Generalizing Dataset Distillation via Deep Generative Prior Supplemental Material

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

A. More Visualizations

Please see our web page for more visualizations: georgecazenavette.github.io/glad.

B. StyleGAN Latent Spaces

Here we further elaborate on our embedding spaces for those unfamiliar with the StyleGAN architecture. Some details will be left out for easier digestion. Visualizations of these embedding spaces can be seen in Figure 1.

For standard image generation with StyleGAN 3 [4] (and other StyleGAN models), a random vector is first sampled from the multi-variate standard normal distribution: $z \sim \mathcal{N}(0, I)$. This random Gaussian vector is then fed through a "mapping" network (typically a simple MLP) to obtain a "style code" W. In a class-conditional StyleGAN, this "mapping" network is the *only* place where the class information is used. This W is then passed to every "style block" of the "synthesis" network as used to modulate the convolutional layers of each block. The "synthesis" network takes in a learned constant as input and uses the modulated convolutions of the style blocks to generate the realistic image.

For each distilled sample in W^+ space, we optimize a *different* W code for each style block and use the synthesis network to generate our synthetic data. The "mapping" network is not used aside from initializing W. In Fn space, we *directly* optimize the feed-forward input to the n^{th} style block as well as the W codes for all subsequent style blocks. Any preceding style blocks and W codes are simply ignored.

C. Dataset Specifications

Our high-resolution data is taken directly from the ImageNet 1k dataset [3] using PyTorch's built-in ImageNet loader [5]. To train our expert trajectories, we use data from the ImageNet training set. To compile *our* training set for the expert trajectories, we select the classes from the given subset, resize the short side of the image to the given resolution, and take a center crop according to the given resolution (as done by `MTT` [1]). The validation set is obtained in the same way from the ImageNet validation set.

For an enumeration of which ImageNet classes are in each of our datasets, please see Table 1.

D. More Experimental Values

In Table 2, we show the performance of the distilled images on the backbone architecture (the architecture used for distillation). We note that we do not expect `GLaD` to perform better than the baseline pixel-based distillation since `GLaD` is designed to *reduce* overfitting to the distilling architecture. Despite this, `DC` and `DM` with `GLaD` perform as well or better than the pixel-based versions (with `MTT` performing somewhat worse).

In Table 3, we show results using 10 distilled images per class. `GLaD` still tends to perform better than pixel space distillation.

In Table 4, we show the baseline results of training each architecture on the whole dataset. We did not spend time tuning the hyper-parameters here, perhaps explaining why ConvNet tends to have the best performance.

E. Hyper-Parameters and Experimental Details

For the experiments on `MTT` and our new method, we base our experiments on the open-source code for `DC`+`DM` (link) [7, 8], `MTT` (link) [1], and `TESLA` (link) [2].

To optimize the distilled images/latents and learnable synthetic step-size (α), we use the same optimizer and hyper-parameters as the original methods. For the W^+ latents, divide the learning rate by 10.

For our `MTT` experiments, we set the number of synthetic steps per iteration (N) as 10, the number of real epochs to match (M) as 2, and the maximum starting epoch (T^+) set to 2. All experiments on `MTT` and our new method are run for 5k iterations and then evaluated via the protocol described in the body of the paper.

All 32×32, 128×128, 256×256, and 512×512 experiments are distilled using ConvNetD3, ConvNetD5, Con-

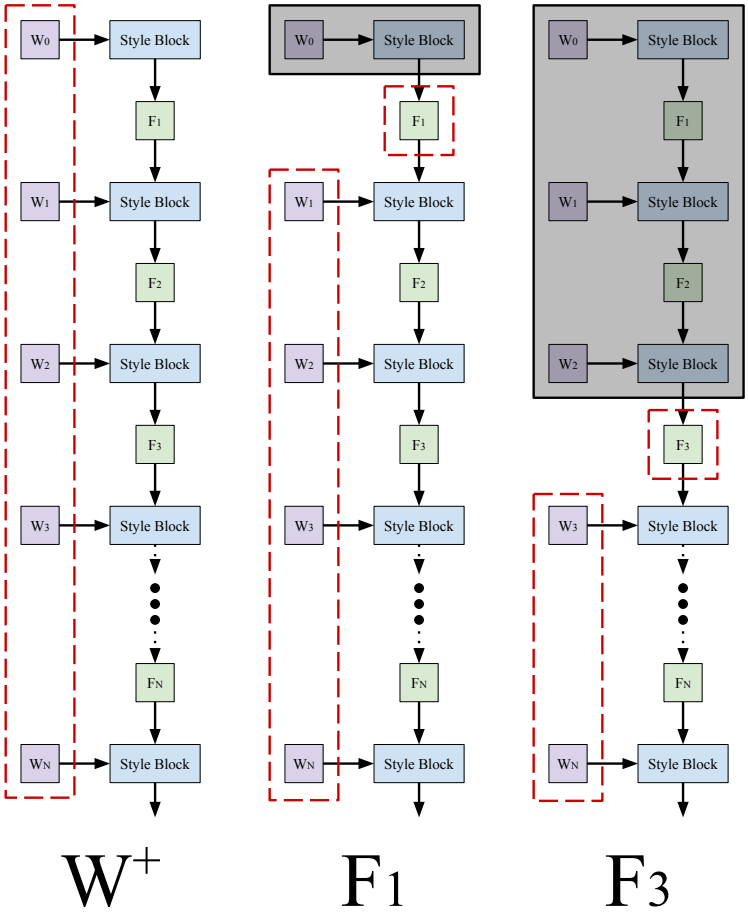

Figure 1. Different optimization spaces of StyleGAN. Latent variables boxed in red are directly optimized while those that are grayed out are not used at all. Note: the "mapping" network is omitted here since we do not use it in any of our optimization spaces.

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

Table 1. Class listings for our ImageNet subsets. Visualizations show classes in the same order given here.

Distil. Alg.	Distil. Space	ImNet-A	ImNet-B	ImNet-C	ImNet-D	ImNet-E	ImNette	ImWoof	ImNet-Birds	ImNet-Fruits	ImNet-Cats
MTT [1]	Pixel	51.7±0.2	53.3±1.0	48.0±0.7	43.0±0.6	39.5±0.9	41.8±0.6	22.6±0.6	37.3±0.8	22.4±1.1	26.6±0.4
	GLaD (Ours)	50.7±0.4	51.9±1.3	44.9±0.4	39.9±1.7	37.6±0.7	38.7±1.6	23.4±1.1	35.8±1.4	23.1±0.4	26.0±1.1
DC [8]	Pixel	43.2±0.6	47.2±0.7	41.3±0.7	34.3±1.5	34.9±1.5	34.2±1.7	22.5±1.0	32.0±1.5	21.0±0.9	22.0±0.6
	GLaD (Ours)	44.1±2.4	49.2±1.1	42.0±0.6	35.6±0.9	35.8±0.9	35.4±1.2	22.3±1.1	33.8±0.9	20.7±1.1	22.6±0.8
DM [7]	Pixel	39.4±1.8	40.9±1.7	39.0±1.3	30.8±0.9	27.0±0.8	30.4±2.7	20.7±1.0	26.6±2.6	20.4±1.9	20.1±1.2
	GLaD (Ours)	41.0±1.5	42.9±1.9	39.4±0.7	33.2±1.4	30.3±1.3	32.2±1.7	21.2±1.5	27.6±1.9	21.8±1.8	22.3±1.6

Table 2. Performance on ConvNet (architecture used to distill).

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 3. Performance with 10 images/class.

Arch.	ImNet-A	ImNet-B	ImNet-C	ImNet-D	ImNet-E
ConvNet	90.6±0.6	92.3±0.2	84.2±0.3	74.5±1.0	76.2±0.6
ResNet18	78.8±1.6	80.2±1.1	69.2±1.6	51.0±0.7	53.2±2.8
VGG11	78.4±1.1	81.4±1.5	74.6±1.2	67.3±1.6	67.8±1.3
AlexNet	81.0±0.3	76.5±1.4	72.2±1.1	65.4±1.1	63.5±1.1
ViT	77.5±0.4	76.4±0.4	75.5±1.4	58.6±0.9	59.5±1.2

Table 4. Training networks from scratch on the *whole* dataset.

vNetD6, and ConvNetD7 respectively as the backbone.

The same suite of differentiable augmentations (originally from the DSA codebase [6]) is used for all experiments: color, crop, cutout, flip, scale, and rotate with the default parameters.

To obtain the expert trajectories used by MTT, we train a model from scratch on the real dataset for 15 epochs of SGD with a learning rate of 10^{-2}, a batch size of 256, and NO momentum or regularization.

Our experiments were run on a combination of RTX2080ti, RTX3090, RTX6000, RTXA5000, and RTXA6000 GPUs depending on availability.

References

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

[2] 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. 1

[3] 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. 1

[4] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 1

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[6] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *ICML*, 2021. 3

[7] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *WACV*, 2023. 1, 3

[8] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2020. 1, 3