Improved Distribution Matching for Dataset Condensation Supplementary Materials

1. Implementation Details

Random, Herding and Forgetting for ImageNet Subset. Following other methods in Table 1 of the main paper, we use the augmentation strategy of DC [2] in the evaluation of three baselines of ImageNet Subset. For **Random**, we randomly pick samples from the real set according to the experiment setting and evaluate the performance using the same model architecture and hyper-parameters. For **Herding**, we first pretrain the ConvNet using the entire ImageNet Subset for 40 epochs, where the accuracy reaches the peak performance, then use the ConvNet to extract features for all images. Finally, we calculate the average feature of all features from the same class and select its closest neighbors among the features of the real images as the coreset for evaluation. Similar to DM [1], we use the L2 distance for closest neighbor calculation. For **Forgetting**, we first train the ConvNet with the entire ImageNet Subset and then count the number of epoch-wise incorrect predictions for each real sample independently. Finally, we select the samples with the largest numbers of incorrect predictions as the coreset. Since the performance of the model continues to increase as the training progresses, we empirically found that **Forgetting** performs much worse than the other two baselines when the epoch number is set to a large number, possibly because ImageNet contains some mislabeled samples. Therefore, we reduce the epoch number for ConvNet training to 5 and randomly select the samples with the largest numbers of incorrect epoch-wise predictions.

Other Details of Our Method. For all the experiments in Table 1 of the main paper, we use a learning rate of 0.2. Besides, we set an extra interval for the **Push** and **Train** model queue operations to reduce the training cost for condensation. For CIFAR-10/100, we set the interval as 30.

2. Cross-architectural Experiments

As a complement to Table 4, Sec. 5.5 of the main paper, we further compare our method to DM with three other architectures. The full table is shown in Table 1. It can be observed that our method outperforms DM in all 4×4 different settings.

Table 1. Cross-architectural performance of our IDM method on CIFAR-10 with 10 Img/Cls. Our IDM achieves a significant improvement over DM. **Bold**: DM with three other architectures that are not included in the main paper.

	C\T	ConvNet	AlexNet	VGG	ResNet
DM	ConvNet	48.9±0.6	38.8±0.5	42.1±0.4	41.2±1.1
	AlexNet	34.4±0.3	28.8±1.1	31.6±0.6	31.4±0.3
	VGG	31.7±0.7	30.1±1.3	31.9±0.4	30.0±1.0
	ResNet	35.5±0.3	31.3±0.3	32.6±0.7	35.3±0.9
Ours	ConvNet	53.0±0.3	44.6±0.8	47.8±1.1	44.6±0.4
	AlexNet	44.8±0.5	41.4±1.4	43.1±0.6	41.0±0.1
	VGG	41.2±0.4	37.4±0.3	41.7±0.4	38.8±0.8
	ResNet	38.3±0.4	37.0±0.7	39.0±0.1	39.0±0.4

3. Start with Pre-trained Models

As we mentioned in Sec 4.2 of the main paper, our method pushes randomly initialized networks to the model queue, and the model fetched from the model queue is trained for K iterations in each condensation step. However, when the model queue is applied to larger and more difficult datasets, sometimes we need a large K and N_{max} to make the model queue large enough to contain sufficiently trained models, i.e., the model trained for $K \times N_{max}$ iteration is well-performing enough to extract meaningful embeddings. This could be computationally intensive. To alleviate the problem, we propose some simple modifications to the model queue that can enable the model queue to start with pretrained networks, i.e., $P_\theta(t)$ where $t > 0$.

The first modification for the problem is a small pre-trained model set, in which the models are trained for t iteration, and the **Push** operation of the model queue pushes a randomly selected model from the pre-trained model set to the model queue. Besides, to preserve the diversity of the starting models for the model queue, we further apply some changes to the optimization scheme for each selected model. When pushing each pre-trained model to the model queue, we initialize the corresponding optimizer with a smaller random perturbation to the learning rate:

$$lr^* = lr + Random(-0.1 \times lr, 0.1 \times lr), \qquad (1)$$

where lr is the original learning rate, $Random(-0.1 \times lr, 0.1 \times lr)$ generates a random float value between $-0.1 \times$

lr and $0.1 \times lr$, and $lr*$ is the new learning rate for the corresponding optimizer. Moreover, we randomly select a subset of image classes and assign the subset to the selected model. In the training operation of the model queue, we only trained the model with the real image samples from the categories in the subset, thus the optimization of each pre-trained model can be diversified for better condensation. The learning rate perturbation and class subset training can be viewed as some model augmentation techniques for model diversity preservation. We use the techniques in the experiments of ImageNet Subset to reduce training effort.

4. Visualization of Condensed Synthetic Sets

Fig. 1 and Fig. 2 visualize the synthetic sets condensed by our method on CIFAR-100 with 1 image per class with and without the proposed partition and expansion augmentation, respectively. Interestingly, there are some repetitive textures in all images of Fig. 2, which might indicate lower utilization of pixels as fine-grained image details are discarded during forward propagation. For more information, we also visualize the results of original DM [1] in Fig. 3, whose textures share similar patterns to ours. Fig. 4 and Fig. 5 visualize the results of CIFAR-10 with 10 images per class with/without our augmentation respectively.

References

[1] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *arXiv preprint arXiv:2110.04181*, 2021. 1, 2, 4

[2] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *Ninth International Conference on Learning Representations 2021*, 2021. 1

Figure 1. Visualization of synthetic set on CIFAR-100 with 1 image per class **with** our partition and expansion augmentation.

Figure 2. Visualization of synthetic set on CIFAR-100 with 1 image per class **without** our partition and expansion augmentation.

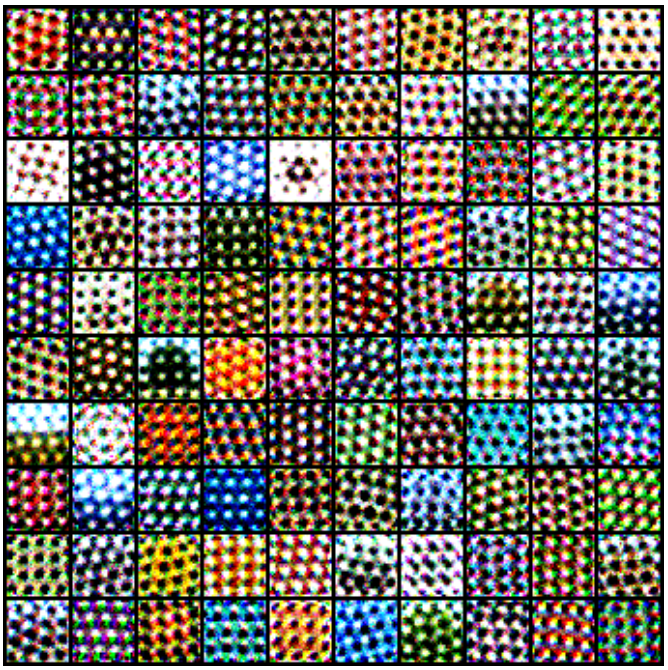

Figure 3. (**Original DM [1] results**) Visualization of synthetic set on CIFAR-100 with 1 image per class **without** partition and expansion augmentation.

Figure 4. Visualization of synthetic set on CIFAR-10 with 10 image per class **with** our partition and expansion augmentation.

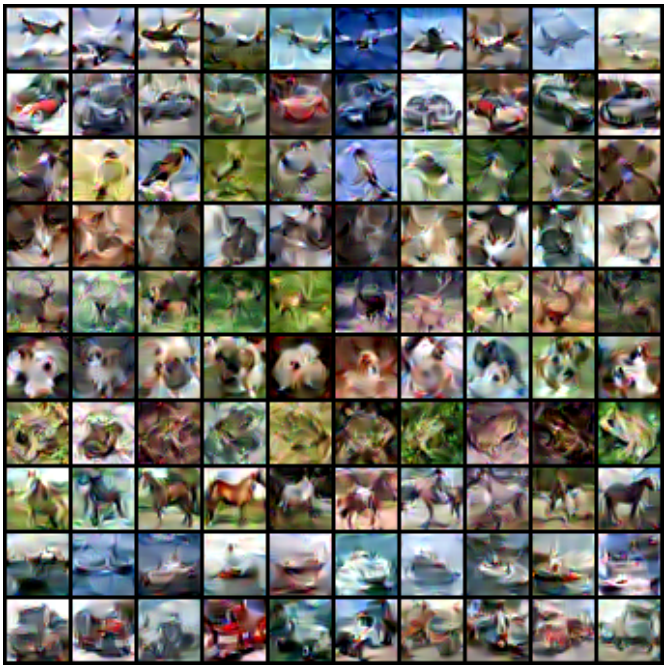

Figure 5. Visualization of synthetic set on CIFAR-10 with 10 image per class **without** our partition and expansion augmentation.