trained network used in Section 4.1 to generate the trained dataset statistics approximately learns $p_T(y|x)$, we can use it to label the images synthesized from Section 4.1, akin to *knowledge distillation* (Bucila et al., 2006; Hinton et al., 2015). This choice is supported by literature suggesting that networks learn the true Bayesian conditional layer distribution $p_T(y|x)$ during training (Menon et al., 2020), meaning that by using knowledge-distillation we match $p_S(y|x)$ with $p_T(y|x)$. Following prior work, we additionally generate labels for the distilled dataset under different augmentations. Reframing the use of knowledge distillation in dataset distillation as minimizing $D_{KL}(p_T(y|x)||p_S(y|x))$ sheds

Table 1. Large scale distillation results on Tiny-ImageNet, ImageNet-1K and ImageNet-21K, with the source model a ResNet-18, and validating of ResNet-18s, 50s, and 101s. D3S outperforms prior work on all benchmarks.

Dataset	IPC	ResNet-18			ResNet-50			ResNet-101			Full Dataset (ResNet-18)
		SRe^{2}L	CDA	D3S (Ours)	SRe^{2}L	CDA	D3S (Ours)	SRe^{2}L	CDA	D3S (Ours)	
Tiny-ImageNet	50	41.4 ± 0.4	48.7	$\mathbf{56.4 \pm 0.3}$	42.2 ± 0.5	49.7	$\mathbf{56.8 \pm 0.3}$	42.5 ± 0.2	50.6	$\mathbf{56.8 \pm 0.7}$	59.8 ± 0.1
	100	49.7 ± 0.3	53.2	$\mathbf{58.8 \pm 0.3}$	51.2 ± 0.4	54.4	$\mathbf{59.8 \pm 0.2}$	51.5 ± 0.3	55.0	$\mathbf{60.3 \pm 0.4}$	
ImageNet-1K	10	21.3 ± 0.6	–	$\mathbf{39.1 \pm 0.3}$	28.4 ± 0.1	–	$\mathbf{41.9 \pm 0.7}$	30.9 ± 0.1	–	$\mathbf{42.1 \pm 3.8}$	69.8 ± 0.1
	50	46.8 ± 0.2	53.5	$\mathbf{60.2 \pm 0.1}$	55.6 ± 0.3	61.3	$\mathbf{65.8 \pm 0.1}$	60.8 ± 0.5	61.6	$\mathbf{65.3 \pm 0.5}$	
	100	52.8 ± 0.3	58.0	$\mathbf{63.0 \pm 0.2}$	61.0 ± 0.4	65.1	$\mathbf{68.2 \pm 0.1}$	62.8 ± 0.2	65.9	$\mathbf{68.9 \pm 0.1}$	
	200	57.0 ± 0.4	63.3	$\mathbf{64.6 \pm 0.1}$	64.6 ± 0.3	67.6	$\mathbf{69.5 \pm 0.0}$	65.9 ± 0.3	68.4	$\mathbf{70.1 \pm 0.0}$	
ImageNet-21K	10	18.5	22.6	$\mathbf{26.9 \pm 0.1}$	27.4	32.4	$\mathbf{34.4 \pm 0.0}$	27.3	34.2	$\mathbf{35.1 \pm 0.2}$	38.0 ± 0.1
	20	20.5	26.4	$\mathbf{28.5 \pm 0.1}$	29.5	**35.3**	$\mathbf{35.4 \pm 0.0}$	31.8	**36.1**	$\mathbf{36.0 \pm 0.1}$	

light as to its efficacy in previous works (Yin et al., 2023; Cui et al., 2023).

4.4. Effectively leveraging multiple networks

Allen-Zhu & Li (2023) explains the efficacy of knowledge/ensemble distillation under the "multi-view" theory: large datasets contain multiple predictive features, and independently trained networks are biased to only use a subset of them each. With this in mind, we choose to use multiple networks to label our distilled set in Section 4.3, akin to ensemble distillation, we call this *Ensemble Labelling* (ENL). However, this use of ensembling only superficially addresses the multi-view problem, as if our images are synthesized from a single network, they are prone to be biased towards the predictive features used by that single network. Therefore, for each batch of synthesized we choose one of M trained models θ_m to optimize the next batch on, we call this *Ensemble Synthesis* (ENS). In all experiments, we use both ENL and per-batch ENS, with careful ablations of these two design choices in Section 6.

4.5. Putting it all together

Combining the methods in the previous sections leads to our method D3S. Simply put, we require M trained networks trained on $p_T(x, s)$. Then, we measure the statistics $\mu^T_{m,l}$ and $\Sigma^T_{m,l}$ in a single forward pass on the full dataset on these networks. We then synthesize the images using Equation (1), using ISU and ENS. This step corresponds to optimizing $D_{KL}\left(p_T(x)||p_S(x)\right)$. Finally, we relabel our synthesized images $p_S(x)$ using the same M models θ_m, using ENL, which effectively minimizes $D_{KL}\left(p_T(y|x)||p_S(y|x)\right)$. The pseudocode for the image synthesis step is provided in Algorithm 1. Pseudocode for the image labeling step, and more detailed pseudocode for image synthesis are available in Appendix A.

5. Results

5.1. Large Scale Image Distillation

We validate the efficacy of D3S on large-scale distillation tasks. We consider three tasks, ranging from smaller scale to larger scale. Firstly, we have Tiny-ImageNet (Le & Yang, 2015), consisting of 200 classes at resolution 64×64 with a total of 100K images. As our medium scale dataset, we consider ImageNet-1K (Deng et al., 2009), with 1000 classes and roughly 1M images with resolution 224×224. Finally, we have our most challenging task, ImageNet-21K (Ridnik et al., 2021), which has 10450 classes and $\sim$11M images with resolution 224×224. For baselines, we consider two recently proposed large-scale dataset distillation algorithms: SRe^{2}L (Yin et al., 2023), and CDA (Yin & Shen, 2023), and consider the task of distilling from ResNet-18s (running Algorithm 1 using $M = 5$ trained ResNet-18s), and evaluate their performance on ResNet-18s, ResNet-50s, and ResNet-101s. We report results in Table 1, using reported results in prior work[3]. During validation, we use the same training duration as prior work (Yin et al., 2023), to avoid confounding factors. Indeed, in Section 7, we show that parameters such as training time have a significant impact on the performance of distillation algorithms, so we control that variable here.

Note that our algorithm is designed with **large scale distillation** in mind, which is why choices such as using precomputed statistics μ_l, Σ_l of the full dataset were made. This obviates the need to iterate over the training dataset during distillation, which is infeasible for datasets such as ImageNet-1K and 21K. Therefore, we focus our efforts on benchmarking our algorithm on larger ResNets on larger scale datasets and avoid the typical 3-5 layer ConvNet seen in prior art, and smaller, low-resolution datasets such as CIFAR-10, and CIFAR-100 (Krizhevsky, 2009). With smaller datasets and models, more complex

[3]CDA does not report standard deviations in any of their results, nor specifies the number of runs used, so we report only the (assumed) mean

Table 2. Cross-architecture generalization of D3S on ImageNet-1K, IPC 50. We use a source model of ResNet-18s, and validate on various other architectures. The performance gain on the smaller source model holds for all other architectures.

Algorithm	Validation Model						
(with RN-18)	RN-18	RN-50	RN-101	DenseNet-121	RegNet-Y-8GF	ConvNeXt-Tiny	DeiT-Tiny
SRe^{2}L	46.80	55.60	60.81	49.74	60.34	53.53	15.41
CDA	53.45	61.26	61.57	57.35	63.22	62.58	31.95
D3S (Ours)	$\mathbf{60.21 \pm 0.07}$	$\mathbf{65.75 \pm 0.07}$	$\mathbf{65.28 \pm 0.51}$	$\mathbf{63.58 \pm 0.03}$	$\mathbf{67.19 \pm 0.09}$	$\mathbf{67.33 \pm 0.06}$	$\mathbf{36.86 \pm 2.30}$

Table 3. Cross-architecture generalization of D3S on ImageNet-21K, IPC 20. We use a source model of ResNet-18s, and validate on various other architectures.

Algorithm	Validation Model				
(with RN-18)	RN-18	RN-50	RN-101	DenseNet-121	RegNet-Y-8GF
CDA	26.42	$\mathbf{35.32}$	$\mathbf{36.12}$	28.66	36.13
D3S (Ours)	$\mathbf{28.53 \pm 0.09}$	$\mathbf{35.45 \pm 0.02}$	$\mathbf{36.03 \pm 0.08}$	$\mathbf{31.97 \pm 0.02}$	$\mathbf{36.42 \pm 0.05}$

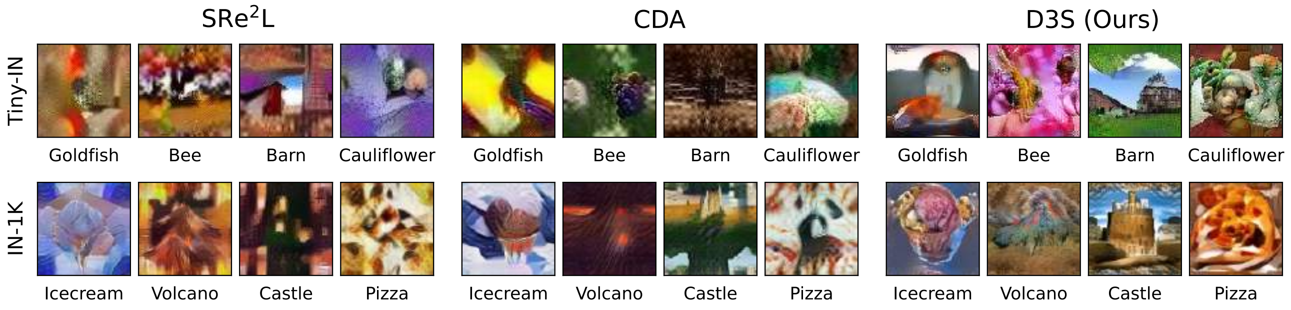

Figure 2. Visualization of distilled images from SRe^{2}L, CDA, and D3S on Tiny-ImageNet (top row) and ImageNet-1K (bottom row). D3S produces significantly more realistic images than the other two algorithms. More visualizations are available in Appendix E

techniques such as trajectory matching (Cazenavette et al., 2022), bilevel-optimization, (Wang et al., 2018; Loo et al., 2023a) and KRR methods (Nguyen et al., 2021b;c; Loo et al., 2022a; Zhou et al., 2022) can be done, but these **do not scale to larger models or datasets**, so we do not compare to those methods here.

Table 1 shows the results. **D3S outperforms prior work on every benchmark**, often by substantial margins. For example, in Tiny-ImageNet-1K, at 50 images per class (IPC), our algorithm achieves 56.4%, which is more than 8% better than the prior best, and outperforms CDA with double the number of images. Note that with the source models achieving performances of 59.8% (see Table 8), we are nearly saturating the performance of the full-dataset with 100 IPC. A similar story is seen in both ImageNet-1K and ImageNet-21K, with our algorithms typically outperforming competing methods with double the images on ResNet-18s. The performance gains hold when validating on larger models, with our method still seeing similar performance gains on ResNet-50s and ResNet-101s. While our method still outperforms prior work on ImageNet-21K, the margin is less. This suggests that the block-diagonal Gaussian approximation of the datasets latent features we describe in Section 4.1 may be a poor approximation for more complex datasets, and a richer one may be necessary.

5.2. Architecture Generalization

A key desiderata of dataset distillation is *cross-arhictecture generalization*, which is the ability of distilled datasets to train models of varying architectures to high accuracy. High generalization suggests that an algorithm is not over-fitting to any specific model and that the features selected by the dataset distillation algorithm are generalizable. We verify that D3S generates images which transfer to other architectures such as DenseNet-121 (Huang et al., 2016), RegNet (Xu et al., 2021), and ConvNeXt (Liu et al., 2022) in Table 2 and Table 3 for ImageNet-1K 50 per class, and ImageNet-21K 20 per class, respectively, with our base distillation model being the ResNet-18. The margin of performance over prior work seen on ResNet-18s still holds for larger models. We conjecture that this ability to transfer is because the bound in Theorem 3.1 is **model-agnostic**, that is, it holds for any architecture or classifier. Indeed, this suggests that using information-theoretic methods for dataset distillation is a promising avenue to tackle the architecture generalization problem and could be the subject of future

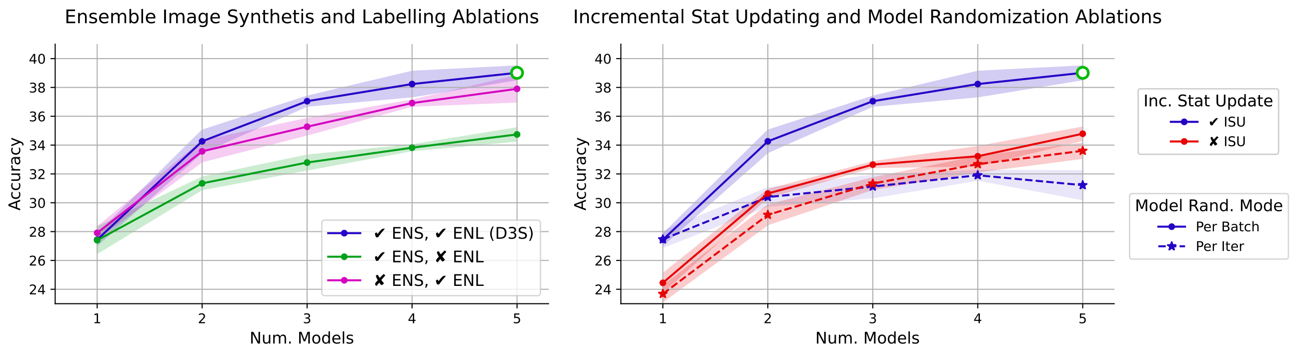

Figure 3. Ablations of the design choices in D3S. Left shows the effect of using Ensemble Synthesis (ENS) and Ensemble Labelling (ENL) at various model counts. Right shows the impact of using Incremental Statistic Updating (ISU) and the choice of model randomization mode. ENS, ENL, and ISU with per-batch randomization provide the best performance.

work.

5.3. Image Visualization

We visually compare images generated by SRe^{2}L (Yin et al., 2023), CDA (Yin & Shen, 2023) and D3S in Figure 2 on Tiny-ImageNet and ImageNet-1K. It is apparent that D3S produces visually more realistic images than compared to the other two algorithms. We hypothesize that this is due to the richer set of statistics given by μ_l and Σ_l, as Σ_l contains covariance information, in addition to the KL-divergence loss in Equation (1). In comparison, SRe^{2}L and CDA both use a Batchnorm statistic-based loss. Not only is this loss not well-founded in theory, it also only contains statistics for convolutional channels *independently*, and ignores the off-diagonal covariance which we have in Σ_l.

6. Ablations

D3S was designed with the goal of minimizing Theorem 3.1 over the *whole* distilled set, but the whole distilled set typically cannot fit in memory. As a result, we introduced Incremental Statistic Updating (ISU) in Section 4.2 and Equation (2), which aims to approximate the effect of full-batch optimization with mini-batches. Here, we validate the importance of this choice, along with the proposed use of multiple trained models in Section 4.4. As stated in Section 4.4, we can use these models in two ways: using the ensemble for image synthesis (ENS), or only for labelling (ENL). D3S uses both. In Figure 3a, we compare D3S with both ENS and ENL to versions with either component removed, varying the number of models used on the task of distilling ImageNet-1K with 10 IPC, evaluated on ResNet-18s. For the configuration with ENS but no ENL, we use a single model to label the images synthesized by the M networks. In Figure 3a, we use ISU in all experiments. The configuration used in Section 5 is given by the green circle, which uses five trained models. We see that using multiple networks in all configurations leads to improved performance, but using the models for both synthesis and labelling outperforms using them for only labelling, which is better than using them only for synthesis. Using multiple modeling for both comes at no extra runtime cost aside from the cost of training these networks, and should therefore be the preferred option. Performance largely saturates after 4 models, however it is likely that slightly more performance can be gained with even greater models.

Figure 3b shows the importance of using ISU. In this figure, we compare D3S using both ENL and ENS, and varying whether we use ISU or not. Additionally, as described in Section 4.4, we randomize the model used for image synthesis after optimizing every batch, which we call “per-batch” randomization in Figure 3a, but one could alternatively consider selecting a model at random after every optimization step instead, which we call “per-iter” randomization. From Figure 3b, we see that while ISU provides a significant improvement over the baselines when using a single model for distillation, the benefits of ISU are only seen in the “per-batch” randomization mode, with all other configurations performing much worse. We hypothesize that the poor performance of “per-iter” randomization is due to the difficulty of optimizing the objective over multiple networks, as a single batch must optimize simultaneously M objectives (M being the model count), given the same number of optimization iterations as the per-batch method, which only has to optimize one. Furthermore, the randomness of model selection adds variance to the optimization. It is infeasible to optimize Equation (1) without randomizing over models as it would increase the runtime per iteration by a factor of M and increase the memory requirements M-fold (without gradient accumulation). Therefore, the proposed configuration of D3S which uses ISU + per-batch randomization is superior.

6.1. Important of the Multivariate KL

While prior work uses a BatchNorm-based loss based on diagonal covariance Gaussians (Yin et al., 2023), ours uses a Multivariate KL loss. We verify the choice of both the

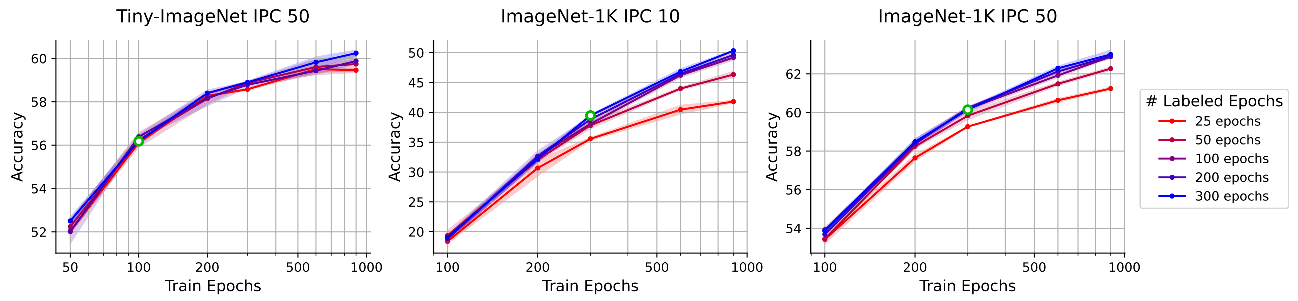

Figure 4. Effect of number of labeled augmentation epochs and training duration during validation for D3S on Tiny-ImageNet 50 IPC, ImageNet-1K 10 IPC, and 50 IPC. Performance is highly sensitive to training duration, with longer training producing better results, while less sensitive to the number of epochs that are labeled.

	ImageNet-1K 10/Cls
CDA (BN Loss, Diagonal)	23.68±0.70%
BN Loss, Multivariate	24.00±0.38%
KL Loss, Diagonal	23.47±0.21%
D3S (KL Loss, Multivariate)	**27.96±0.37%**

Table 4. The Effect of using the KL loss vs. the BatchNorm loss, and Multivariate vs. Diagonal Covariance. Using both the full covariance and the KL divergence loss is necessary to achieve high performance.

KL divergence and multivariate covariance is necessary in Table 4. Here, we distill ImageNet-1K to 10/cls, without no ENS or ISU to isolate the effect of the loss function. Additionally, we vary either using diagonal covariance estimates, or the full covariance, and consider whether to use the BatchNorm loss used in (Yin et al., 2023), or the KL-divergence proposed here. We see that both using the multivariate and KL divergence is necessary to gain performance, as using only one results in similar performance to CDA ($\sim$24%), whereas using multivariate KL gives 28%.

7. Undertraining and Overlabelling in Dataset Distillation

Minimization of the conditional label distribution in Theorem 3.1 requires labelling the image under different augmentations. Following (Yin et al., 2023), this is done without storing additional images by storing the data augmentation parameters, such as the cropping coordinates, alongside the label. But this is not without cost, as the labels and cropping coordinates themselves additionally cost storage. As these augmentations must be made for each training epoch, the cost of storing these labels grows with $E \times |S| \times C$, where E is the number of epochs of downstream training, $|S|$ is the size of the distilled dataset, and C is the number of classes. This can quickly outpace the size of the distilled images themselves. For example, for ImageNet-21K, with 10450 classes, in 15 epochs, we require approximately as much storage for the labels as the images ($15 \times 10450 \approx 3 \times 224 \times 224$). In order to make the comparison in Section 5, we followed prior work, generating new labels for each epoch of training (300). But now, we take a more critical approach and ask how many of these labels we actually need.

To answer this question, we relabel our distilled datasets for Tiny-ImageNet 50 IPC and ImageNet-1K 10 and 50 IPC for fewer epochs: as few as 25, up to the standard 300, and train on these distilled sets for up to 900 epochs, with results shown in Figure 4. When the number of training epochs exceeds the number of epochs which we have labels for, we simply repeat the labels, but randomizing their order. The standard configuration used in Section 5, with 300 epochs for both labelling and training, respectively, is given by the green circles in Figure 4. From Figure 4, we draw two main conclusions:

We do not need many labels for high generalization. For Tiny-ImageNet, we less than 1.5% performance drop when labelling for only 25 epochs compared to 300 when training for going from 60.2% to 58.9%, and for ImageNet-1K, we only see performance drops when labelling for fewer than 100 epochs and training for long, which requires 3x less the storage space to store the labels compared to labelling 300 epochs. This suggests that even with relatively few labelled augmentations, models trained on these distilled datasets do not overfit unless trained for long.

Training for long almost always improves performance. By training for up to 900 epochs, we can achieve performance on ImageNet-1K with 10 IPC of 50.1%, over 10% higher than the performance achieved at 300 epochs, at 39.1%. Similarly, training on the 50 IPC distilled datasets for 900 epochs sees 2.6% gain over training for 300, reaching 62.7%, and likely training further will improve results. This finding that longer training improves performance is consistent with prior studying in knowledge-distillation (Beyer et al., 2021), which finds mimicking teacher labels for very longer under various augmentations leads to improved performance. Unsurprisingly, as we are using a knowledge-distillation-like method for generating distilled dataset labels, we see a similar finding in dataset distillation. A more concerning conclusion one could draw is the sensitivity of dataset distillation results in response to these parameters. These findings suggest that researchers should be rigorous in adhering to standardized downstreaming training protocols in dataset distillation literature.

8. Discussion, Conclusions, and Limitations

In this work, we introduce D3S, a scalable distillation algorithm based on the theory of distribution shift, and show its efficacy in distilling large-scale datasets such as ImageNet-1K. D3S is one method to optimize the bound in Theorem 3.1, but requires approximations such as a multivariate normal one described in Section 4.1. Future work could look at more sophisticated methods of optimizing Theorem 3.1 or extending D3S to other domains such as language.

Additionally, in Section 4.2 and Section 4.4, we introduce methods for computing statistics over the whole distilled data and leveraging multiple networks, respectively, which could prove useful in future distillation algorithms. Finally, in Section 7 we report interesting findings of the sensitivity of dataset distillation to training time, as well as a elucidated the cost of storing multiple labels for distilled sets. Future work could study more efficient label parameterizations.

Overall, our work provides a novel, effective, and theoretically robust approach to dataset distillation, which could serve as the basis for future distillation techniques.

Impact Statement

This work details with Dataset Distillation, which can make training more efficient by reducing the energy requirements of training models. Additionally, due to the widespread use of deep learning, this also has other harmful societal impacts, but none of these are particular to the topics studied in this paper.

References

Adachi, K., Yamaguchi, S., and Kumagai, A. Covariance-aware feature alignment with pre-computed source statistics for test-time adaptation to multiple image corruptions. *2023 IEEE International Conference on Image Processing (ICIP)*, pp. 800–804, 2022. URL `https://api.semanticscholar.org/CorpusID:248426981`.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=Uuf2q9TfXGA`.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

Azizzadenesheli, K., Liu, A., Yang, F., and Anandkumar, A. Regularized learning for domain adaptation under label shifts. *CoRR*, abs/1903.09734, 2019. URL `http://arxiv.org/abs/1903.09734`.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010. URL `http://www.springerlink.com/content/q6qk230685577n52/`.

Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. Knowledge distillation: A good teacher is patient and consistent. *CoRR*, abs/2106.05237, 2021. URL `https://arxiv.org/abs/2106.05237`.

Bohdal, O., Yang, Y., and Hospedales, T. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020.

Borsos, Z., Mutny, M., and Krause, A. Coresets via bilevel optimization for continual learning and streaming. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 14879–14890, 2020.

Braverman, V., Feldman, D., Lang, H., Statman, A., and Zhou, S. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

Braverman, V., Drineas, P., Musco, C., Musco, C., Upadhyay, J., Woodruff, D. P., and Zhou, S. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pp. 517–528, 2020.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. Model compression. volume 2006, pp. 535–541, 08 2006. doi: 10.1145/1150402.1150464.

Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J.-Y. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016. URL `http://arxiv.org/abs/1606.03657`.

Cohen, M. B., Musco, C., and Musco, C. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 1758–1777, 2017.

Cohen-Addad, V., Larsen, K. G., Saulpic, D., and Schwiegelshohn, C. Towards optimal lower bounds for k-median and k-means coresets. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1038–1051, 2022.

Cortes, C., Mansour, Y., and Mohri, M. Learning bounds for importance weighting. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/59c33016884a62116be975a9bb8257e3-Paper.pdf.

Cui, J., Wang, R., Si, S., and Hsieh, C.-J. Scaling up dataset distillation to imagenet-1k with constant memory, 2023.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Goldfeld, Z. and Greenewald, K. Sliced mutual information: A scalable measure of statistical dependence. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=SvrYl-FDq2.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Huang, L. and Vishnoi, N. K. Coresets for clustering in euclidean spaces: importance sampling is nearly optimal. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 1416–1429, 2020.

Johansson, F. D., Sontag, D., and Ranganath, R. Support and invertibility in domain-invariant representations, 2019.

Jubran, I., Tukan, M., Maalouf, A., and Feldman, D. Sets clustering. In *International Conference on Machine Learning*, pp. 4994–5005. PMLR, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Li, H., Pan, S. J., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5400–5409, 2018. doi: 10.1109/CVPR.2018.00566.

Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. URL https://arxiv.org/abs/2201.03545.

Loo, N., Hasani, R., Amini, A., and Rus, D. Efficient dataset distillation using random feature approximation. *Advances in Neural Information Processing Systems*, 2022a.

Loo, N., Hasani, R., Amini, A., and Rus, D. Efficient dataset distillation using random feature approximation. *arXiv preprint arXiv:2210.12067*, 2022b.

Loo, N., Hasani, R., Lechner, M., and Rus, D. Dataset distillation with convexified implicit gradients, 2023a. URL https://arxiv.org/abs/2302.06755.

Loo, N., Hasani, R., Lechner, M., and Rus, D. Dataset distillation fixes dataset reconstruction attacks. *arXiv preprint arXiv:2302.01428*, 2023b.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers. *Advances in Neural Information Processing Systems*, 32, 2019.

Maalouf, A., Statman, A., and Feldman, D. Tight sensitivity bounds for smaller coresets. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2051–2061, 2020.

Maalouf, A., Eini, G., Mussay, B., Feldman, D., and Osadchy, M. A unified approach to coreset learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022a.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9977–9994, 2022b.

Maalouf, A., Tukan, M., Braverman, V., and Rus, D. AutoCoreset: An automatic practical coreset construction framework. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23451–23466. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/maalouf23a.html.

Maalouf, A., Tukan, M., Loo, N., Hasani, R., Lechner, M., and Rus, D. On the size and approximation error of distilled datasets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation: Learning bounds and algorithms. *CoRR*, abs/0902.3430, 2009. URL http://arxiv.org/abs/0902.3430.

Menon, A. K., Rawat, A. S., Reddi, S. J., Kim, S., and Kumar, S. Why distillation helps: a statistical perspective. *CoRR*, abs/2005.10419, 2020. URL https://arxiv.org/abs/2005.10419.

Meyer, R. A., Musco, C., Musco, C., Woodruff, D. P., and Zhou, S. Fast regression for structured inputs. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

Mirzasoleiman, B., Bilmes, J. A., and Leskovec, J. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pp. 6950–6960, 2020a.

Mirzasoleiman, B., Cao, K., and Leskovec, J. Coresets for robust training of deep neural networks against noisy labels. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020b.

Nguyen, A. T., Tran, T., Gal, Y., Torr, P. H. S., and Baydin, A. G. KL guided domain adaptation. *CoRR*, abs/2106.07780, 2021a. URL https://arxiv.org/abs/2106.07780.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=l-PrrQrK0QR.

Nguyen, T., Novak, R., Xiao, L., and Lee, J. Dataset distillation with infinitely wide convolutional networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021c. URL https://openreview.net/forum?id=hXWPpJedrVP.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization, 2016.

Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. Imagenet-21k pretraining for the masses, 2021.

Sangermano, M., Carta, A., Cossu, A., and Bacciu, D. Sample condensation in online continual learning, 2022. URL https://arxiv.org/abs/2206.11849.

Shen, J., Qu, Y., Zhang, W., and Yu, Y. Wasserstein distance guided representation learning for domain adaptation, 2018.

Shen, Z. and Xing, E. A fast knowledge distillation framework for visual recognition. In *ECCV*, 2022.

Such, F. P., Rawal, A., Lehman, J., Stanley, K. O., and Clune, J. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. *CoRR*, abs/1912.07768, 2019. URL http://arxiv.org/abs/1912.07768.

Tukan, M., Maalouf, A., and Feldman, D. Coresets for near-convex functions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tukan, M., Zhou, S., Maalouf, A., Rus, D., Braverman, V., and Feldman, D. Provable data subset selection for efficient neural network training. *arXiv preprint arXiv:2303.05151*, 2023.

Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., and You, Y. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12196–12205, 2022.

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

Xu, J., Pan, Y., Pan, X., Hoi, S., Yi, Z., and Xu, Z. Regnet: Self-regulated network for image classification, 2021.

Yin, Z. and Shen, Z. Dataset distillation in large data era, 2023.

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

Yu, R., Liu, S., and Wang, X. Dataset distillation: A comprehensive review. *arXiv preprint arXiv:2301.07014*, 2023.

Zhao, B. and Bilen, H. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, 2021.

Zhao, B. and Bilen, H. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6514–6523, 2023.

Zhao, B., Mopuri, K. R., and Bilen, H. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=mSAKhLYLSsl`.

Zhao, H., des Combes, R. T., Zhang, K., and Gordon, G. J. On learning invariant representation for domain adaptation. *CoRR*, abs/1901.09453, 2019. URL `http://arxiv.org/abs/1901.09453`.

Zhou, Y., Nezhadarya, E., and Ba, J. Dataset distillation using neural feature regression. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Algorithm 2 Dataset Distillation with Distribution Shift ensemble labelling procedure

```
Input: $M$ trained teacher models: $f_{\theta_m}(x)_{m=1}^{M}$
       Distilled dataset made with Algorithm 3, $X_S$
       Validation batch size $|B|$ and number of labelling epochs $E$
Output: Labels for $E$ epochs: $Y$, augmentation auxiliary information $Z_{\text{aug}}$, and batch indices $I$
Initialize labels $Y \leftarrow []$ as empty list
Initialize auxiliary augmentation info $Z_{\text{aug}} \leftarrow []$ as empty list
Initialize batch indices $I \leftarrow []$ as empty list
for epoch $e$ in $\{1, \cdots, E\}$ do
  Initialize epoch labels $Y_e \leftarrow []$ as empty list
  Initialize auxiliary augmentation info $Z_{e,\text{aug}} \leftarrow []$ as empty list
  Initialize epoch batch indices $I \leftarrow []$ as empty list
  for batch index $b$ in $\{1, \cdots, \lceil \frac{|S|}{|B|} \rceil\}$ do
    Sample new batch $X_b \subset X_S$ of indices $I_b$
    Apply augmentation to $\tilde{X}_b \leftarrow \texttt{aug}(X_b)$, and record augmentation parameters $z_{b,\text{aug}}$
    Set batch labels $X_b = 0 \in \mathbb{R}^{\mathbb{C}}$, with $C$ the number of classes
    for model index $m$ in $\{1, \cdots, M\}$ do {Main optimization loop}
      $Y_b \leftarrow Y_b + \frac{1}{M} f_{\theta_m}(\tilde{X}_b)$          ▷ Add in logit space
    end for
    Append $Y_b$ to $Y_e$
    Append $Z_{b,\text{aug}}$ to $Z_{e,\text{aug}}$
    Append $I_b$ to $I_e$
  end for
  Append $Y_e$ to $Y$
  Append $Z_{e,\text{aug}}$ to $Z_{\text{aug}}$
  Append $I_e$ to $Y$
end for
Return: $Y$, $Z_{\text{aug}}$, $I$
```

A. Algorithm Details

A.1. Statistic Computation

We perform statistic computation after each convolutional layer ($L = 17$ for a ResNet-18 V1(He et al., 2015)). Specifically, the output for a convoluation layer has dimension $B \times C_l \times H_l \times W_l$, where B is the batch size, C_l the number of convolutional channels, H_l and W_l, the height and width, respectively. We compute μ_l as by averaging over the B, H_l and W_l dimensions, leading to a vector $\mu_l \in \mathbb{R}^{C_l}$. For Σ_l, we treat the output as $B \times H_l \times W_l$ samples of dimensions C_l, and compute the covariance leading to a matrix $\Sigma_l \in \mathbb{R}^{C_l \times C_l}$.

A.2. Image Synthesis

Algorithm 3 provides detailed steps of the the D3S Image Synthesis algorithm.

A.3. Label Generation

We use the same labelling procedure as prior work (Yin et al., 2023; Yin & Shen, 2023), which is adapted from Shen & Xing (2022), with the modification that we use an ensemble of $M = 5$ models to generate the labels, and average the output in logit space. This requires generated labels for E epochs on the distilled dataset under random augmentations (and storing the augmentation parameters such as a scale, shift, etc.) in order to train the downstream model for E epochs. We provide pseudocode in Algorithm 2.

Algorithm 3 Dataset Distillation with Distribution Shift (D3S)

```
Input: $M$ trained teacher models: $f_{\theta_m}(x)_{m=1}^{M}$
        Precomputed full-dataset statistics for L layers of M models: $\{\mu_{m,l}^T\}_{m,l=1}^{M,L}, \{\Sigma_{m,l}^T\}_{m,l=1}^{M,L}$
        Randomly initialized initial distilled dataset $X_S$ of size $|S|$ and target one-hot labels $y_S$
        Scalar label loss coefficient $\alpha$, Learning rate $\eta$, iterations per batch $T$, batch size $|B|$
Output: Distilled dataset images $X_S$
for $m \leq M, l \leq L$ do
  Initialize: $\tilde{\mu}_{m,l}^S \leftarrow 0 \in R^{C_l}$ for $1 \leq m \leq M, 1 \leq l \leq L$
  Initialize: $\tilde{\Sigma}_{m,l}^S \leftarrow 0 \in R^{C_l \times C_l}$ for $1 \leq m \leq M, 1 \leq l \leq L$    ▷ Initialize running distilled dataset statistics for ISU
end for
for batch index $b$ in $\{1, \cdots, \lceil \frac{|S|}{|B|} \rceil\}$ do
  Select new batch $X_b \subset X_S$ and labels $y_b \subset y_S$ of size $|B|$
  Select model $m \leftarrow b \mod M$    ▷ Pick which model we are optimizing on for the batch

  for Iteration $t$ in $\{1, \cdots, K\}$ do {Main optimization loop}
    Forward $X_b$ through $f_{\theta_m}$, to obtain $\hat{y}_b = f_{\theta_m}(X_b)$ batch mean and covariances $\{\mu_{m,l}^{S,b}\}_{l=1}^{L}, \{\Sigma_{m,l}^{S,b}\}_{l=1}^{L}$
    $\mathcal{L}_{D3S} \leftarrow \alpha \mathcal{L}_{\text{x-ent}}(y, f_\theta(x_S))$    ▷ Set label alignment loss
    for layer l in $\{1, \cdots, L\}$ do
      $\hat{\mu}_{m,l}^{S,b} \leftarrow \frac{1}{b}\mu_{m,l}^{S,b} + (1 - \frac{1}{b})\tilde{\mu}_{m,l}^S$
      $\hat{\Sigma}_{m,l}^{S,b} \leftarrow \frac{1}{b}\Sigma_{m,l}^{S,b} + (1 - \frac{1}{b})\tilde{\Sigma}_{m,l}^S$    ▷ Mix batch with running statistics for ISU
      $\mathcal{L}_{D3S} \leftarrow \mathcal{L}_{D3S} + \frac{1}{2}\left( \log \frac{|\Sigma_{m,l}^T|}{|\hat{\Sigma}_{m,l}^{S,b}|} + \text{Tr}(\Sigma_{m,l}^{T-1}\hat{\Sigma}_{m,l}^{S,b}) + (\mu_{m,l}^T - \hat{\mu}_{m,l}^{S,b})^\intercal \Sigma_l^{T-1}(\mu_{m,l}^T - \hat{\mu}_{m,l}^{S,b}) - C_l \right)$    ▷ Add KL
      divergence for layer $l$
    end for
    $X_b \leftarrow X_b - \eta \frac{\partial \mathcal{L}_{D3S}}{\partial X_b}$    ▷ Optimize $X_b$
  end for

  for model $m$ in $\{1, \cdots, M\}$, layer $l$ in $\{1, \cdots, L\}$ do
    $\tilde{\mu}_{m,l}^S \leftarrow \frac{1}{b}\mu_{m,l}^{S,b} + (1 - \frac{1}{b})\tilde{\mu}_{m,l}^S$
    $\tilde{\Sigma}_{m,l}^S \leftarrow \frac{1}{b}\Sigma_{m,l}^{S,b} + (1 - \frac{1}{b})\tilde{\Sigma}_{m,l}^S$    ▷ Update running statistics for ISU
  end for
end for
Return: $X_S$
```

Algorithm 4 Dataset Distillation with Distribution Shift soft label training procedure

```
Input: Randomly initialized network $f_\theta(x)$ with parameters $\theta$
       Validation batch size $|B|$ and number of training epochs $E$
       Labels for $E$ epochs: $Y$, augmentation auxiliary information $Z_{\text{aug}}$ and batch indices $I$, made with Algorithm 2
       Knowledge distillation temperature $T$, learning rate $\eta$
Output: Trained model $\theta$.
for epoch $e$ in $\pi\{1, \cdots, E\}$ do {Note we randomly permute the epoch order}
  Take $Y_e$, $Z_{e,\text{aug}}$ and $I_e$ corresponding to epoch $e$ from $Y$, $Z_{\text{aug}}$ and $I$, respectively.
  for batch index $b$ in $\{1, \cdots, \lceil \frac{|S|}{|B|} \rceil\}$ do
    Take $Y_b$, $Z_{b,\text{aug}}$ and $I_b$ corresponding to batch $b$ from $Y_e$, $Z_{e,\text{aug}}$ and $I_e$, respectively.
    Sample new batch $X_b \subset X_S$ according to indices $I_b$
    Apply augmentation to $\tilde{X}_b \leftarrow \texttt{aug}(X_b, Z_{b,\text{aug}})$      ▷ Use same augmentation as in the labelling phase
    Forward pass through network $\hat{Y}_b \leftarrow f_\theta(\tilde{X}_b)$
    $\mathcal{L} \leftarrow T^2 D_{KL}(\texttt{softmax}(Y_b/T)||\texttt{softmax}(\hat{Y}_b/T))$      ▷ Knowledge-Distillation loss
    $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$
  end for
end for
Return: $\theta$
```

A.4. Distilled Dataset Validation

We use the labels from Algorithm 2, and train in a knowledge-distillation fashion for E epochs, with hyperparamters in Table 10. We use a student-teacher temperature for knowledge distillation (Hinton et al., 2015) of $T = 20$ for all experiments, consistent with prior work (Yin et al., 2023). It is possible that other choices of temperature perform better. Pseudocode for training with these labels is available in Algorithm 4.

B. Theorem 3.1 details

We provide the proof of Theorem 3.1, which we restate here for convenience:

Theorem 3.1. *If $-\log \hat{p}(y|x)$ is bounded by positive constant C, we have, and we have $l_S = \mathbb{E}_{p_S}[-\log \hat{p}(y|x)]$ and $l_T = \mathbb{E}_{p_T}[-\log \hat{p}(y|x)]$, then:*

$$\begin{aligned} l_T &\leq l_S + \frac{C}{2\sqrt{2}} \sqrt{D_{KL}\left(p_T(x,y)||p_S(x,y)\right)} \\ &= l_S + \frac{C}{2\sqrt{2}} \sqrt{D_{KL}\left(p_T(x)||p_S(x)\right) + D_{KL}\left(p_T(y|x)||p_S(y|x)\right)} \end{aligned}$$

Proof. We have:

$$l_T = \int -\log \hat{p}(y|x) p_T(x,y) dx dy \tag{3}$$

$$l_S = \int -\log \hat{p}(y|x) p_S(x,y) dx dy \tag{4}$$

Then:

$$\begin{aligned}
l_T &= \int -\log \hat{p}(y|x) p_T(x,y) dxdy && (5)\\
&= \int -\log \hat{p}(y|x) \left(p_S(x,y) - p_S(x,y) + p_T(x,y)\right) dxdy && (6)\\
&= \int -\log \hat{p}(y|x) p_S(x,y) dxdy + \int -\log \hat{p}(y|x) \left(p_T(x,y) - p_S(x,y)\right) dxdy && (7)\\
&= l_S + \int -\log \hat{p}(y|x) \left(p_T(x,y) - p_S(x,y)\right) dxdy && (8)\\
&= l_S + \int_{\mathcal{A}} -\log \hat{p}(y|x) \left(p_T(x,y) - p_S(x,y)\right) dxdy - \int_{\mathcal{B}} -\log \hat{p}(y|x) \left(p_S(x,y) - p_T(x,y)\right) dxdy && (9)
\end{aligned}$$

Where we define $\mathcal{A} = \{x,y|p_T(x,y) \geq p_S(x,y)\}$ and $\mathcal{B} = \{x,y|p_T(x,y) < p_S(x,y)\}$, so the second integral is Equation (9) is positive. Note also we have that $0 \leq -\log \hat{p}(y|x) \leq C$. Continuing:

$$\begin{aligned}
l_T &= l_S + \int_{\mathcal{A}} -\log \hat{p}(y|x) \left(p_T(x,y) - p_S(x,y)\right) dxdy - \int_{\mathcal{B}} -\log \hat{p}(y|x) \left(p_S(x,y) - p_T(x,y)\right) dxdy && (10)\\
&\leq l_S + \int_{\mathcal{A}} -\log \hat{p}(y|x) \left(p_T(x,y) - p_S(x,y)\right) dxdy && (11)\\
&\leq l_S + C \int_{\mathcal{A}} \left(p_T(x,y) - p_S(x,y)\right) dxdy && (12)\\
&= l_S + C \int_{\mathcal{A}} \left(p_T(x,y) - p_S(x,y)\right) dxdy && (13)\\
&= l_S + \frac{C}{2} \delta \left(p_T(x,y) || p_S(x,y)\right) && (14)
\end{aligned}$$

Where $\delta\left(p_T(x,y)||p_S(x,y)\right)$ is the Total-Variational distance defined as

$$\delta\left(p_T(x,y)||p_S(x,y)\right) = \frac{1}{2} \int \left|p_T(x,y) - p_S(x,y)\right| dxdy \tag{15}$$

We then apply Pinsker's inequality:

$$\delta(p||q) \leq \sqrt{\frac{1}{2} D_{KL}(p||q)}$$

leading to:

$$\begin{aligned}
l_T &\leq l_S + \frac{C}{2} \delta \left(p_T(x,y) || p_S(x,y)\right) && (16)\\
&\leq l_S + \frac{C}{2\sqrt{2}} \sqrt{D_{KL}\left(p_T(x,y)||p_S(x,y)\right)} && (17)\\
&= l_S + \frac{C}{2\sqrt{2}} \sqrt{D_{KL}\left(p_T(x)||p_S(x)\right) + D_{KL}\left(p_T(y|x)||p_S(y|x)\right)} && (18)
\end{aligned}$$

□

When $D_{KL}(p||q) \geq 2$, Pinsker's inequality is vacuous, but we can use the Bretagnolle–Huber inequality instead, leading to:

Theorem B.1. *If $-\log \hat{p}(y|x)$ is bounded by positive constant C, we have:*

$$\begin{aligned} l_T &\leq l_S + \frac{C}{2}\sqrt{1 - e^{-D_{KL}(p_T(x,y)||p_S(x,y))}} \\ &= l_S + \frac{C}{2\sqrt{2}}\sqrt{1 - e^{-D_{KL}(p_T(x)||p_S(x)) - D_{KL}(p_T(y|x)||p_S(y|x))}} \end{aligned}$$

Proof. Follow the proof of Theorem 3.1 until Equation (14). Then apply the Bretagnolle–Huber inequality instead:

$$\delta(p||q) \leq \sqrt{1 - e^{-D_{KL}(p||q)}} \tag{19}$$

$$\begin{aligned} l_T &\leq l_S + \frac{C}{2}\delta\left(p_T(x,y)||p_S(x,y)\right) && (20) \\ &\leq l_S + \frac{C}{2}\sqrt{1 - e^{-D_{KL}(p_T(x,y)||p_S(x,y))}} && (21) \\ &= l_S + \frac{C}{2\sqrt{2}}\sqrt{1 - e^{-D_{KL}(p_T(x)||p_S(x)) - D_{KL}(p_T(y|x)||p_S(y|x))}} && (22) \end{aligned}$$

□

C. Additional Experiments

C.1. CIFAR-10/100

In this section we verify that our method also works for smaller datasets, namely CIFAR-10/CIFAR-100. As before, we distill from a ResNet-18, and train on a ResNet-18. For CIFAR-100, we train for 800 epochs (similar to previous baselines), and for CIFA0-10 we train for 1600. Table 5 shows that our method is competitive with other methods on smaller dataset as well. We reiterate that D3S is primarily designed with large-scale distillation in mind, and that high performance on CIFAR-10/100 is not the main goal. (Yin et al., 2023) and (Yin & Shen, 2023) do not report performance on CIFAR-10, so we do not have baselines.

	SRe2L	CDA	D3S (Ours)
10/cls	23.48±0.80	49.8	**56.73±1.30**
50/cls	51.35±0.79	64.4	**69.65±0.26**

Table 5. Distillation performance on CIFAR-100

	D3S (Ours)
10/cls	44.93±1.36
50/cls	75.81±2.27

Table 6. Distillation performance on CIFAR-10

D. Implementation Details

D.1. Dataset Details

See Table 7 for details of the datasets used in this paper.

Table 7. Details of Datasets

	Tiny-ImageNet (Le & Yang, 2015)	ImageNet-1K (Deng et al., 2009)	ImageNet-21K (Ridnik et al., 2021)
Number of Classes	200	1000	10450
Training Set Size	100,000	1,281,167	11,060,223
Validation Set Size	10,000	50,000	522,500
Resolution	64×64	224×224	224×224
Source Model Accuracy	$59.83 \pm 0.03\%$	$69.83 \pm 0.03\%$	$38.06 \pm 0.02\%$

Table 8. Hyperparameters used for training source models

Parameter	Tiny-ImageNet	ImageNet-1K	ImageNet-21K
Model	ResNet-18	ResNet-18	ResNet-18
Initialization	Random	Random	Pretrained on IN-1K
Optimizer	SGD	SGD	AdamW
Learning Rate/Momentum	0.2/0.9	0.1/0.9	3e-4
Weight Decay	1e-4	1e-4	1e-4
Batch Size	256	256	1024
Augmentation	RandomResizedCrop + Horizontal Flip	RandomResizedCrop + Horizontal Flip	CutoutPIL, RandAugment
LR Scheduler	CosineAnneal, Linear warmup = 5	StepLR (decay 0.1 every 30 epochs)	OneCycleLR
Epochs	50	90	80
Accuracy	$59.83 \pm 0.03\%$	$69.83 \pm 0.03\%$	$38.06 \pm 0.02\%$

D.2. Source model training details

We use $M = 5$ models for all experiments, except where otherwise stated such as in Section 6. For training these models we have the configurations in Table 8.

These configurations largely follow prior work. We use the library from Ridnik et al. (2021) for code for training the ImageNet-21K models. For ImageNet-1K, we use the Pytorch library code for training. For Tiny-ImageNet ResNet-18s, we replace the first 7x7 convolution with a 3x3 one, and remove the first max-pool layer, consistent with prior work (Yin et al., 2023).

D.3. Image Synthesis Details

We provide hyperparameters for image synthesis in Table 9. For the augmentation, we use the recently propose curriculum augmentation Yin & Shen (2023), where we gradually increase the size of magnitudes of the augmentation linearly up to the full scale. The full scale is the standard RandomResizeCrop with max scale 1.0 and min scale 0.08.

Table 9. Hyperparameters used for Image Synthesis

Parameter	Tiny-ImageNet	ImageNet-1K	ImageNet-21K
Optimizer	Adam	Adam	Adam
Adam β_1, β_2	0.5/0.9	0.5/0.9	0.5/0.9
Initial LR	0.4	0.25	0.05
Batch Size	200	200	200
Augmentation	RandomResizedCrop + Horizontal Flip	RandomResizedCrop + Horizontal Flip	RandomResizedCrop + Horizontal Flip
LR Scheduler	CosineAnneal	CosineAnneal	CosineAnneal
Iterations per Batch (K)	4000	1000	1000

D.4. Validation Details

Table 10 shows the hyperparameters used during validation of the distilled dataset, used for tables 1, 2 and 3. We use a fixed temperature of $T = 20$ for the knowledge-distillation loss during validation. We report the average and standard deviation over $n = 5$ indepedently trained networks for Table 1 for ResNet-18s, and $n = 3$ for other models. For results in Section 7, the epoch parameter is varied.

Table 10. Hyperparameters used for Validation

Parameter	Tiny-ImageNet	ImageNet-1K	ImageNet-21K
Optimizer	SGD	AdamW	AdamW
Learning Rate/Momentum	0.2/0.9	1e-3	2e-3
Weight Decay	1e-4	1e-4	1e-4
Batch Size	64	128	128
Augmentation	RandomResizedCrop + Horizontal Flip	RandomResizedCrop + Horizontal Flip	RandomResizedCrop + Horizontal Flip
LR Scheduler	CosineAnneal	CosineAnneal	CosineAnneal
Epochs	100	300	300

D.5. Hardware

All experiments were run on either single 4090s with 24GB VRAM or single RTX 6000 Adas with 48GB VRAM.

D.6. Runtime and Memory Consumption

Image synthesis on Tiny-ImageNet consumes 7729`MiB`, and for ImageNet-1K and ImageNet-21K 9851`MiB` for batch sizes of 200 used for all experiments. For 1000 iterations, it takes 155s for Tiny-ImageNet and 170s for ImageNet-1K and ImageNet-21K on a RTX 6000 Ada. Combined with Table 9, the procedure takes a total of 17.2h for Tiny-ImageNet 100 IPC, 47.2h for ImageNet-1K 200 IPC, and 50.0h for ImageNet-21K 20IPC.

D.7. Training Time Experiments

This section contains details of the experiments in Section 7. As seen in Algorithm 2, we generate labels for E epochs, which is 100 for Tiny-ImageNet or 300 for ImageNet-1K/21K (see table Table 10. For the experiments in Section 7 we vary the number of epochs labelled, and the training epochs. Let the training duration be E_T and the and the labelled epochs be E_L. If we have $E_L = E_T$(the default configuration), then we train for a random permutation of the epochs, i.e. the epochs maybe labelled $[0, 1, 2, 3, 4]$ for $E_L = 5$ but we may train in the order $[3, 2, 1, 0, 4]$, and we randomize over runs to add randomness. In the case that $E_L > E_T$, we take the first E_L epochs as the random permutation. E.g. if $E_L = 5$ and $E_T = 3$, then we might train for epoch indices $[1, 0, 4]$. When $E_L < E_T$, we concatenate random permutations of the E_L labelled epochs until we reach E_T. E.g. if $E_L = 5$ and $E_T = 10$, then we might train for epoch indices $[4, 2, 1, 3, 0, 4, 0, 3, 2, 1]$. This requires a small modification of the code from (Shen & Xing, 2022).

E. Image Visualization

Here we provide visualizations of distilled images from Tiny-ImageNet, ImageNet-1K, and ImageNet-21K.

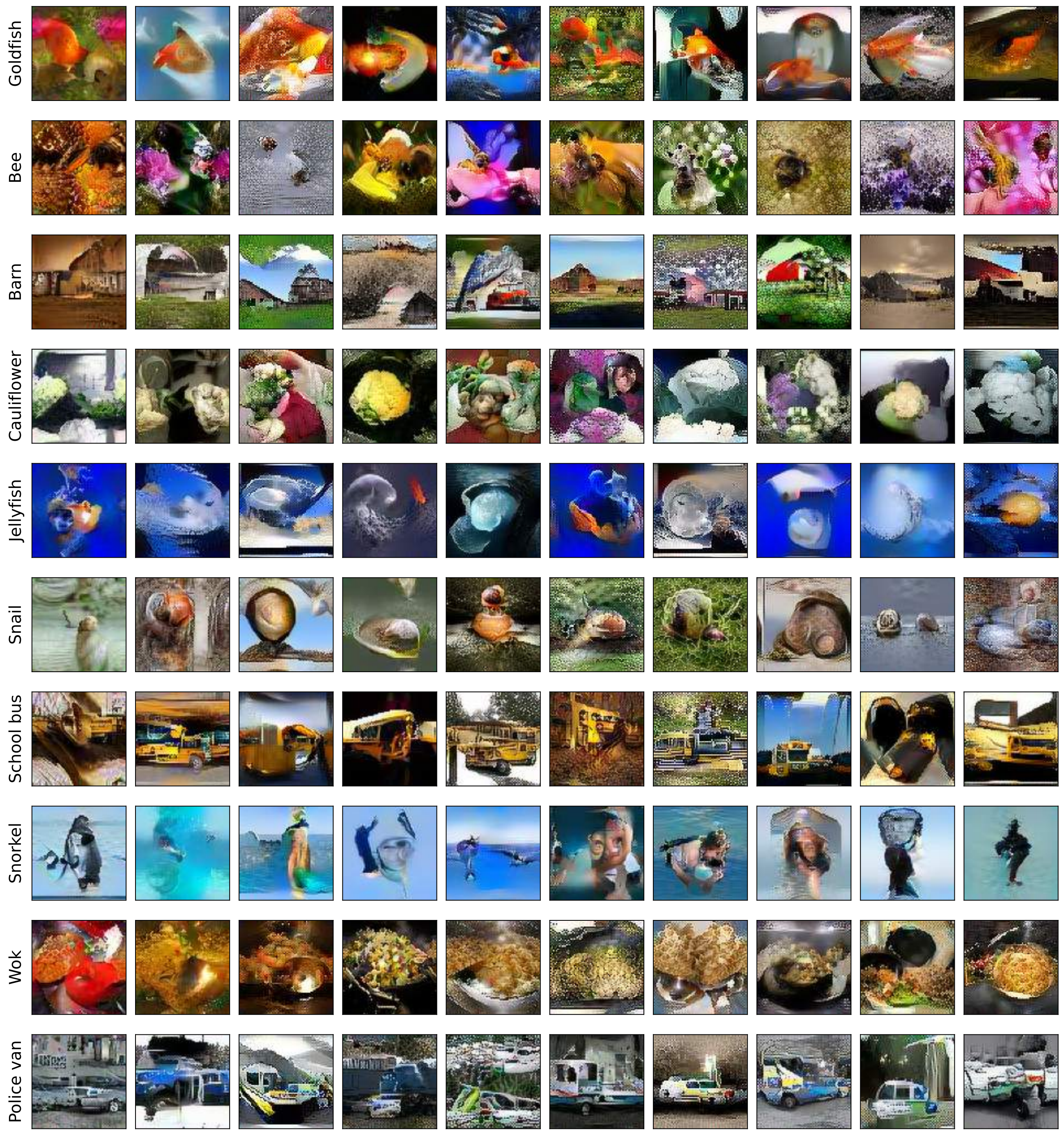

Figure 5. Visualization of distilled images from D3S on Tiny-ImageNet

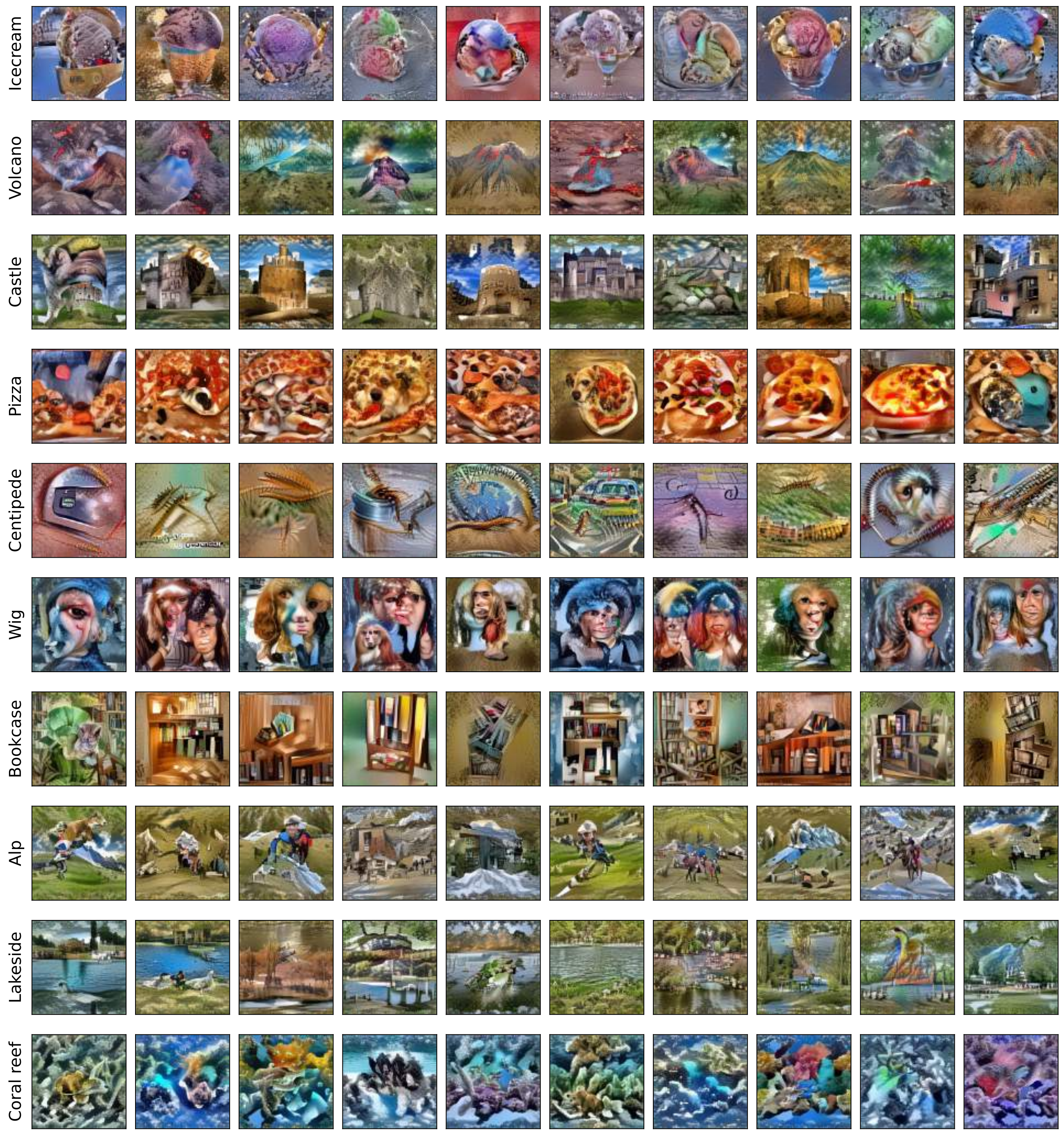

Figure 6. Visualization of distilled images from D3S on ImageNet-1K

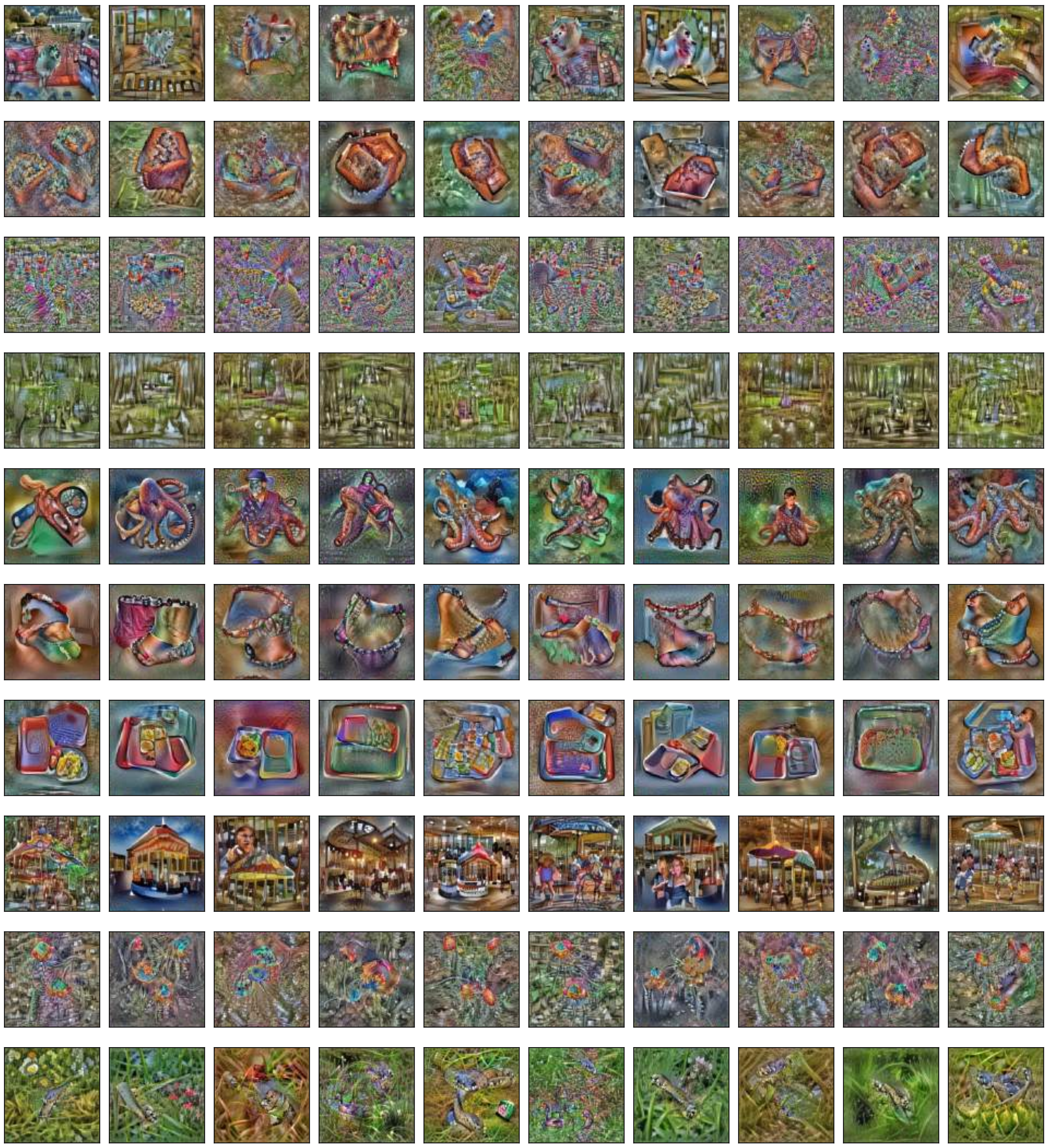

Figure 7. Visualization of distilled images from D3S on ImageNet-21K