Backprop-Free Dataset Distillation
–Supplementary Materials–

Anonymous Author(s)
Affiliation
Address
`email`

In this document, we include more details about the technical methods, more experimental results of our backprop-free dataset distillation method, and more discussion on limitations and future works, which cannot be accommodated in the main paper due to the page limit. Our method first trains a meta generator to generate synthetic samples and then an adaptation stage is executed for a target dataset. We provide algorithmic details of the adaptation stage, a summary of hyper-parameters, and configurations of our generator architecture. Then, we conduct more evaluations on the cross-number-of-channel, cross-resolution, cross-ipc, and cross-number-of-classes performance of our method. More discussions of the adaptation performance and more qualitative examples are also included. Finally, we discuss limitations of the proposed method and potential future works.

A More Details

Adaptation Algorithm: Alg. 1 of the main paper demonstrates the procedure of meta learning to obtain a meta synthetic sample generator. On a downstream target dataset, the meta network is adapted to a specific network with a limited number of steps. The adaptation algorithm is similar to the meta-training step of the meta learning algorithm. Here, we present the full details in Alg. 1. We can prepare multiple initialization of synthetic samples through randomly sampling from the target dataset. Recall that the main pipeline of our algorithm is to first obtain analytical synthetic labels in a random neural space θ: $Y_s^* = f_\theta(X_s)W_t^\theta$. Here, the optimal kernel-ridge-regression parameters of the target dataset W_t^θ can be computed by $W_t^\theta = f_\theta(X_t)^\top(f_\theta(X_t)f_\theta(X_t)^\top)^{-1}Y_t$, if the number of real samples n_t is smaller than the feature dimension p. Otherwise, $W_t^\theta = (f_\theta(X_t)^\top f_\theta(X_t))^{-1}f_\theta(X_t)^\top Y_t$.

Algorithm 1 Adaptation Algorithm of Synthetic Sample Generator for a Target Dataset

Input: (X_t, Y_t): A Target Dataset; T: Number of Adaptation Steps; α: Learning Rate of Generator; θ: Parameter of a Random Neural Network; ω: Parameter of a Meta Generator; $\mathcal{I}$: A Set of Randomly Initialized Synthetic Samples.

Output: ω': Parameter of a Target-Specific Generator.

```
1:  W_t^θ = f_θ(X_t)^⊤ (f_θ(X_t) f_θ(X_t)^⊤)^{-1} Y_t;
2:  for Each X_s in I do
3:      Y_s^* = f_θ(X_s) W_t^θ;                                   ▷ Eq. 3 in the main paper
4:  end for
5:  Initialize generator parameters ω' with ω;
6:  for 1 ≤ i ≤ T do
7:      Sample a batch of real data (X_t^i, Y_t^i) of from (X_t, Y_t);
8:      Sample a initialized synthetic data (X_s, Y_s^*) from I;
9:      X_s^* = g_{ω'}(X_s);                                      ▷ Forward propagation
10:     Sample neural parameters θ^* from a random distribution;
11:     L = ‖f_{θ*}(X_t) f_{θ*}(X_s^*)^⊤ (f_{θ*}(X_s^*) f_{θ*}(X_s^*)^⊤)^{-1} Y_s^* − Y_t‖_2^2;   ▷ Eq. 1 in the main paper
12:     Update ω' via ω' ← ω' − α∇_{ω'} L;                        ▷ Back propagation
13: end for
```

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

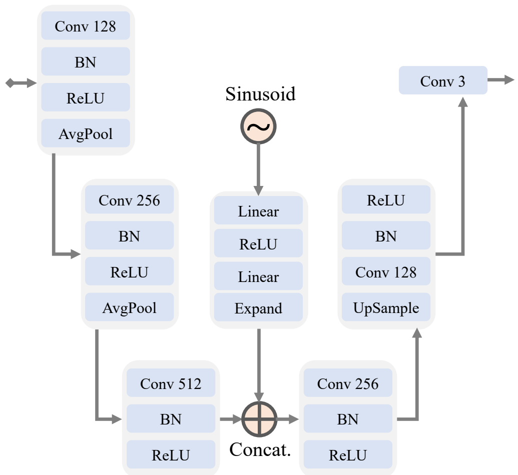


Figure 1: Architecture of our generator network.

Hyper-Parameter	Notation	Value
Meta Learning Stage		
Number of Meta Testing Steps	T'	200,000
Number of Meta Training Steps	T	5
Maximal Number of Classes	$\max(C)$	100
Minimal Number of Classes	$\min(C)$	10
Maximal Number of Synthetic Samples	$\max(n_s)$	1,000
Minimal Number of Synthetic Samples	$\min(n_s)$	10
Number of Real Samples	n_t	2,000
Learning Rate in Meta-Training	α	1e-4
Learning Rate in Meta-Testing	β	1e-5
Parameter of Adam Optimizer	(β_1, β_2)	$(0.9, 0.999)$
Parameter of Cosine Learning Rate Scheduler	η	0.1
Adaptation Stage		
Number of Adaptation Steps	T	1,000
Batch Size of Real Data	n_t	1,024
Learning Rate of Generator	α	1e-4
Parameter of Adam Optimizer	(β_1, β_2)	$(0.9, 0.999)$
Parameter of Cosine Learning Rate Scheduler	η	0.1

Table 1: List of hyper-parameters.

	Dataset	MNIST			FashionMNIST		
	IPC	1	10	50	1	10	50
	Ratio (%)	0.017	0.17	0.83	0.017	0.17	0.83
Random	Acc. (%)	64.9±3.5	95.1±0.9	97.9±0.2	51.4±3.8	73.8±0.7	82.5±0.7
Full	Acc. (%)		99.6±0.0			93.5±0.1	
DC [12]	Acc. (%)	91.7±0.5	97.4±0.2	98.8±0.2	70.3±0.7	83.4±0.3	82.9±0.2
	Time (sec.)	157	3581	19811	155	3597	19829
DSA [10]	Acc. (%)	88.7±0.6	98.8±0.2	99.2±0.1	70.3±0.7	84.6±0.1	88.7±0.1
	Time (sec.)	172	3908	21259	173	3854	21118
IDC [4]	Acc. (%)	89.1±0.1	97.8±0.1	98.8±0.1	70.6±0.4	85.2±0.4	88.9±0.1
	Time (sec.)	22062	22798	28389	21929	23160	28499
MTT [1]	Acc. (%)	88.7±1.0	96.6±0.4	98.1±0.1	75.3±0.9	87.2±0.3	88.3±0.1
	Time (sec.)	3114	9323	9987	3107	9305	10092
DM [11]	Acc. (%)	89.7±0.6	97.5±0.1	98.6±0.1	71.5±0.5	83.8±0.2	88.2±0.3
	Time (sec.)	1115	1177	1457	1105	1172	1456
FRePo [13]	Acc. (%)	93.0±0.4	98.6±0.1	99.2±0.0	75.4±0.5	85.5±0.2	89.2±0.1
	Time (sec.)	6112	9174	21678	6115	8463	21549
Ours	Acc. (%)	91.3±0.2	97.8±0.2	99.0±0.0	73.8±0.8	84.7±0.2	88.3±0.1
	Time (sec.)	**153**$_{\times 40}$	**392**$_{\times 10}$	**1012**$_{\times 21}$	**147**$_{\times 42}$	**432**$_{\times 22}$	**1005**$_{\times 21}$

Table 2: Comparisons on test accuracy and running time with state of the arts on single-channel datasets. The acceleration marked by the red subscript is computed against the method with the best accuracy. IPC: Number of Images Per Class; Ratio: ratio of distilled images to the whole training set. Results demonstrate the cross-channel generalization ability of our meta generator.

After the calculation of analytical labels, we fix them and train the synthetic sample generator initialized by parameters of the meta generator for some steps. The optimization objective is similar to those in Zhou *et al.* [13] and Loo *et al.* [6]. The difference is that the optimization target is parameters of the generator instead of synthetic samples.

Summary of Hyper-Parameters: For a clear view, we summarize the hyper-parameters and their values in both meta learning and adaptation stages as shown in Tab. 1. All experiments follow these default settings of hyper-parameters if not specified. Other configurations unmentioned follow the settings of the baseline FRePo [13].

Generator Architecture: We illustrate the detailed configurations of our generator architecture in Fig. 1. It essentially adopts an encoder-decoder structure with 3 `Conv-BatchNorm-ReLU` blocks and 2 `AvgPool` layers for down-sampling for the encoder and a symmetric structure for the decoder. Notably, to make the network aware of different sizes of synthetic datasets, we concatenate the size embedding to bottle-necked features after the encoder. Inspired by the positional embedding in Transformer models [8] and the time-step embedding in diffusion models [3, 7], we encode the size

IPC	1	10
Baseline	28.20±0.77	48.26±1.26
Ours	**36.51±0.47**	**49.20±0.10**

Table 3: Comparisons with the baseline FRePo on ImageNette under 128 resolution. Results demonstrate the cross-resolution generalization ability of our meta generator.

# of Classes	20		50	
IPC	1	10	1	10
Baseline	23.42±1.08	49.40±0.53	16.84±0.30	39.61±0.21
Ours	37.95±0.44	53.62±0.09	29.53±0.20	41.90±0.38

Table 4: Comparisons with the baseline FRePo on various CIFAR100 subsets. Results demonstrate the cross-number-of-classes generatlization ability of our meta generator.

Dataset	CIFAR10		CIFAR100	
IPC	20	5	5	2
DC	41.8±0.6	25.9±0.4	13.3±0.3	6.7±0.2
DSA	41.5±0.4	27.6±0.2	14.9±0.3	8.1±0.1
IDC	51.9±0.5	30.2±0.4	13.3±0.3	11.0±0.1
MTT	55.9±0.3	29.8±0.4	26.7±0.5	13.7±0.3
DM	46.8±0.5	25.3±0.3	15.7±0.3	8.0±0.2
FRePo	59.1±0.7	38.3±0.9	30.0±0.6	19.9±0.3
Ours	**60.8±0.4**	**46.1±0.8**	**30.6±0.3**	**25.4±0.5**

Table 5: Comparisons with state of the arts on cross-IPC generalization.

by sinusoidal signals and a learnable non-linear transformation function. Embedding features are replicated and expanded along the spatial axes before concatenation with features from the encoder.

B More Results

Cross-Number-of-Channel Generalization: In the meta learning stage, a meta generator is trained taking RGB images as input and output. Here, we demonstrate that it is also be feasible for the meta generator to be adapted for target datasets that have different numbers of channels. Specifically, we additionally train convolution layers for channel adaptation to map the number of channels from the original number to 3 and from 3 to original number at the beginning and the ending positions of the generator, respectively. The parameters of these adaptors are initialized from a uniform distribution and are optimized jointly with parameters of the generator.

Here, we conduct experiments on MNIST [5] and FashionMNIST [9] datasets. Both of them contain 10 classes with 60,000 single-channel images. Results are shown in Tab. 2 following the same comparison protocols as Tab. 1 of the main paper, where the generator in our method is adapted for 10,000 steps in each setting. Experiments demonstrate that our method can achieve comparable performance with those state-of-the-art ones in a significantly shorter period of time. The conclusion is the same as that in the main paper.

Cross-Resolution Generalization: Although the meta generator is trained under 32 resolution, it is possible for it to be adapted for datasets with different resolutions, thanks to the fully-convolutional architecture of the generator. We demonstrate the cross-resolution generalization performance on ImageNette [2], which contains 10 classes and 9,469 images. Following the FRePo baseline [13], we conduct experiments on 1 and 10 IPCs under 128 resolution. Results in Tab. 3 demonstrate the feasibility of such cross-resolution generalization.

Cross-Number-of-Class Generalization: Here, we conduct experiments on CIFAR100 subsets with random 20 and 50 classes respectively and compare the performance with the FRePo baseline [13]. Results in Tab. 4 demonstrate that the meta generator performs robustly on datasets with various numbers of classes.

Cross-IPC Generalization: For existing methods, when budgets for synthetic datasets change, they have to either repeat the time-consuming training loop of dataset distillation, which is inconvenient if not infeasible at all, or prune some synthetic data heuristically, which leads to inferior performance. For example, as shown in Tab. 5, on CIFAR10, if the original synthetic IPC is 50 and the new IPC becomes 20 or 5, random pruning would lead to unsatisfactory performance for existing methods. By contrast, the generator in our backprop-free dataset distillation can work for arbitrary sizes of

Dataset	IPC	DC [12]	DSA [10]	IDC [4]	MTT [1]	DM [11]	FRePo [13]	Ours
MNIST	1	**88.7±0.5**	87.7±0.6	76.1±0.1	73.1±0.8	87.8±0.7	64.8±0.9	87.8±0.2
	10	96.2±0.2	96.7±0.1	95.1±0.1	92.8±0.2	96.2±0.1	96.3±0.1	**97.2±0.1**
	50	95.7±0.2	98.3±0.1	98.4±0.1	96.6±0.1	98.0±0.1	98.5±0.1	**98.6±0.1**
FashionMNIST	1	70.3±0.7	70.3±0.7	64.4±0.4	70.5±1.2	71.1±0.3	61.5±0.3	**71.9±0.4**
	10	79.8±0.2	79.0±0.3	82.9±0.2	80.1±0.5	83.0±0.1	81.2±0.2	**83.4±0.2**
	50	78.5±0.2	86.9±0.1	87.0±0.1	86.2±0.1	86.8±0.2	85.9±0.1	**87.2±0.1**
CIFAR10	1	28.2±0.7	28.1±0.7	25.3±1.0	36.8±0.5	26.8±0.8	27.2±0.5	**42.6±0.3**
	10	39.7±0.5	48.7±0.3	49.5±0.3	50.8±0.5	48.8±0.2	49.4±0.3	**58.9±0.4**
	50	39.1±1.0	56.0±0.4	61.7±0.2	56.5±0.5	57.7±0.3	61.8±0.2	**66.8±0.2**
CIFAR100	1	12.4±0.2	13.8±0.2	15.4±0.2	13.2±0.6	11.9±0.2	10.1±0.2	**20.8±0.2**
	10	21.1±0.2	31.3±0.4	28.9±0.3	30.2±0.4	30.0±0.4	26.6±0.4	**32.2±0.3**

Table 6: Comparisons with state of the arts on various benchmarks under the same number of training steps. IPC: Number of Images Per Class. Results demonstrate the superior efficiency of our method.

synthetic datasets once adapted, which makes it handle such scenarios better. We present another example on CIFAR100, the original IPC is 10 and the new IPC is 5 or 2.

Comparisons under the Same Steps: To better demonstrate the superiority of the proposed method, we compare our method with state of the arts with the number of training/adaptation steps controlled the same. As shown in Tab. 6, under 1000 steps, our method outperforms others significantly especially on relatively challenging datasets with more patterns, like CIFAR10 and CIFAR100. Furthermore, in Fig. 2, 3, 4, and 5, we visualize the performance of generators in each setting with different adaptation steps on MNIST, FashionMNIST, CIFAR10, and CIFAR100 datasets respectively as supplements to Fig. 4 in the main paper. It can be shown that our method can achieve the most satisfactory performance with only a limited number of adaptation steps compared with the baseline FRePo and generators from scratch, which indicates that the proposed method is more suitable for scenarios requiring high efficiency, like processing data streams. Note that for 1 IPC, we observe that using analytical labels would often lead to inferior performance compared with vanilla one-hot labels. We speculate that it is because soft labels by the analytical solution are relatively not good at leading the generator to synthesize class-discriminative patterns when the size of synthetic dataset is small. Thus, we do not use analytical labels for 1 IPC by default.

Qualitative Results: In Fig. 6, we supply qualitative visualization of initialized synthetic samples and results by generator under 1 and 10 IPC on CIFAR10 and 1 IPC on CIFAR100, as supplements to Fig. 6 in the main paper.

C Limitations and Future Works

Our backprop-free dataset distillation method mainly focuses on the efficiency issue in existing methods. Although it can be demonstrated that our method can result in better performance in only limited time, it does not reduce the time and memory complexity of computing the matching metrics since we adopt the same objectives as previous approaches. When adapting for large synthetic datasets, it may still face the issue on GPU memory in existing works. Nevertheless, it is possible for our method to adapt on some small IPCs and then generalize to large synthetic datasets, as discussed in the main paper, which can serve as a remedy to this limitation. Besides, initialized samples of synthetic datasets come from real data, and results by generator still look somehow realistic, which may potentially make the method vulnerable to privacy attack, especially for data like personal information. Also, in scenarios like storing synthetic samples of human faces, the generator may break the integrity of faces and lead to an infringement of portrait rights if being misused.

Future works may focus on more effective training objective, training pipeline, and architecture of the generator in meta learning or/and adaptation stages to further improve the cross-dataset, cross-ipc, and cross-architecture generalization. It would also be valuable to extend the backprop-free DD to other tasks and modalities beyond image classification and explore advanced input and output parameterizations of the generator.

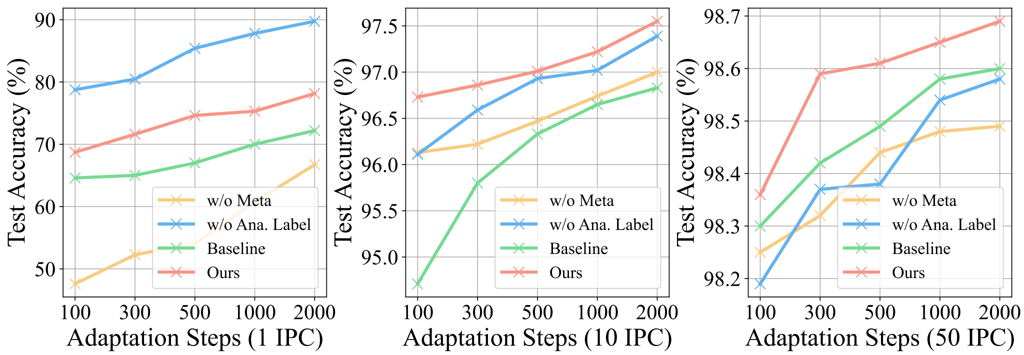


Figure 2: Performance of generators with various adaptation steps on MNIST.

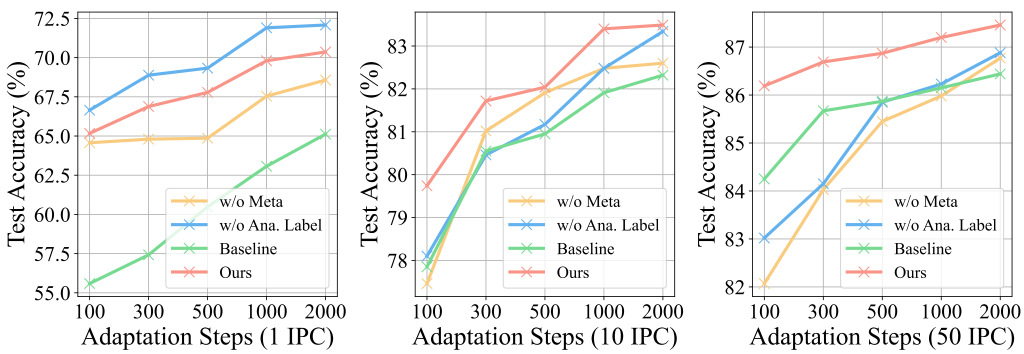


Figure 3: Performance of generators with various adaptation steps on FashionMNIST.

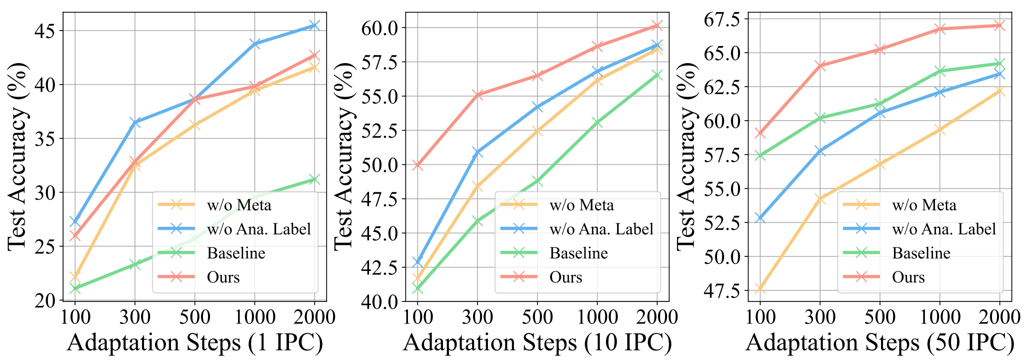


Figure 4: Performance of generators with various adaptation steps on CIFAR10.

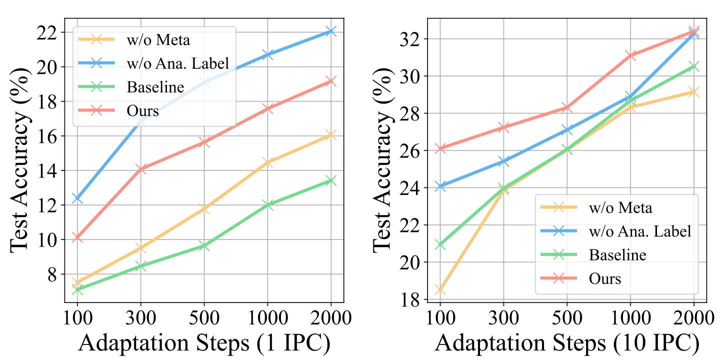
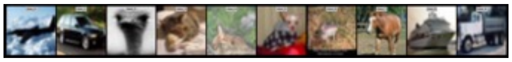


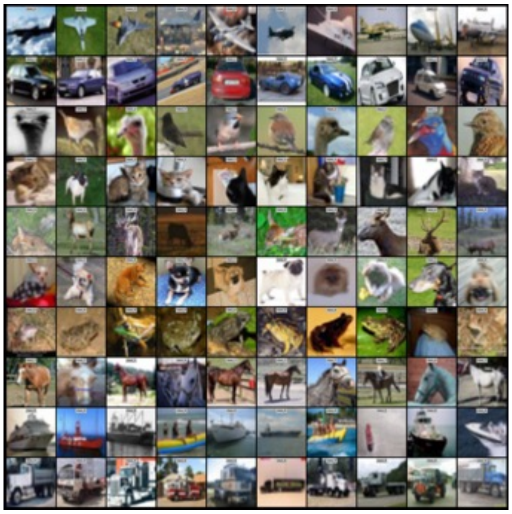
Figure 5: Performance of generators with various adaptation steps on CIFAR100.



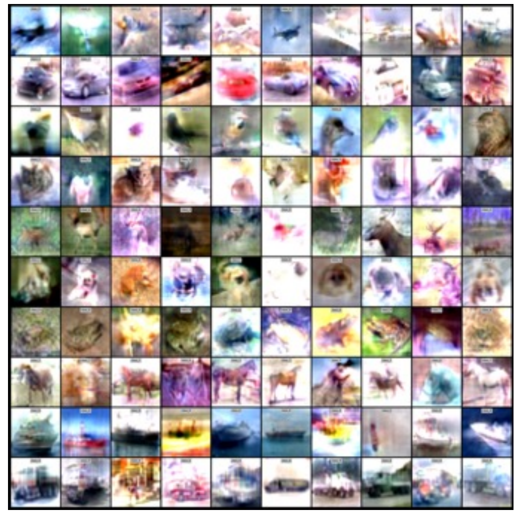
Before Generator

After Generator

1 IPC, CIFAR10

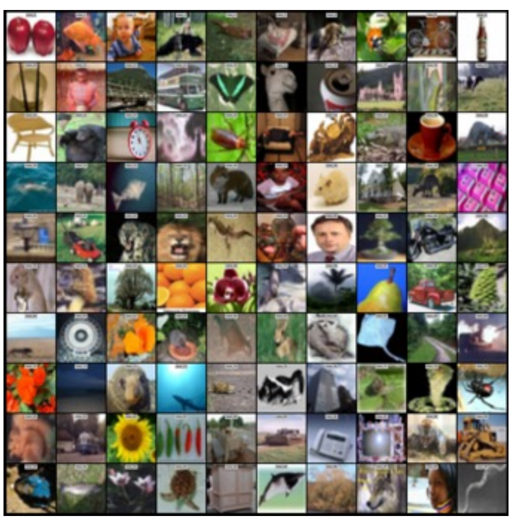


Before Generator

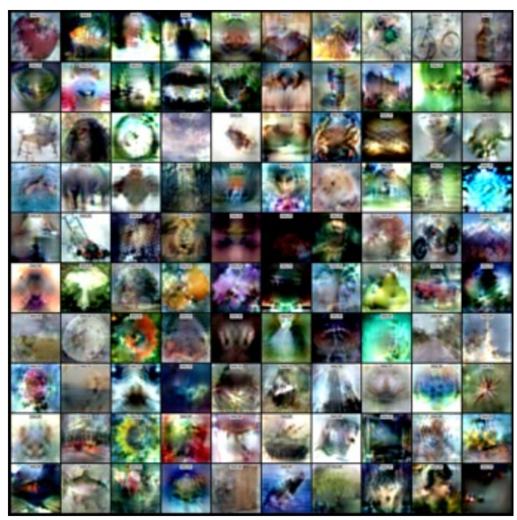


After Generator

10 IPC, CIFAR10



Before Generator



After Generator

1 IPC, CIFAR100

Figure 6: More visualizations of samples before and after generator on CIFAR10 and CIFAR100.

References

[1] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. *arXiv preprint arXiv:2203.11932*, 2022.

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

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[4] 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. *arXiv preprint arXiv:2205.14959*, 2022.

[5] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[6] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[10] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.

[11] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023.

[12] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

[13] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022.