Teddy: Efficient Large-Scale Dataset Distillation via Taylor-Approximated Matching -Supplementary Materials-

Ruonan Yu, Songhua Liu, Jingwen Ye, and Xinchao Wang*

National University of Singapore
{ruonan,songhua.liu}@u.nus.edu, {jingweny,xinchao}@nus.edu.sg

A Comprehensive Theoretical Analysis

Given the original dataset, denoted as $\mathcal{T} = (X_t, Y_t)$, where $X_t \in \mathbb{R}^{N_t \times d}$ and $Y_t \in \mathbb{R}^{N_t \times c}$, DD aims to learn a much smaller synthetic dataset $\mathcal{S} = (X_s, Y_s)$, where $X_s \in \mathbb{R}^{N_s \times d}$ and $Y_s \in \mathbb{R}^{N_s \times c}$, such that models trained on $\mathcal{S}$ and $\mathcal{T}$ are of comparable performance. Here, N_t and N_s are the number of data in $\mathcal{T}$ and $\mathcal{S}$, respectively, and $N_s \ll N_t$. d is number of features for each data, and c represents the number of classes for classification task.

For better generalization ability of $\mathcal{S}$, it always adopts multiple inner loops to train a novel student model. Considering the small size of $\mathcal{S}$, models trained on it can easily overfit. Thus, we have $l(\mathcal{S}; \theta_{\mathcal{S}}^{(T)}) < \epsilon$, where $l(\cdot; \theta)$ is the loss function, T is the number of inner loops, and ϵ is close to zero. With the Karush-Kuhn-Tucker (KKT) conditions, the problem can be formulated as:

$$\begin{aligned} \mathcal{S} &= \arg\min_{\mathcal{S}} \mathbb{E}_{\theta^{(0)} \sim \Theta}[l(\mathcal{T}; \theta_{\mathcal{S}}^{(T)}) + u(l(\mathcal{S}; \theta_{\mathcal{S}}^{(T)}) - \epsilon)] \\ &= \arg\min_{\mathcal{S}} \mathbb{E}_{\theta^{(0)} \sim \Theta}[l(\mathcal{T}; \theta_{\mathcal{S}}^{(T)}) + ul(\mathcal{S}; \theta_{\mathcal{S}}^{(T)})], \end{aligned} \tag{1}$$

where u is the Lagrange multiplier, Θ is the distribution of networks, and

$$\begin{aligned} \theta_{\mathcal{S}}^{(t)} &= \theta^{(0)} - \alpha \sum_{i=0}^{t-1} g_{\mathcal{S}}^{(i)} = \theta_{\mathcal{S}}^{(t-1)} - \alpha g_{\mathcal{S}}^{(t-1)}, \\ g_{\mathcal{S}}^{(t)} &= \nabla_{\theta_{\mathcal{S}}^{(t)}} l(\mathcal{S}; \theta_{\mathcal{S}}^{(t)}). \end{aligned} \tag{2}$$

Following the Proposition 1, we recursively apply the first-order Taylor expansion to the first term of the optimization objective, and it can be transformed into

* Corresponding author.

follows:

$$\begin{aligned} l(\mathcal{T};\theta_{\mathcal{S}}^{(T)}) &= l(\mathcal{T};\theta_{\mathcal{S}}^{(T-1)} - \alpha g_{\mathcal{S}}^{(T-1)}) \\ &= l(\mathcal{T};\theta_{\mathcal{S}}^{(T-1)}) - \alpha g_{\mathcal{T}}^{(T-1)} \cdot g_{\mathcal{S}}^{(T-1)} \\ &= l(\mathcal{T};\theta_{\mathcal{S}}^{(T-i)}) - \alpha \sum_{t=T-i}^{T-1} g_{\mathcal{T}}^{(t)} \cdot g_{\mathcal{S}}^{(t)} \\ &= l(\mathcal{T};\theta^{(0)}) - \alpha \sum_{t=0}^{T-1} g_{\mathcal{T}}^{(t)} \cdot g_{\mathcal{S}}^{(t)}, \end{aligned} \tag{3}$$

where $g_{\mathcal{T}}^{(t)} = \nabla_{\theta_{\mathcal{S}}^{(t)}} l(\mathcal{T};\theta_{\mathcal{S}}^{(t)})$. The Taylor-approximated version of the optimization objective is as follows:

$$\begin{aligned} \mathcal{S} &= \arg\min_{\mathcal{S}}[-\alpha \sum_{t=0}^{T-1} g_{\mathcal{T}}^{(t)} \cdot g_{\mathcal{S}}^{(t)} + u l(\mathcal{S};\theta_{\mathcal{S}}^{(T)})] \\ &= \arg\min_{\mathcal{S}}[-\sum_{t=0}^{T-1} \frac{g_{\mathcal{T}}^{(t)} \cdot g_{\mathcal{S}}^{(t)}}{||g_{\mathcal{T}}^{(t)}||||g_{\mathcal{S}}^{(t)}||} + u^{'} l(\mathcal{S};\theta_{\mathcal{S}}^{(T)})], \end{aligned} \tag{4}$$

where $\theta^{(0)} \sim \Theta$, and $u^{'} = \frac{u}{\alpha||g_{\mathcal{T}}^{(t)}||||g_{\mathcal{S}}^{(t)}||}$. The first term of the Eq. 4 is the sum of the cosine distance of the gradients for the synthetic and original data, respectively, on the all checkpoints of the student trajectory. In other words, the meta-learning-based optimization objective can be Taylor-approximated as the sum of the multi-step gradient matching. Here, we denote $e^{(t)} = \frac{g^{(t)}}{||g^{(t)}||}$, and for each gradient matching at step t, we have:

$$-e_{\mathcal{T}}^{(t)} \cdot e_{\mathcal{S}}^{(t)} = \frac{1}{2}||e_{\mathcal{T}}^{(t)} - e_{\mathcal{S}}^{(t)}||^2 - 1. \tag{5}$$

So for each step of gradient matching, we only need to optimize the l^2 distance between the gradients of the original data and the synthetic data, such that:

$$\mathcal{S} = \arg\min_{\mathcal{S}}[\frac{1}{2} \sum_{t=0}^{T-1} ||e_{\mathcal{T}}^{(t)} - e_{\mathcal{S}}^{(t)}||^2 + u^{'} l(\mathcal{S};\theta_{\mathcal{S}}^{(T)})]. \tag{6}$$

However, the Eq. 6 shows that each update step of $\mathcal{S}$ should compute the gradient of $\mathcal{S}$ and $\mathcal{T}$ on all checkpoints of the student trajectory, which is a large amount of computational costs. Here, we reduce the costs by adopting another approximation strategy, turning the second-order optimization into the first-order one. Following the Proposition 2, for simplicity in explanation, we only consider the last linear layer of the network is updated during the generation process, and the parameter is denoted as W. The preceding layers are regraded as the feature

extractor, denoted as f_θ. For each gradient, we have:

$$\begin{aligned} g &= \frac{\partial l(\cdot;\theta)}{\partial W} = \frac{1}{|X|} f_\theta(X)^T (f_\theta(X)W - Y) \\ &= \frac{1}{|X|} f_\theta(X)^T f_\theta(X)W - \frac{1}{|X|} f_\theta(X)^T Y. \end{aligned} \tag{7}$$

The first term of Eq. 7 is a weighted covariance matrix of the feature space, and the second term is the class-wise mean of the feature space. The gradient matching for l^2 distance can be transformed as:

$$\begin{aligned} &||(\frac{1}{N_t} f_\theta(X_t)^T f_\theta(X_t) - \frac{1}{N_s} f_\theta(X_s)^T f_\theta(X_s))W \\ &- (\frac{1}{N_t} f_\theta(X_t)^T Y - \frac{1}{N_s} f_\theta(X_s)^T Y)||^2 \frac{1}{||W||^2} \\ \leq &||\frac{1}{N_t} f_\theta(X_t)^T f_\theta(X_t) - \frac{1}{N_s} f_\theta(X_s)^T f_\theta(X_s)||^2 \\ &+ ||\frac{1}{N_t} f_\theta(X_t)^T Y - \frac{1}{N_s} f_\theta(X_s)^T Y||^2 \frac{1}{||W||^2}. \end{aligned} \tag{8}$$

From Eq. 8, it shows that the upper bound of the gradient matching is the first-order and the second-order statistic information matching in feature space. Here, we need to consider the prior condition such that $\theta^{(0)} \sim \Theta$. It means that for every $\theta^{(0)}$ samples from the distribution Θ, Eq. 8 reaches the minimum value. In this case, the first term of Eq. 8, the covariance for the original data and synthetic data, is equivalent for all $\theta^{(0)}$ samples, and the class-wise mean. Also, the equality in Eq. 8 holds. As for $\theta^{(t)}$, as long as the samples are sufficient, the condition still holds. To be more specific, here we assume $f_\theta(X_t) \in \mathbb{R}^{N_t \times f_d}$, and $f_\theta(X_s) \in \mathbb{R}^{N_s \times f_d}$, f_d is the dimension of the feature. $f_\theta(X_t) = [f_\theta(x_1^t)^T, \ldots, f_\theta(x_{N_t}^t)^T]^T$, and $f_\theta(x_i^t) \in \mathbb{R}^{f_d}$. Also, $f_\theta(X_s) = [f_\theta(x_1^s)^T, \ldots, f_\theta(x_{N_s}^s)^T]^T$, and $f_\theta(x_i^s) \in \mathbb{R}^{f_d}$. $f_\theta(X_t) = [F_1^t, \ldots, F_{f_d}^t]$, $F_i^t \in \mathbb{R}^{N_t}$, and $f_\theta(X_s) = [F_1^s, \ldots, F_{f_d}^s]$, $F_i^s \in \mathbb{R}^{N_s}$. For the first term of Eq. 8, we have:

$$\begin{aligned} &\frac{1}{N_t} f_\theta(X_t)^T f_\theta(X_t) - \frac{1}{N_s} f_\theta(X_s)^T f_\theta(X_s) \\ =&\frac{1}{N_t}[(F_1^t)^T, \ldots, (F_{f_d}^t)^T]^T [F_1^t, \ldots, F_{f_d}^t] \\ &- \frac{1}{N_s}[(F_1^s)^T, \ldots, (F_{f_d}^s)^T]^T [F_1^s, \ldots, F_{f_d}^s] \\ =&\begin{bmatrix} Var(F_1^t) & \cdots & Cov(F_1^t, F_{f_d}^t) \\ \vdots & \ddots & \vdots \\ Cov(F_{f_d}^t, F_1^t) & \cdots & Var(F_{f_d}^t) \end{bmatrix} \\ &- \begin{bmatrix} Var(F_1^s) & \cdots & Cov(F_1^s, F_{f_d}^s) \\ \vdots & \ddots & \vdots \\ Cov(F_{f_d}^s, F_1^s) & \cdots & Var(F_{f_d}^s) \end{bmatrix}. \end{aligned} \tag{9}$$

Eq. 9 shows that if the covariance matching holds, then variance matching also holds. To improve the calculation efficiency, here we adopt variance matching instead of covariance matching. As for the second term, it is the class-wise mean matching in feature space. Specifically, we have:

$$\begin{aligned}
&\frac{1}{N_t} f_\theta(X_t)^T Y - \frac{1}{N_s} f_\theta(X_s)^T Y \\
=&\frac{1}{N_t}[f_\theta(x_1^t), \ldots, f_\theta(x_{N_t}^t)]\mathcal{M}_c^t \\
&- \frac{1}{N_s}[f_\theta(x_1^s), \ldots, f_\theta(x_{N_s}^s)]\mathcal{M}_c^s.
\end{aligned} \tag{10}$$

Here, $\mathcal{M}_c^t \in \mathbb{R}^{N_t \times c}$ and $\mathcal{M}_c^s \in \mathbb{R}^{N_s \times c}$, are the matrices to indicate whether the samples belong to the classes. For instance, $M_{ij} = 1$ means the i^{th} sample is belong to the j^{th} class, otherwise, it is not. The Eq. 10 can be:

$$\begin{aligned}
&\frac{1}{N_t}[\sum_{i=1}^{N_t} f_\theta(x_i^t) M_{i1}^t, \ldots, \sum_{i=1}^{N_t} f_\theta(x_i^t) M_{ic}^t] \\
&- \frac{1}{N_s}[\sum_{i=1}^{N_s} f_\theta(x_i^s) M_{i1}^s, \ldots, \sum_{i=1}^{N_s} f_\theta(x_i^s) M_{ic}^s] \\
=&\frac{1}{N_t}[\sum_{x_i^t \in Cl_1} f_\theta(x_i^t), \ldots, \sum_{x_i^t \in Cl_c} f_\theta(x_i^t)] \\
&- \frac{1}{N_s}[\sum_{x_i^s \in Cl_1} f_\theta(x_i^s), \ldots, \sum_{x_i^s \in Cl_c} f_\theta(x_i^s)],
\end{aligned} \tag{11}$$

where Cl_i is the set of samples belong to i^{th} class. When the classes of the original dataset is balanced, or the number of samples in every class of the original dataset is same, the second term of Eq. 8 can be replaced by the global mean. Therefore, the Eq. 6 can be transformed as follows:

$$\begin{aligned}
&\sum_{t=0}^{T-1}(\sum_l ||\mu_l(f_{\theta_\mathcal{S}^{(t)}}(X_s)) - \mu_l(f_{\theta_\mathcal{S}^{(t)}}(X_t))||_2 \\
&+ \sum_l ||\sigma_l^2(f_{\theta_\mathcal{S}^{(t)}}(X_s)) - \sigma_l^2(f_{\theta_\mathcal{S}^{(t)}}(X_t))||_2) \\
&+ u \cdot l(\mathcal{S}; \theta_\mathcal{S}^{(T)}),
\end{aligned} \tag{12}$$

where $\theta^{(0)} \sim \Theta$, μ_l and σ_l^2 refer to the mean and variance of the l^{th} layer features. Although Eq. 12 has significantly improved efficiency, computing the mean and variance of checkpoints at each inner loop remains a substantial computational overhead, especially when there are numerous inner loops and a relatively large original dataset. Here, we propose substituting performance-comparable weak teachers for student models and reducing the number of rounds required for

inner loop. As for the student trajectory starting at t and ending at $t+m$, we have:

$$\begin{aligned} l(\mathcal{T};\theta_{\mathcal{S}}^{(t+m)}) &= l(\mathcal{T};\theta_{\mathcal{S}}^{(t)} - \alpha \sum_{i=t}^{t+m-1} g_{\mathcal{S}}^{(i)}) \\ &= l(\mathcal{T};\theta_{\mathcal{S}}^{(t)}) - \alpha(\sum_{i=t}^{t+m-1} g_{\mathcal{S}}^{(i)}) \cdot g_{\mathcal{T}}^{(t)}. \end{aligned} \tag{13}$$

From Eq. 13, it shows that the multi-step gradient of the model on the synthetic dataset compared to the single-step gradient on the original dataset. Therefore, we cache the weak teacher for every m steps from T_b to T_e. Here, we also utilize the running mean and running variance stored in the batch normalization layers of weak teachers to replace the mean and variance of the original dataset in feature space, reducing computational costs. The optimization objective is as follows:

$$\begin{aligned} &\sum_{t=T_b}^{T_e} (\sum_{l} ||\mu_l(f_{\theta_{\mathcal{T}}^{(t)}}(X_s)) - RM^l_{\theta_{\mathcal{T}}^{(t)}}(X_t))||_2 \\ &+ \sum_{l} ||\sigma_l^2(f_{\theta_{\mathcal{T}}^{(t)}}(X_s)) - RV^l_{\theta_{\mathcal{T}}^{(t)}}(X_t))||_2 \\ &+ u \cdot l(\mathcal{S};\theta_{\mathcal{T}}^{(t)})), \end{aligned} \tag{14}$$

where T_b and T_e is the starting checkpoint and the end point of the teacher training trajectory.

B Error Analysis for Taylor Approximation

In Eq. 3, we recursively apply the first-order Taylor expansion to the original optimization objective to decouple the bi-level optimization process. This section will further analyze the error caused by Taylor approximation part, and theoretical reasonability.

Proposition 1 *Taylor approximation minimizes the upper bound of the original loss function.*

Proof. Considering each Taylor approximation iteration independently is as follows:

$$\begin{aligned} l(\mathcal{T};\theta_{\mathcal{S}}^{(t)}) &= ||f(X_t;\theta_{\mathcal{S}}^{(t)}) - Y_t|| \\ &\leq ||f(X_t;\theta_{\mathcal{S}}^{(t)}) - f(X_t;\theta_{\mathcal{T}}^{(t)})|| + ||f(X_t;\theta_{\mathcal{T}}^{(t)}) - Y_t|| \\ &\leq \beta||\theta_{\mathcal{S}}^{(t)} - \theta_{\mathcal{T}}^{(t)}|| + ||f(X_t;\theta_{\mathcal{T}}^{(t)}) - Y_t|| \\ &= \beta||g_{\mathcal{S}}^{(t-1)} - g_{\mathcal{T}}^{(t-1)}|| + C \\ &= \beta g_{\mathcal{S}}^{(t-1)} \cdot g_{\mathcal{T}}^{(t-1)} + C, \end{aligned} \tag{15}$$

where β is the Lipschitz constant and C is irrelevant to the optimization.

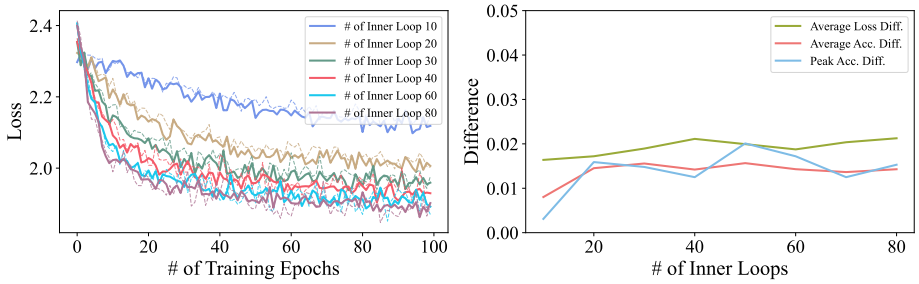

Fig. 1: Left: training loss of the original DD (dashed line) and our approximated objectives (solid line); Right: the difference of average loss, average accuracy and peak accuracy during training.

Imbalanced dataset Acc.(%) / F1(%)	SRe^{2}L	Ours	Ours with Class-wise Statistic
60%∼100%	24.2 / 24.2	**32.7** (+ 8.5) / **32.5** (+ 8.3)	**36.0** (+ 11.8) / **36.0** (+ 11.8)
40%∼100%	23.4 / 23.0	**33.6** (+ 10.2) / **33.3** (+ 10.3)	**35.6** (+ 12.2) / **35.3** (+ 12.3)

Table 1: Results on imbalanced dataset with each class has 60%-100% and 40%-100% samples.

Acc. (%)	5	10	50
SRe^{2}L	5.4	17.8	48.4
Ours	**21.0** (+ 15.6)	**37.2** (+ 19.4)	**58.5** (+ 10.1)

mAP (%)	10 Imgs	20 Imgs	40 Imgs
baseline	3.6	5.6	9.9
ours	6.4 (+ 2.8)	**10.4** (+ 4.8)	**15.3** (+ 5.4)

Table 2: Results on larger model (left) and other task (right).

We also conduct validation experiments under the setting of CIFAR10 IPC 5 with the original DD optimization objective and our Taylor-approximation version. The results are shown in Fig. 1. We evaluate the difference of average losses, average accuracy per iteration, and peak accuracy during training. The results demonstrate that the bound is tight in practice, and indicate that errors introduced by our approximation are negligible and the overall training dynamics are comparable.

C More Experimental Results

C.1 Experimental Results on Imbalanced Dataset

In Eq. 11, we assume that the dataset is balanced. This section will further analyze this condition and show the robustness of our proposed method. We conduct the experiments under the setting of Tiny-ImageNet IPC 20. For each class of Tiny-ImageNet, we randomly select 60%∼100% and 40%∼100% of the original daraset to build the imbalanced dataset. The results are shown in the Table 1. Even if imbalanced models are used, our method still demonstrates some resistance to this issue compared with the baseline as shown in Table 1. This can be attributed to more informative statistics from diversified teachers for

model-to-data synthesis. Moreover, the performance could be further improved by considering the class-wise statistics.

C.2 Experimental Results on Larger Network

To further demonstrate the effectiveness of our proposed method for larger networks, we conduct the experiments on ResNet50. Here, ResNet50 is utilized as the model pool base architecture, and the downstream network. The results are shown in the Table 2 (left). Our method achieves superior performance for larger models.

C.3 Experimental Results on Other Task

Current DD field mainly focuses on classification, with detection and other tasks research still blank. Here, we simply apply our method on the detection task to show the generalizability of our proposed method across different tasks. We conduct the experiments on Pascal VOC, and the architecture of the detector is Faster RCNN. Table 2 (right) shows the preliminary results of our method and baseline adapted to object detection on Pascal VOC, demonstrating the potential of our method in wider applications.

C.4 Experimental Results on Small Dataset

Our proposed method, Teddy, is designed mainly for large-scale datasets as we do several Taylor-approximation strategies to improve the efficiency of the original solution to the DD definition, which may cause information loss. However, it still shows quite competitive performance compared with baselines SRe^{2}L [1] and DM [2] on small datasets. Here, we adopt CIFAR-10 and CIFAR-100 for experiments. Specifically, we drop the soft label adopted in both SRe^{2}L and our method for fair comparison. Also, we only use the DSA strategy for data augmentation. The evaluation results are shown in Table 3. It shows that our weak-teacher strategy is a more favorable surrogate of the original ones than fully-converged models (SRe^{2}L) and random initialization without training (DM).

D More Implementation Details

D.1 Model Pool Generation

In our experiments, we have two strategies to generate the model pool: prior-generation and post-generation. The base network architecture is ResNet18. For prior-generation, we follow the official torchvision code to train ResNet18 for ImageNet-1K and modified ResNet18 for Tiny-ImageNet. We adopt different stage teacher models for different IPC settings. The size of the model pool is 9 for ImageNet-1K and 8 for Tiny-ImageNet. For more details, please refer to Table 4.

Method	CIFAR-10			CIFAR-100		
	1	10	50	1	10	50
DM (noise) [2]	25.6 ± 0.2	49.8 ± 0.3	59.5 ± 0.1	11.4 ± 0.2	29.6 ± 0.3	36.9 ± 0.1
DM (real) [2]	26.0 ± 0.8	48.9 ± 0.6	63.0 ± 0.4	11.4 ± 0.3	29.7 ± 0.3	43.6 ± 0.4
SRe^{2}L [1] w/o soft label (noise)	24.9 ± 0.2	35.8 ± 0.4	37.0 ± 0.2	10.8 ± 0.2	18.4 ± 0.1	25.6 ± 0.1
SRe^{2}L [1] w/o soft label (real)	27.1 ± 0.4	44.7 ± 0.2	55.7 ± 0.2	10.7 ± 0.3	24.6 ± 0.1	39.2 ± 0.1
Ours w/o soft label (noise)	29.2 ± 0.3	51.0 ± 0.1	63.6 ± 0.1	12.5 ± 0.2	30.6 ± 0.1	44.4 ± 0.2
Ours w/o soft label (real)	**30.1 ± 0.4**	**53.0 ± 0.3**	**66.1 ± 0.1**	**13.5 ± 0.2**	**33.4 ± 0.1**	**49.4 ± 0.4**

Table 3: Results on small datasets CIFAR-10 and CIFAR-100. For a fair comparison, we drop the soft label adopted in both SRe^{2}L and our proposed method and only use DSA as the data augmentation strategy.

As for post-generation, we utilize the pre-trained ResNet18 provided by the official torchvision for ImageNet-1K and well-trained modified ResNet18 for Tiny-ImageNet. We use DepGraph to perform structural pruning with the random standard, and finetune the pruned models for very limited epochs, *e.g.*, 0-2 epochs, with learning rate from 0.1 to 0.001. It is worth noting that during finetuning, we simulate the accuracy curve of teacher models in trajectory-based model pool. The GFLOPs of the target pruned model is 1.2G, and the number of parameters is 7.72M. The size of model pool is 10 for ImageNet-1K and 9 for Tiny-ImageNet. For more details, please refer to Table 4.

D.2 Data Generation

For ImageNet-1K, we randomly select 3 models from the prior-generated model pool or 4 models from the post-generated model pool to generate the i^{th} image for all classes. As for Tiny-ImageNet, we randomly select 3 models from prior-generated model pool or 5 models from the post-generated model pool. The batch size is 100, and the number of synthetic data generation iterations is 6000. For more details, please refer to Table 5.

D.3 Validation

For the validation part, the generated data is first augmented using the CutMix strategy, and then the soft labels of the augmented data are generated by the model pool. The number of validation epochs for Tiny-ImageNet is 100, and for ImageNet-1K is 300. The training batch size is 256 for Tiny-ImageNet and 1024 for ImageNet-1K. The optimizer we adopted is AdamW, and the optimizer momentum is 0.9 and 0.999, respectively. The base learning rate is 0.001, the weight decay is 0.01, and the learning rate scheduler is Cosine.

E More Visualization Results

Here, we provide additional visualizations of the synthetic dataset generated by our method, including different settings on Tiny-ImageNet (Fig. 2) and

	Hypeparameter	Prior-Generation	Post-Generation
Tiny-ImageNet	Optimizer	SGD	SGD
	Base Learning Rate	0.2	0.1-0.001
	Learning Rate Scheduler	Cosine	Step
	Weight Decay	1e-4	1e-4
	Learning Rate Step Size	-	0-2
	Momentum	-	0.9
	Batch Size	256	32
	Model Pool Size	8	9
	Training / Finetuning Epochs	46, 50	0-2
	Stage of Teachers	11-46, with step 5; 16-50, with step 5	-
ImageNet-1K	Optimizer	SGD	SGD
	Base Learning Rate	0.1	0.1-0.001
	Learning Rate Scheduler	Step	Step
	Weight Decay	1e-4	1e-4
	Learning Rate Step Size	30	0-2
	Momentum	0.9	0.9
	Batch Size	32	32
	Model Pool Size	9	10
	Training / Finetuning Epochs	41, 61, 71	0-2
	Stage of Teachers	1-41, with step 5; 21-61, with step 5; 31-71, with step 5	-

Table 4: The hyper-parameters for model pool generation part.

	Hypeparameter	Prior-Generation	Post-Generation
Tiny-ImageNet	Optimizer	Adam	Adam
	Base Learning Rate	0.1	0.1
	Learning Rate Scheduler	Cosine	Cosine
	Weight of BN Loss	1.0	1.0
	Weight Decay	1e-4	1e-4
	Momentum	0.5, 0.9	0.5, 0.9
	Batch Size	100	100
	Model Pool Size	8	9
	Generation Iterations	4000	4000
	Number of Teachers Ensemble	3 for data generation at one time	5 for data generation at one time
ImageNet-1K	Optimizer	Adam	Adam
	Base Learning Rate	0.25	0.25
	Learning Rate Scheduler	Cosine	Cosine
	Weight of BN Loss	0.01	0.01
	Weight Decay	1e-4	1e-4
	Momentum	0.5, 0.9	0.5, 0.9
	Batch Size	100	100
	Model Pool Size	9	10
	Generation Iterations	6000	6000
	Number of Teachers Ensemble	3 for data generation at one time	4 for data generation at one time

Table 5: The hyper-parameters for data generation part.

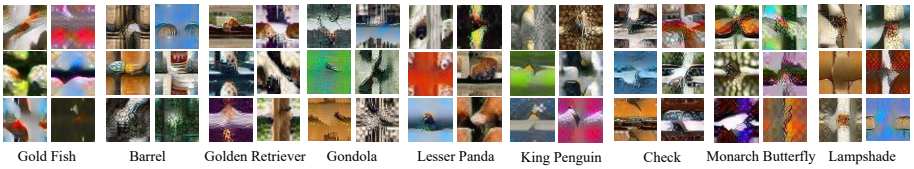

Fig. 2: Visualization results of our proposed method for Tiny-ImageNet IPC 50.

ImageNet-1K (Fig. 3). From a visualization perspective, our method showcases a broader range of perspectives and diversity. Here, for instance, in the whistle category, it can be observed that the focal points of the generated model pool by different methods are distinct. Post-generation focuses on the object itself, while prior-generation also pays attention to the surrounding environment, *e.g.*, the animal blowing the whistle.

References

1. Yin, Z., Xing, E., Shen, Z.: Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. Advances in Neural Information Processing Systems **36** (2024)
2. Zhao, B., Bilen, H.: Dataset condensation with distribution matching. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6514–6523 (2023)

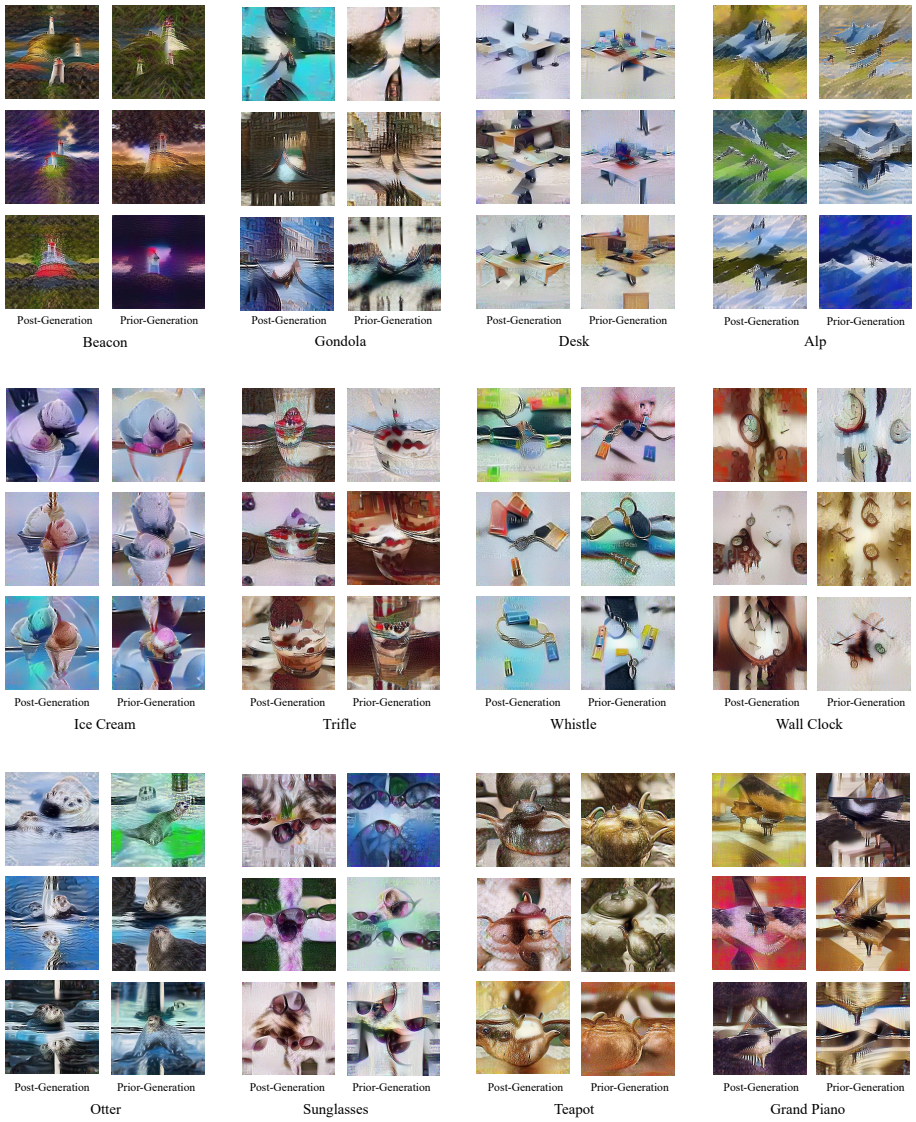

Fig. 3: Visualization results of our proposed method for ImageNet-1K IPC 50.