Generalizing Dataset Distillation via Deep Generative Prior

George Cazenavette[1] Tongzhou Wang[1] Antonio Torralba[1] Alexei A. Efros[2] Jun-Yan Zhu[3]

[1]Massachusetts Institute of Technology [2]UC Berkeley [3]Carnegie Mellon University

georgecazenavette.github.io/glad

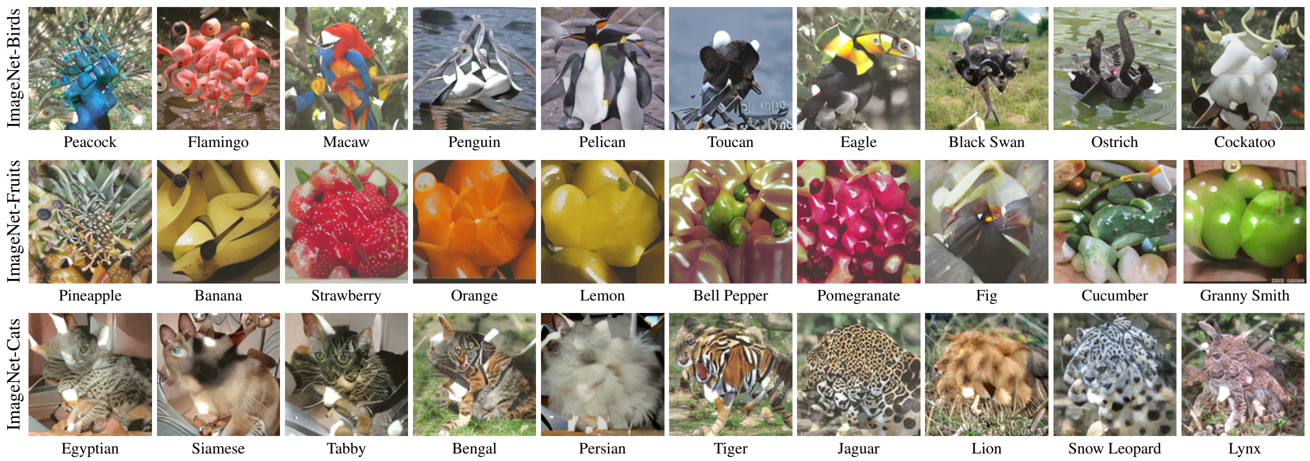


Figure 1. Visualizations of distilled images from three ImageNet subsets. Rather than distilling into pixel space, our *Deep Generative Prior* constrains the images to be more coherent, leading to better cross-architectural generalization.

Abstract

Dataset Distillation aims to distill an entire dataset's knowledge into a few synthetic images. The idea is to synthesize a small number of synthetic data points that, when given to a learning algorithm as training data, result in a model approximating one trained on the original data. Despite a recent upsurge of progress in the field, existing dataset distillation methods fail to generalize to new architectures and scale to high-resolution datasets. To overcome the above issues, we propose to use the learned prior from pre-trained deep generative models to synthesize the distilled data. To achieve this, we present a new optimization algorithm that distills a large number of images into a few intermediate feature vectors in the generative model's latent space. Our method augments existing techniques, significantly improving cross-architecture generalization in all settings.

1. Introduction

Many recent advancements in machine learning come from combining large networks and big data. Such trained models have shown strong capabilities to perform a wide range of diverse tasks [12, 22, 48] and are considered by some as an ongoing paradigm shift [9]. While such approaches show great potential to improve the frontier of AI, we, as a scientific community, are also curious about the underlying principles and limitations. Do networks have to be large to express the functions of interest? Do datasets have to be big? Can training on "small data" be equally successful?

The seminal work on Knowledge Distillation [31] and recent discoveries such as Lottery Ticket Hypothesis [26] have revealed small models are often sufficient to approximate the same functions as large trained models (which are sometimes useful in optimization). Dataset Distillation, proposed by [61], investigates the analogous yet orthogonal question on datasets: is there a small succinct dataset sufficient for training models? In other words, Dataset Distillation aims to distill a large dataset into a small (synthetic) one, such that training on the small dataset yields comparable performance (Figure 2). Since its proposal, Dataset Distillation has gained much attention in the research community, leading to many applications [14, 23, 43, 53], and a growing series of methods that solve the distillation problem: generating a discrete set of images capable of effectively training a model [13, 60, 61, 64, 66–68]. In optimizing for a small, synthetic vision dataset, such methods typically optimize the *raw* pixel values of the images.

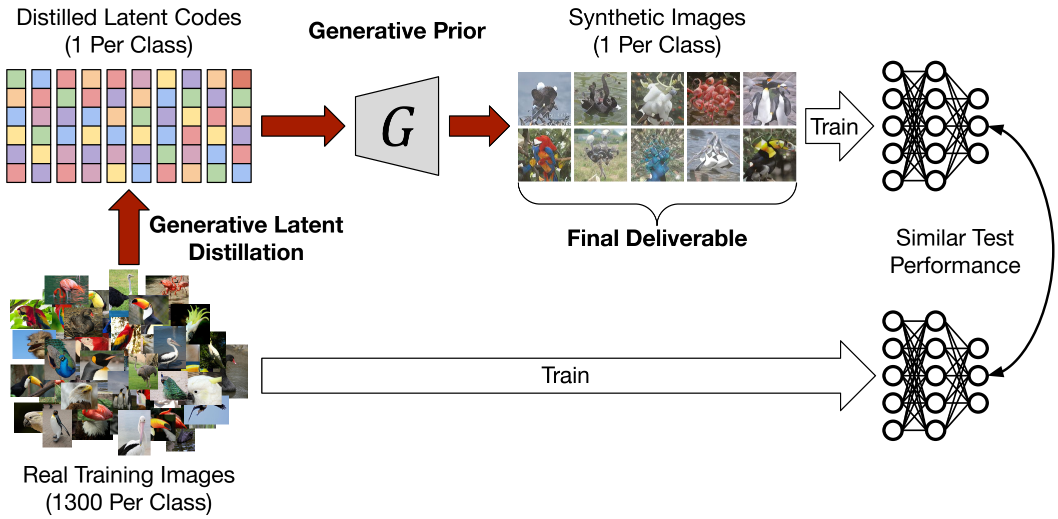


Figure 2. Rather than directly distilling a dataset into synthetic pixels (like all previous methods), our new work instead distills into the latent space of a *deep generative prior*. This enforces a tuneable amount of coherence in the output synthetic images, leading to far better generalization to new architectures.

Unfortunately, these methods face two major challenges, limiting both their scientific value and empirical applications. First, the distilled synthetic dataset is often optimized w.r.t. a specific network architecture, but does not generalize well to other architectures. Second, while producing insightful distilled images on toy datasets, these methods generally fail to work well on larger-resolution datasets (e.g., $\geq 128 \times 128$ resolution) and tend to distill visually noisy images with subpar performance.

In this work, we argue that both issues are caused by parameterizing the synthetic dataset in pixel space. Directly optimizing pixels can be susceptible to learning high-frequency patterns that overfit the specific architecture used in training. To address this, we consider regularizing the distillation process to some prior that may help cross-architecture generalization. However, how and where to perform this regularization poses a delicate balance. For example, restricting our synthetic set to the real data manifold can significantly reduce the cross-architecture performance gap but is too strong a regularization to learn good distilled datasets. In the limit, it reduces to dataset/coreset selection [7, 10, 29, 57], which is known to not work as well [13, 61, 64, 67].

We propose Generative Latent Distillation (`GLaD`), which utilizes a *deep generative prior* by parameterizing the synthetic dataset in the *intermediate* feature space of generative models, such as Generative Adversarial Networks (GANs) [28]. It encourages the learned datasets to be more generalizable to novel architectures but is also lax enough to *not* prohibitively restrict the expressiveness of the distilled dataset. `GLaD` acts as an add-on module and can easily be applied to all existing and future methods of dataset distillation.

There is flexibility in choosing which generative model to use as our prior. By using a generator trained on the target dataset (which is input to the distillation algorithm), our prior uses no additional data or information but consistently improves various distillation algorithms. However, for the sole purpose of obtaining the best distilled synthetic dataset, we may use more powerful generators trained on larger datasets and also obtain significant gains. On the other extreme, we explore using randomly initialized generators and generators trained on out-of-distribution datasets. We show that they generate aesthetically pleasing synthetic images with distinct visual characteristics and also achieve comparable distillation performance. In short, while different generator choices affect distillation results in interesting ways, `GLaD` consistently improves performance over many datasets and multiple distillation algorithms.

Within a deep generative model, there is a spectrum of different latent space choices, corresponding to different layers in the model [2, 47, 71]. Our analysis reveals a trade-off between realism (earlier layer latents) and flexibility (later layer latents), and highlights that using an intermediate latent space achieves a nice balance and consistent performance gain (over the wildly used raw-pixel parametrization).

In Section 4, we perform extensive experiments on CIFAR-10 (a common dataset distillation benchmark) and ImageNet subsets at resolutions up to 512×512. We integrate `GLaD` with three distinct current distillation algorithms (Gradient Matching [67], Distribution Matching [66], and Trajectory Matching [13]), and consistently observe significant improvements in cross-architecture generalization of the three currently most accessible methods of dataset distillation. Our analysis of results from different configurations provides a better understanding of the effect of different generative models and latent spaces. Additionally, our method drastically reduces the high-frequency noise present in high-resolution datasets distilled into pixel space, leading to visually pleasing images that may have implications in artistic and design applications (e.g., [14]).

GLaD is a plug-and-play addition to any existing and future distillation methods, allowing them to scale up to more realistic datasets and generalize better to different architectures. Given the goal of distilling large-scale datasets, we propose that leveraging generative models and differentiable parameterizations is a natural path forward.

Our contributions are summarized as follows:

- We propose *Generative Latent Distillation* (GLaD) to add a *deep generative prior* to Dataset Distillation by distilling into an intermediate feature space of a generative model trained on real data.
- Our extensive analysis and ablations highlight the importance of a *deep generative prior* in addressing the two major challenges of Dataset Distillation: cross-architecture generalization and high-resolution data.
- We show that GLaD is robust to the type of generator used, still performing well with randomly initialized generators and those trained on other datasets.
- Our method acts as a plug-and-play addition to all existing and future methods of dataset distillation, allowing researchers to easily use it for future work.

2. Related Work

Dataset Distillation was introduced by [61] as an investigation as to how training on very little data could update a model to certain desired behaviors. A number of improvements have been made since then, including soft learned labels [54], data augmentation [64], and trajectory/gradient matching [13, 67]. Many applications have also been explored, such as neural architecture search [32], continual learning [64, 67], etc. Some works [32, 33] also employ generative models but train them from scratch to produce good synthetic training samples, in a manner completely different from our formulation. Several concurrent works tackle the dataset distillation via a re-parameterization of the distilled data as a set of bases [21, 42] or a greater number of lower-resolution images [39], motivated by a memory compression standpoint. Our method differs in that we use generative model parameterizations to achieve better distillation performance. Concurrent work [65] also learns synthetic datasets in the latent space of a generative model by first inverting the *entire* training set into this latent space before fine-tuning the latents on the distillation objective.

GAN Inversion and Latent Spaces. Our work applies generative priors by parameterizing distilled images into different latent spaces of GANs. This is related to the line of research of GAN inversion [11, 70], whose goal is to project

Algorithm 1 Generative Latent Distillation

Input: Alg: Distillation algorithm (MTT, DC, or DM).
Input: $\mathcal{T}$: Real training set.
Input: $\mathcal{A}$: Differentiable augmentation function.
Input: G: Pre-trained generator.
Input: P_z: Distribution of latent initializations.
$\mathcal{Z} \sim P_z$ ▷ Initialize distilled latents
for each distillation step... **do**
    $\mathcal{S} = G(\mathcal{Z})$ ▷ Get images from latents
    $\mathcal{L} = \mathrm{Alg}(\mathcal{S}, \mathcal{T})$ ▷ Compute distillation loss
    $\mathcal{Z} \leftarrow \mathrm{SGD}(\mathcal{Z}; \mathcal{L})$ ▷ Update $\mathcal{Z}$ with respect to $\mathcal{L}$
end for
Output: Distilled images $\mathcal{S} = G(\mathcal{Z})$

a real input image to GAN latent codes for image editing and data augmentation purposes [15, 16, 34, 55, 62, 69]. These works have proposed various latent spaces (roughly similar to different activation spaces) [2, 3, 63, 71], wherein image editing or model fine-tuning can be efficiently performed [5, 6, 46, 49]. The choice of specific latent space represents a trade-off between expressiveness and reconstruction quality, as shown by recent studies [56, 72]. In our experiments, we leverage a spectrum of such latent spaces designed for the StyleGAN-family of models [35, 37, 38, 50] and discover that such a trade-off also exists in our case. The ideal distilled images are not realistic but can benefit from some level of generative prior. Our experiments perform an in-depth analysis on this aspect and show that an *intermediate* generative prior (i.e., starting from an intermediate layer) yields the best performance.

3. Generative Latent Distillation (GLaD)

To date, all existing methods of dataset distillation [8, 13, 44, 45, 60, 61, 64, 67] have relied on a "backbone" architecture to formulate the distillation objective. Since optimizing distilled images in pixel space allows too much freedom to over-fit to the backbone architecture, we propose introducing a *deep generative prior* to the distillation process as a form of regularization by optimizing latent codes of a pre-trained generative model rather than raw pixels. We call our method *Generative Latent Distillation* (GLaD).

3.1. Preliminaries on Dataset Distillation Methods

For completeness, we briefly review the three methods of dataset distillation on which we conduct experiments: Gradient Matching (DC), Distribution Matching (DM), and Matching Training Trajectories (MTT). These three methods all seek to distill a small synthetic dataset $\mathcal{S}$ such that a model trained from scratch on $\mathcal{S}$ will have similar performance as a model trained on the full, real dataset $\mathcal{T}$.

Dataset Condensation (DC) enforces that the gradients of the classification loss ℓ w.r.t. the synthetic images match

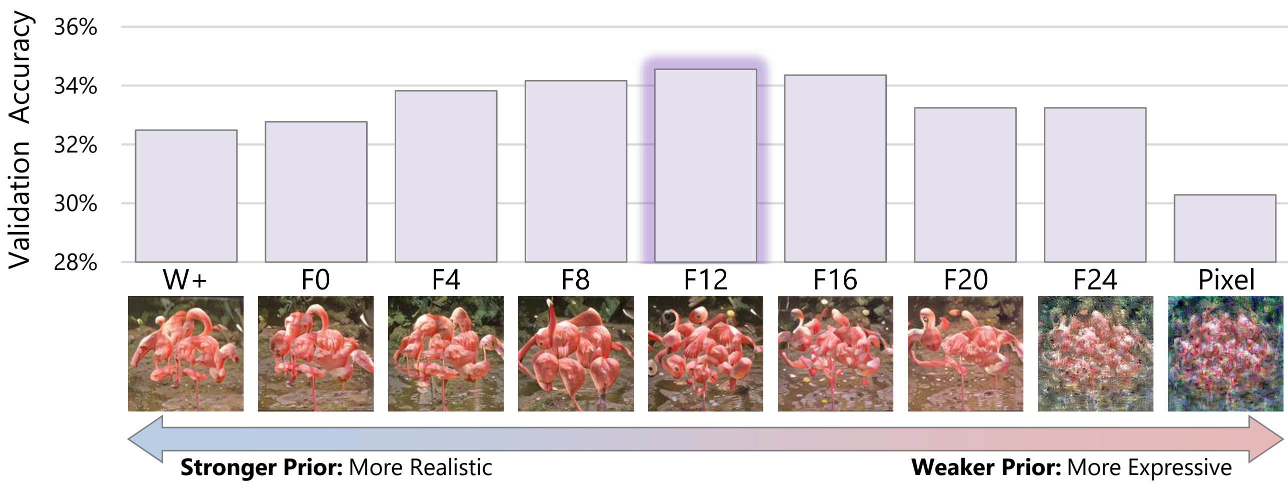


Figure 3. **Cross-Architecture performance of MTT [13] per distillation space averaged over ImageNet-[A,B,C,D,E].** Using spaces of earlier layers (**left side**) imposes stronger priors and yields more realistic images but is not as expressive as using spaces of later layers (**right side**). A proper amount of generative prior is helpful for cross-architecture generalization, while too strong a prior limits expressivity and thus hurts distillation performance. An intermediate space (e.g., F12) balances this trade-off.

those of the real images. For a network with parameters θ trained on the synthetic data for some number of iterations, the gradient matching loss is

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

Distribution Matching (DM) takes a different approach and instead requires that a feature extractor yields similar output for real and synthetic images of a corresponding class. For a randomly initialized feature extractor ψ, the distribution matching loss is

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

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

Both the DC and DM methods are augmented with Differential Siamese Augmentation (DSA) such that the random differential augmentations applied to real and synthetic images at each iteration are initialized with the same random seed.

Matching Training Trajectories (MTT) focuses on consistent training over a longer time horizon by encouraging the synthetic data to induce similar update trajectories in parameter space. Prior to distillation, many *expert trajectories* $\{\theta_t^*\}_0^T$ are obtained by training networks from scratch on the full real dataset and storing parameter snapshots at given intervals. At each distillation step, a random expert trajectory and starting timestamp θ_t^* are sampled. A student network is then initialized at the given expert timestamp $\hat{\theta}_t := \theta_t^*$ and trained for N iterations on the *synthetic data*. The trajectory-matching loss is then calculated as the normalized mean-squared error between the final parameters of the student network $\hat{\theta}_{t+N}$ and those of a future timestep (M steps ahead) of the expert trajectory θ_{t+M}^*:

$$\mathcal{L}_{\texttt{MTT}} = \frac{\|\hat{\theta}_{t+N} - \theta_{t+M}^*\|^2}{\|\theta_t^* - \theta_{t+M}^*\|^2}. \tag{3}$$

We make use of a recent method, dubbed TESLA [19], which solves the memory issue of MTT by re-formulating the loss function to an equivalent form that only requires the storing of one set of network gradients at a time, thereby reducing the memory usage from linear to constant w.r.t. the number of inner loops N. However, we do not use this method's soft label assignment.

3.2. GLaD: Adding a Deep Generative Prior

Rather than naively optimizing the synthetic pixels directly (as in all previous methods [8, 13, 44, 45, 60, 61, 64, 67]), we propose applying a *deep generative prior* to our distillation process by means of distilling into the latent space of a pre-trained generative model. We find that such a prior greatly increases the generalization of the synthetic dataset to architectures other than the "backbone" model used in the distillation process (the lack of such generalization being one of the largest limitations of previous methods). Furthermore, our new parametrization facilitates the distillation of even larger-resolution synthetic data devoid of the high-frequency noise induced by previous distillation methods. Lastly, the deep generative prior acts as a plug-and-play addition to any existing and future methods of dataset distillation.

Concretely, we consider a deep generative model G that outputs samples $G(z)$ given latent vector z (e.g., a GAN).

Distil. Alg.	Distil. Space	ImNet-A	ImNet-B	ImNet-C	ImNet-D	ImNet-E	ImNette	ImWoof	ImNet-Birds	ImNet-Fruits	ImNet-Cats
MTT [13]	Pixel	33.4±1.5	34.0±3.4	31.4±3.4	27.7±2.7	24.9±1.8	24.1±1.8	16.0±1.2	25.5±3.0	18.3±2.3	18.7±1.5
	GLaD (Ours)	**39.9**±1.2	**39.4**±1.3	**34.9**±1.1	**30.4**±1.5	**29.0**±1.1	**30.4**±1.5	**17.1**±1.1	**28.2**±1.1	**21.1**±1.2	**19.6**±1.2
DC [67]	Pixel	38.7±4.2	38.7±1.0	33.3±1.9	26.4±1.1	27.4±0.9	28.2±1.4	17.4±1.2	28.5±1.4	20.4±1.5	19.8±0.9
	GLaD (Ours)	**41.8**±1.7	**42.1**±1.2	**35.8**±1.4	**28.0**±0.8	**29.3**±1.3	**31.0**±1.6	**17.8**±1.1	**29.1**±1.0	**22.3**±1.6	**21.2**±1.4
DM [66]	Pixel	27.2±1.2	24.4±1.1	23.0±1.4	18.4±0.7	17.7±0.9	20.6±0.7	14.5±0.9	17.8±0.8	14.5±1.1	14.0±1.1
	GLaD (Ours)	**31.6**±1.4	**31.3**±3.9	**26.9**±1.2	**21.5**±1.0	**20.4**±0.8	**21.9**±1.1	**15.2**±0.9	**18.2**±1.0	**20.4**±1.6	**16.1**±0.7

Table 1. **ImageNet (128×128) Performance on Unseen Architectures**. These results come from training AlexNet, VGG11, ResNet18, and a Vision Transformer on our synthetic datasets (that were distilled using a ConvNet) and averaging their performances on the real validation sets. Applying the deep generative prior by distilling into F-space rather than pixel space significantly improves the cross-architecture generalization of all methods across all sampled datasets.

Distil. Alg.	Distil. Space	ImNet-A	ImNet-B	ImNet-C	ImNet-D	ImNet-E
DC	Pixel	52.3±0.7	45.1±8.3	40.1±7.6	36.1±0.4	38.1±0.4
	GLaD (Ours)	**53.1**±1.4	**50.1**±0.6	**48.9**±1.1	**38.9**±1.0	**38.4**±0.7
DM	Pixel	52.6±0.4	50.6±0.5	47.5±0.7	35.4±0.4	36.0±0.5
	GLaD (Ours)	**52.8**±1.0	**51.3**±0.6	**49.7**±0.4	**36.4**±0.4	**38.6**±0.7

Table 2. Performance with 10 images/class.

Distillation Method	Distillation Space	**Unseen** Evaluation Architecture				
		AlexNet	ResNet18	VGG11	ViT	Average
MTT [13]	Pixel	26.8±0.6	23.4±1.3	24.9±0.8	21.2±0.4	24.1±0.8
	GLaD (Rand G)	27.4±0.3	30.1±1.2	29.0±0.8	21.9±0.3	27.1±0.7
	GLaD (Trained G)	**27.9**±0.6	**30.2**±0.6	**31.3**±0.7	**22.7**±0.4	**28.0**±0.6
DC [67]	Pixel	25.9±0.2	27.3±0.5	28.0±0.5	22.9±0.3	26.0±0.4
	GLaD (Rand G)	**30.1**±0.5	27.3±0.2	28.0±0.9	21.2±0.6	**26.6**±0.5
	GLaD (Trained G)	26.0±0.7	**27.6**±0.6	**28.2**±0.6	**23.4**±0.2	26.3±0.5
DM [66]	Pixel	22.9±0.2	22.2±0.7	23.8±0.5	21.3±0.5	22.6±0.5
	GLaD (Rand G)	23.7±0.3	21.7±1.0	24.3±0.8	21.4±0.2	22.8±0.6
	GLaD (Trained G)	**25.1**±0.5	**22.5**±0.7	**24.8**±0.8	**23.0**±0.1	**23.8**±0.5

Table 3. **CIFAR-10 Performance on Unseen Architectures**. Unlike the high-resolution data, we only see a large improvement to MTT and a moderate improvement to DM. Interestingly, we also see that distilling into the latent space of an *un-trained* generator still yields results on-par or better than pixel-space distillation.

At distillation time, we parameterize the small synthetic dataset $\mathcal{S}$ as

$$\mathcal{S} \triangleq \{G(z) \colon z \in \mathcal{Z}\}, \tag{4}$$

where $\mathcal{Z}$ is a set of latent vectors. Since G is fully differentiable, we can optimize $\mathcal{Z}$ w.r.t. any distillation objective, such as $\mathcal{L}_{\texttt{DC}}$, $\mathcal{L}_{\texttt{DM}}$ or $\mathcal{L}_{\texttt{MTT}}$. Please see Algorithm 1 for a complete write-up of our method.

3.3. Choosing a Generative Model and Latent Space

For a flexible and effective parameterization, we fill the role of our deep generative prior with the recently proposed StyleGAN-XL [51], a modified version of StyleGAN3 [36]. StyleGAN generators can not only output high-fidelity images [35], but also (1) provide multiple flexible latent spaces for parameterizing images [2, 47] and (2) inherently impose diverse and interesting priors via architecture (even at random initialization) [4]. In our experiments, this allows us to probe the effects of choosing latent spaces from different layers and using generators trained on out-of-distribution datasets or even at random initialization.

Distilling into the latent space of StyleGAN-XL can be

Distillation Space	ImNet-A	ImNet-B	ImNet-C	ImNet-D	ImNet-E
Pixel	38.3±4.7	32.8±4.1	27.6±3.3	25.5±1.2	23.5±2.4
GLaD (ImageNet G)	37.4±5.5	**41.5**±1.2	**35.7**±4.0	27.9±1.0	29.3±1.2
GLaD (Random G)	**39.3**±2.0	40.3±1.7	35.0±1.7	27.9±1.4	**29.8**±1.4
GLaD (Pokémon G)	39.1±2.0	39.4±1.5	35.3±1.3	**28.0**±1.2	29.5±1.3
GLaD (FFHQ G)	38.3±5.2	40.2±1.1	34.9±1.1	27.2±0.9	29.4±2.1

Table 4. Higher-resolution (256×256) datasets distilled (using DC) into the latent space of a randomly-initialized generator as well as generators on Pokémon or FFQH still realize significant improvements in cross-architecture generalization over their pixel-space counterparts.

thought of as a pseudo-inversion task. However, we found that distilling into even extended W⁺ latent space[1] [2], the most flexible of the traditional StyleGAN inversion spaces, was too restrictive for our objective (see Figure 3). It limits the synthetic images to be realistic, but, unlike real image inversion, these images are optimized for the distillation task and do not require realism. Prior inversion works propose the "Fn" spaces in StyleGAN as an alternative that allows images to be more diverse and flexible [47, 71]. As such, we choose to distill into these "Fn" spaces, which means optimizing the n-th hidden layer of StyleGAN-XL's "synthesis" network's latent representation along with all subsequent W⁺ modulation codes. Note that in this work, z and $\mathcal{Z}$ refer to the concatenation of the Fn feature map with the W⁺ codes, **not** the traditional StyleGAN z-space. Please see the supplemental material for details of the StyleGAN-XL architecture and the "Fn" spaces.

Since these "Fn" spaces are from intermediate layers and do not have associated prior distributions, we initialize our latent z vectors using the empirical distribution of latent vectors of the corresponding classes. With access to the earlier layers of G, this can be easily computed. Please see the supplemental material for details of our initialization scheme.

3.4. Memory Saving via Checkpointing

As the forward pass through modern deep generative models usually requires copious amounts of GPU VRAM, our method (if implemented naively) becomes difficult to run on a limited number of GPUs. To circumvent this issue, we

[1] Here W space refers to the output space of StyleGAN-XL's MLP "mapping" network, and W⁺ allows for different W space vectors at different layers.

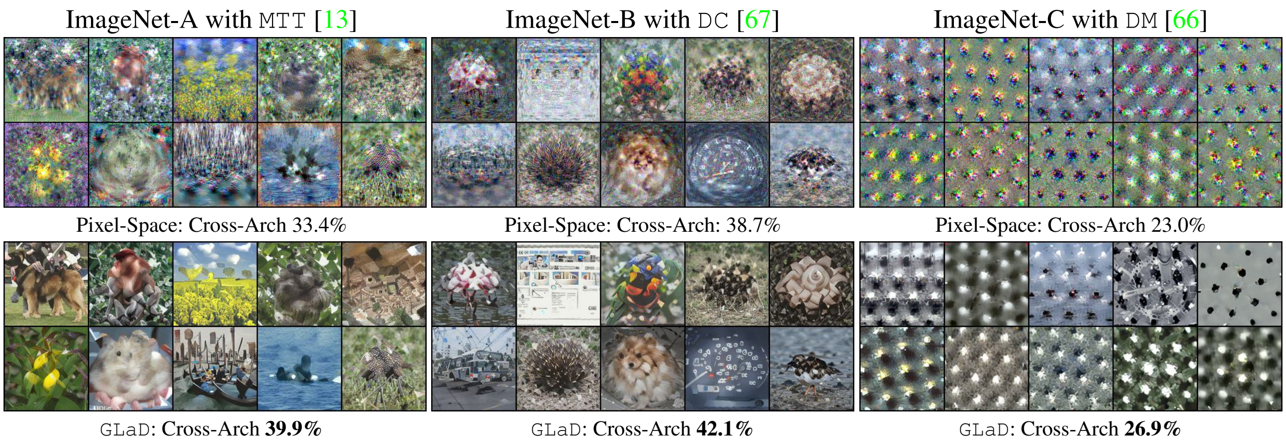


Figure 4. Across all sampled ImageNet subsets and all distillation methods, the addition of the generative prior (**bottom**) offers significantly better cross-architecture generalization than pixel-space distillation (**top**).

(a) Pixel-Space Distillation; 24.1% Cross Arch. Acc.

(b) F2-Space Distillation, Random Generator; 27.1% Cross Arch. Acc.

(c) F2-Space Distillation, Pre-Trained Generator; 28.0% Cross Arch. Acc.

Figure 5. CIFAR-10 distillations using `MTT`. The images distilled with both the random and pre-trained generators lead to better cross-architecture generalization than those distilled into pixel space.

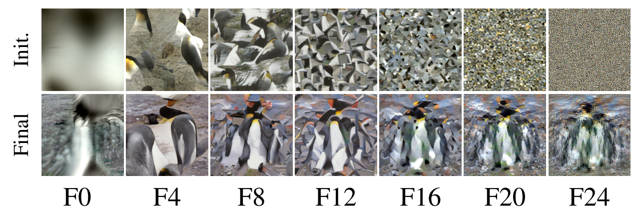


Figure 6. Replacing the feed-forward initialization of the F latents with random noise has drastically different effects in the initial image (**top**) depending on which layer is used. In the optimized images (**bottom**), we see artifacts with varying levels of granularity depending on the layer.

employ a technique inspired by gradient checkpointing [17]. At each distillation iteration, we first obtain our synthetic images $\mathcal{S} = G(\mathcal{Z})$ *without* tracking any gradients. We then calculate our distillation loss $\mathcal{L}$, compute the gradient of this loss with respect to our synthetic images ($\partial\mathcal{L}/\partial\mathcal{S}$), and delete the computation graph used to compute $\mathcal{L}$ and its gradient. To compute $\partial\mathcal{L}/\partial\mathcal{Z}$, we *re*-compute the forward pass through G, $\mathcal{S} = G(\mathcal{Z})$, this time tracking gradients such that we know $\partial\mathcal{S}/\partial\mathcal{Z}$. From here, application of the chain rule gives us $\partial\mathcal{L}/\partial\mathcal{Z} = (\partial\mathcal{L}/\partial\mathcal{S})(\partial\mathcal{S}/\partial\mathcal{Z})$ which we use to update the latent codes for our synthetic data. For example, with 128×128-resolution StyleGAN-XL, this memory-saving trick allows us to save nearly 2GB memory per synthetic image with F0 space.

4. Experiments

We evaluate our method (`GLaD`) for distilling CIFAR-10 [40] and 10-class subsets of ImageNet 1k [20], utilizing StyleGAN-XL [51] generators trained on these datasets (obtained from the official released model checkpoints).

The code for our experiments is based on the open-source repositories for `DC`, `DM`, and `MTT` and will be released upon publication. For each method, we integrate the deep generative prior directly into the existing code base. For results with and without the generative prior, we use the same set of hyper-parameters (N, M, T^+, #iterations, etc.) to ensure a fair comparison. For each method, we choose an Fn space and use it for all datasets. Please see the supplemental material for experiment details.

Datasets. For low-resolution data, we apply our method to CIFAR-10 [40]. For higher-resolution data, we use *subsets* of ImageNet-1k [20]. Previous dataset distillation work [13] introduced several subsets based largely on categories and aesthetics, including birds, fruits, and cats. Two other conventional subsets are ImageNette and ImageWoof [25]. We also introduce several new 10-class subsets based on the evaluation performance of a ResNet-50 model that has been pre-trained on ImageNet. In this work, "ImageNet-A" consists of the top-10 classes with "ImageNet-B" consisting of the next 10 and so on for "ImageNet-C," "ImageNet-D," and "ImageNet-E." The classes composing all the subsets can be found in the supplemental material.

Evaluation Protocol. After distilling our synthetic datasets with their respective algorithm, we then evaluate them on a set of unseen architectures. To evaluate a synthetic dataset on a given architecture, we train a network from scratch on

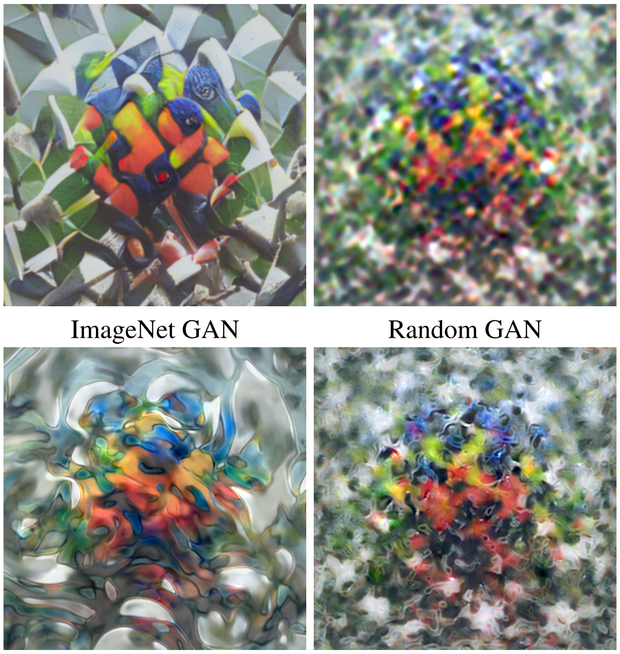


Figure 7. Distilling into the latent space of generators that have **not** been trained on the relevant data still produces good images. Here we show a "lorikeet" distilled using a randomly initialized generator as well as generators trained on ImageNet, Pokémon, and FFHQ.

the distilled dataset and then evaluate it on the validation set.

The training regiment is the same for all networks and datasets: SGD with momentum, ℓ_2 weight decay, and 500 epochs of linear warm-up followed by another 500 of cosine decay. An appropriate (fixed) starting learning rate is used for each architecture, and the final validation set evaluation is done using the exponential moving average of the model's weights. This process is repeated 5 times, and the mean validation accuracy $\pm$ 1 standard deviation is reported. Further details can be found in the supplemental material.

Network Architectures. As with prior dataset distillation works [19, 44, 45, 60, 61, 64, 66–68], we use the ConvNet [27] architecture as our backbone network. A Depth-n ConvNet consists of n blocks followed by a fully-connected layer where each block consists of a 3×3 convolutional layer with 128 filters, instance normalization [58], ReLU non-linearity, and 2×2 average pooling with stride 2.

We used two sets of models for our cross-architecture generalization experiments. For our CIFAR experiments, we use the AlexNet [41], VGG-11 [52], and ResNet-18 [30] included with the codebase of `DC`/`DSA`/`DM`/`MTT` along with a Vision Transformer [24] from the DC-BENCH [18] repository. For our experiments on higher-resolution data, we use slightly modified versions of the networks to accommodate the larger images.

Our experiments are enumerated below to highlight the contributions of our proposed method.

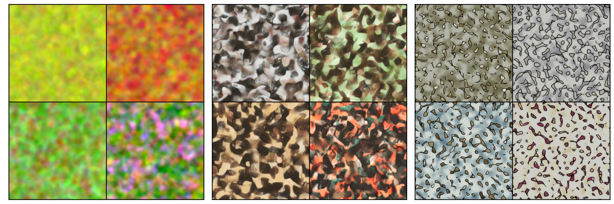


Figure 8. When initializing the latents with random noise, each generator has a unique type of structure present in the output. The final optimized images (Figure 7) contain artifacts of this structure.

4.1. Finding a Suitable Latent Space

Given the depth of StyleGAN-XL [51], there are *many* possible latent spaces that `GLaD` can use to parameterize our synthetic dataset. To find the best latent space for cross-architecture generalization, we experiment with several ImageNet subsets. In Figure 3, we see the final distilled images for the "flamingo" class. Earlier latent spaces enforce a stronger prior on the distilled image, while later latent spaces offer more flexibility for the optimization to fit to the distillation objective (`MTT`, `DC`, or `DM`). Examining Figure 3, we see that the images distilled into F12 space result in the best cross-architecture generalization for `MTT`. Through analogous experiments, we found F16 to be optimal for `DC` and F20 for `DM`.

4.2. Improving Cross-Architecture Generalization

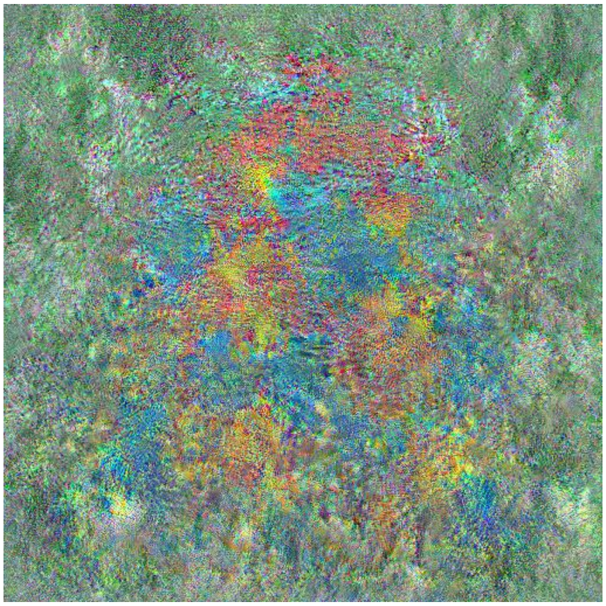
Arguably the most lacking point of all previous dataset distillation methods, cross-architecture generalization gives a good understanding of how well the distillation method "understands" the classification task rather than simply overfitting to a given architecture. In Table 1, we show cross-architecture results for `MTT`, `DC`, and `DM` with and without the deep generative prior. For each method and dataset, a 1 image-per-class synthetic set is distilled using a Depth-5 ConvNet as the "backbone" architecture. To evaluate cross-architecture generalization, we use the distilled set to train AlexNet [41], ResNet-18 [30], VGG-11 [52], and ViT-b/16 [24] from scratch and record the validation accuracy. We record the average validation accuracy across these 4 architectures in Table 1.

For every tested dataset, `GLaD`'s addition of the generative prior slightly or significantly improved the cross-architecture generalization of all 3 methods.

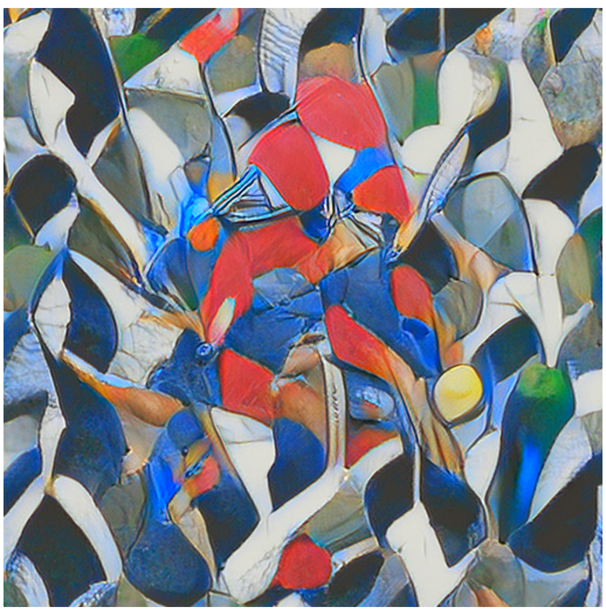
In Table 3, we also include cross-architecture results for CIFAR-10. Even on this lower-resolution data, `GLaD` significantly improves the performance of both original `MTT` and original `DM` while only showing marginal gains on `DC`.

4.3. Latent Initialization and Generator Choices

For all previous experiments, we initialized our latents z (which contain output spaces of intermediate layers) via



Pixel-Space



GLaD

Figure 9. Generative Latent Distillation allows us to distill datasets into high-resolution artistic images (**bottom**), while high-resolution images distilled into pixel space degenerate into high-frequency patterns (**top**). This example is a 512×512 "macaw" distilled using MTT in pixel space versus GLaD.

a partial feed-forward pass through the generator. Here, we experiment with an alternative initialization wherein the feed-forward initialization is replaced with Gaussian noise (with matching mean and variance). As the StyleGAN generator is expecting a somewhat coherent representation in the latent space, the images at initialization have interesting artifacts that vary in granularity depending on the latent space used (Figure 6).

Using such Gaussian initialization, we found that GLaD can successfully use generators that were trained on *completely* different datasets (such as FFQH [37] and Pokémon [1]) or even generators that have not been trained at all (a-la Deep Image Prior [59]). In Table 4, we compare the pixel-space and standard GAN results to generative latent distillations using generators trained on the FFHQ and Pokémon along with generators that have not been trained at all. These results use DC, and results using MTT and DM can be found in the supplemental material.

The images distilled using the FFHQ, Pokémon, and random generators still offer cross-architectural generalization improvements over those distilled directly into pixel space, ofter matching or surpassing the results using the standard ImageNet generator.

We also notice that initializing the latents with random Gaussian noise results in aesthetically pleasing images with different "artistic" properties based on the generator used (Figure 7). This trend also extends to even larger images, allowing our method to create class-based digital arts in different styles, such as the "mosaic" in Figure 9.

5. Discussion and Limitations

In this work, we have proposed leveraging a generative prior to dataset distillation (GLaD). By applying our deep generative prior, we introduce a new method that significantly improves the generalization of the distilled images. This trend extends all the way to (and likely beyond) 512×512 images, allowing us to generate high-quality distilled images at higher resolutions than ever before. Since GLaD acts as a plug-and-play addition to any dataset distillation method, future works can use it to increase the generality of their data and generate higher-resolution images.

Limitations. Introducing StyleGAN-XL to the distillation pipeline creates a massive new memory sink. Our checkpointing trick allows us to mitigate this issue to some extent. However, this also comes at the expense of a second forward pass through the generator. Given that a single pass through the generator is time-consuming, a second pass doubles the overhead. Additionally, a large enough synthetic set requires passes through the generator to be done in multiple batches, further increasing the extra time needed.

Fortunately, GLaD is compatible with any differentiable generative model, so the development of more efficient generative models in the future will naturally reduce the cost of our method as well.

Acknowledgment. This work was partly done by George Cazenavette during his study at CMU. This work is supported, in part, by y the NSF Graduate Research Fellowship and J.P. Morgan Chase Faculty Research Award.

References

[1] Pokémon. pokemon.com. 8

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *ICCV*, 2019. 2, 3, 5

[3] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *CVPR*, 2020. 3

[4] Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *NeurIPS*, 2021. 5

[5] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *ECCV*. Springer, 2020. 3

[6] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM TOG*, 2019. 3

[7] Cenk Baykal, Murad Tukan, Dan Feldman, and Daniela Rus. Small coresets to represent large training data for support vector machines. 2018. 2

[8] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *NeurIPS Workshop*, 2020. 3, 4

[9] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[10] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *NeurIPS*, 2020. 2

[11] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 3

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 1

[13] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6

[14] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Wearable imagenet: Synthesizing tileable textures via dataset distillation. In *CVPR Workshops*, 2022. 1, 3

[15] Lucy Chai, Jonas Wulff, and Phillip Isola. Using latent space regression to analyze and leverage compositionality in gans. In *ICLR*, 2021. 3

[16] Lucy Chai, Jun-Yan Zhu, Eli Shechtman, Phillip Isola, and Richard Zhang. Ensembling with deep generative views. In *CVPR*, 2021. 3

[17] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 6

[18] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Dc-bench: Dataset condensation benchmark. In *NeurIPS Datasets and Benchmarks Track*. 7

[19] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory. *arXiv preprint arXiv:2211.10586*, 2022. 4, 7

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[21] Zhiwei Deng and Olga Russakovsky. Remember the past: Distilling datasets into addressable memories for neural networks. In *NeurIPS*. 3

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019. 1

[23] Tian Dong, Bo Zhao, and Lingjuan Liu. Privacy for free: How does dataset condensation help privacy? In *ICML*, 2022. 1

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 7

[25] Fastai. Fastai/imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french. 6

[26] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*, 2019. 1

[27] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018. 7

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014. 2

[29] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 2007. 2

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[31] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2015. 1

[32] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016. 3

[33] Chengeng Huang and Sihai Zhang. Generative dataset distillation. In *2021 7th International Conference on Big Data Computing and Communications (BigCom)*. IEEE, 2021. 3

[34] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. In *ECCV*, 2020. 3

[35] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020. 3, 5

[36] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 5

[37] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3, 8

[38] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 3

[39] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun,

Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. *ICML*, 2022. 3

[40] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 7

[42] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. *NeurIPS*, 2022. 3

[43] Wojciech Masarczyk and Ivona Tautkute. Reducing catastrophic forgetting with learning on synthetic data. In *CVPR Workshops*, 2020. 1

[44] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *ICLR*, 2020. 3, 4, 7

[45] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. In *NeurIPS*, 2021. 3, 4, 7

[46] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. *PAMI*, 2021. 3

[47] Gaurav Parmar, Yijun Li, Jingwan Lu, Richard Zhang, Jun-Yan Zhu, and Krishna Kumar Singh. Spatially-adaptive multilayer selection for gan inversion and editing. In *CVPR*, 2022. 2, 5

[48] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1

[49] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM TOG*, 2022. 3

[50] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. volume abs/2201.00273, 2022. 3

[51] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022. 5, 6, 7

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7

[53] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *ICML*, 2020. 1

[54] Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *IJCNN*, 2021. 3

[55] Ayush Tewari, Mohamed Elgharib, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control. *ACM TOG*, 2020. 3

[56] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. In *SIGGRAPH*, 2021. 3

[57] Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *JMLR*, 2005. 2

[58] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 7

[59] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018. 8

[60] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. *CVPR*, 2022. 1, 3, 4, 7

[61] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018. 1, 2, 3, 4, 7

[62] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Weiming Zhang, Lu Yuan, Gang Hua, and Nenghai Yu. A simple baseline for stylegan inversion. *arXiv preprint arXiv:2104.07661*, 2021. 3

[63] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *CVPR*, 2021. 3

[64] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *ICML*, 2021. 1, 2, 3, 4, 7

[65] Bo Zhao and Hakan Bilen. Synthesizing informative training samples with gan. *NeurIPS Workshop*, 2022. 3

[66] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *WACV*, 2023. 1, 2, 5, 6, 7

[67] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2020. 2, 3, 4, 5, 6

[68] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *NeurIPS*, 2022. 1, 7

[69] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*. Springer, 2020. 3

[70] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 3

[71] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Barbershop: Gan-based image compositing using segmentation masks. *ACM TOG*, 2021. 2, 3, 5

[72] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020. 3