DKDM: Data-Free Knowledge Distillation for Diffusion Models with Any Architecture

Supplementary Material

7. Training and Sampling of DDPMs

In this section, we present the algorithms for training and sampling from standard DDPMs. The specific details have been previously introduced in Sec. 2.

Algorithm 2 Diffusion Models Training

repeat
  $\boldsymbol{x}^0 \sim q(\boldsymbol{x}^0), t \sim [1, ..., T], \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  obtain the noisy sample $\boldsymbol{x}^t$ using Eq. (4)
  compute L_{simple} using Eq. (11)
  compute L_{vlb} using Eqs. (9) and (12)
  take a gradient descent step on $\nabla_{\boldsymbol{\theta}} L_{\text{hybrid}}$
until converged

Algorithm 3 Diffusion Models Sampling

$\hat{\boldsymbol{x}}^T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t := T, ..., 1$ **do**
  if $t > 1$ then $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, else $\boldsymbol{z} = \mathbf{0}$
  $\hat{\boldsymbol{x}}^{t-1} = \frac{1}{\sqrt{\alpha_t}}(\hat{\boldsymbol{x}}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^t, t)) + \sqrt{\Sigma_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^t, t)}\boldsymbol{z}$
end for
return $\hat{\boldsymbol{x}}^0$

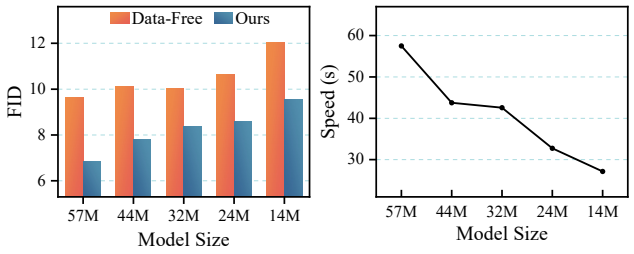
8. Hyperparameters

8.1. Hyperparameters for Main Results

For experiments of both dynamic iterative distillation and baselines in pixel space, we use the hyperparameters specified by Ning et al. [37], which are also in line with those adopted by Dhariwal and Nichol [8], as reported in Tab. 6. Settings of these training process are basically the same as those used in [8] and [37], including mixed precision training, EMA and so on. All the models are trained on NVIDIA A100 GPUs (with 40G memory). For experiments in latent sace, we follow hyperparameters utilized by Rombach et al. [44], detailed in Tab. 7.

8.2. Hyperparameters for Cross-Architecture Distillation

Similar to the configuration used by Peebles and Xie [39], the hyperparameters employed for the ViT-based diffusion model in Sec. 4.2 are presented in Tab. 10. This particular configuration was chosen due to the characteristic of ViT-based diffusion models generally requiring more time for image generation compared to their CNN-based counterparts. To illustrate this point, when generating 2,500 images on single A100 40GB GPU with 50 Improved DDPM steps, it takes approximately 57 seconds for a 57M CNN diffusion model, whereas a 19M ViT diffusion model requires 66 seconds. Our final choice of this configuration was driven by the aim to achieve fairness in experimentation and analysis.



(a) Performance (b) Speed

Figure 6. Performance and speed of DMs across different architectures on CIFAR10.

8.3. Hyperparameters for Various Architectures

Tab. 8 shows the hyperparameters of different students trained in Sec. 9 for results on various architectures.

9. Results on Various Architectures

We further compare the performance between our method and data-free training on CIFAR10 across a diverse range of architectures. Specifically, we tested five different model sizes by directly specifying the architecture. Both the teacher and student models employ Convolutional Neural Networks (CNNs) and the results are shown in Fig. 6. Detailed descriptions of these architectures are available in Sec. 8.3. Typically, we distilled a 14M model from a 57M teacher model, maintaining competitive performance and doubling the generation speed, compared with the 57M model through data-free training. Here the speed is measured by the average time taken to generate 256 images on a single NVIDIA A100 GPU. Additionally, the 44M and 33M student models demonstrated similar speeds, suggesting that DKDM could benefit from integrating with efficient architectural design techniques to further enhance the speed and quality of DMs. This aspect, however, is beyond our current scope and is designated for future research.

Hyperparameter	CIFAR10 32x32		CelebA 64x64		ImageNet 32x32	
	Teacher	Student	Teacher	Student	Teacher	Student
Model Parameters	57M	14M	295M	57M	57M	14M
Diffusion Steps	1,000	1,000	1000	1000	1000	1,000
Noise Schedule	cosine	cosine	cosine	cosine	cosine	cosine
Channels	128	64	192	96	128	64
Residual Blocks	3	3	3	2	3	3
Channels Multiple	1,2,2,2	1,2,2,2	1,2,3,4	1,2,3,4	1,2,2,2	1,2,2,2
Heads Channels	32	32	64	32	32	32
Attention Resolution	16,8	16,8	32,16,8	32,16,8	16,8	16,8
BigGAN Up / Downsample	True	True	True	True	True	True
Dropout	0.3	0.3	0.1	0.1	0.3	0.3
Batch Size	128	128	256	256	512	512
Distillation Iterations	-	200K	-	100K	-	500K
Learning Rate	-	1e-4	-	1e-4	-	1e-4
Early Stop	-	True	-	True	-	False
ρ	-	0.4	-	0.4	-	0.4

Table 6. Hyperparameters for main results in pixel space. "Teacher" refers to the architecture of pretrained models we utilized. "Student" denotes the student architecture we adopt.

Hyperparameter	CelebA-HQ 256x256		FFHQ 256x256	
	Teacher	Student	Teacher	Student
f	4	4	4	4
z-shape	64x64x3	64x64x3	64x64x3	64x64x3
$\|Z\|$	8192	8192	8192	8192
Diffusion steps	1000	1000	1000	1000
Noise Schedule	linear	linear	linear	linear
Model Parameters	274 M	86 M	274 M	86 M
Channels	224	128	224	128
Depth	2	2	2	2
Channel Multiplier	1, 2, 3, 4	1, 2, 3, 4	1,2,3,4	1,2,3,4
Attention Resolutions	32, 16, 8	32, 16, 8	32, 16, 8	32, 16, 8
Head Channels	32	32	32	32
Batch Size	48	48	42	42
DistillationIterations	-	410k	-	640k
Learning Rate	-	9.60e-05	-	8.40e-05
Early Stop	-	False	-	False
ρ	-	4	-	4

Table 7. Hyperparameters for main results in latent space. "Teacher" refers to the architecture of pretrained models we utilized. "Student" denotes the student architecture we adopt.

10. GPU Memory and Training Speed

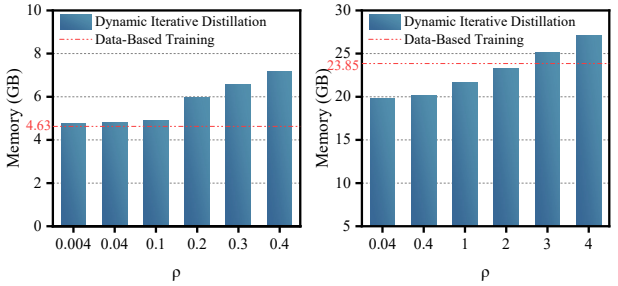
This section compares the training memory and speed between data-based training and our dynamic iterative distillation. As detailed in Sec. 3.3, the size of the expanded batch set $\hat{\mathcal{B}}_1^+$ used in our approach is $|\hat{\mathcal{B}}_j^+| = \rho T|\hat{\mathcal{B}}_j^s|$, with ρ serving as the controlling factor. Increasing ρ evidently raises the GPU memory requirements, as shown in Fig. 7. However, we observed that the incremental GPU memory consumption is relatively minor. Interestingly, our method consumes almost the same amount of GPU memory in latent space as data-based training. This efficiency is achieved because our method eliminates the need to convert from pixel space to latent space—a process required in data-based training during each training iteration. Therefore, as

Model Size	**57M**	**44M**	**32M**	**24M**	**14M**
Diffusion steps	1,000	1,000	1,000	1,000	1,000
Noise schedule	cosine	cosine	cosine	cosine	cosine
Channels	128	128	96	96	64
Residual Blocks	3	2	3	2	3
Channels Multiple	1, 2, 2, 2	1, 2, 2, 2	1, 2, 2, 2	1, 2, 2, 2	1, 2, 2, 2
Heads Channels	32	32	32	32	32
Attention Resolution	16, 8	16, 8	16, 8	16, 8	16, 8
BigGAN Up / Downsample	True	True	True	True	True
Dropout	0.3	0.3	0.3	0.3	0.3
Batch Size	128	128	128	128	128
Distillation Iterations	200K	200K	200K	200K	200K
Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4
Early Stop	True	True	True	True	True
ρ	0.4	0.4	0.4	0.4	0.4

Table 8. Hyperparameters for CNN diffusion models of various architectures on CIFAR10.

	CIFAR10 32		FFHQ 256	
	Data-Based Training	Ours	Data-Based Training	Ours
Training Time per 100 Iterations	18.00s	19.17s	68.77s	55.52

Table 9. Training speed comparison for CIFAR10 (pixel space) and FFHQ 256 (latent space) on a single A100 GPU. Data-free and data-based training yield equivalent results.



(a) CIFAR10 32 (b) FFHQ 256

Figure 7. GPU memory occupied during training on CIFAR10 in pixel space and FFHQ 256 in latent space. Data-free and data-based training yield equivalent results.

Model Size	Layers N	Patch Size	Hidden size d	Heads
19M	7	2	384	6

Table 10. Hyperparameters for ViT-based diffusion models.

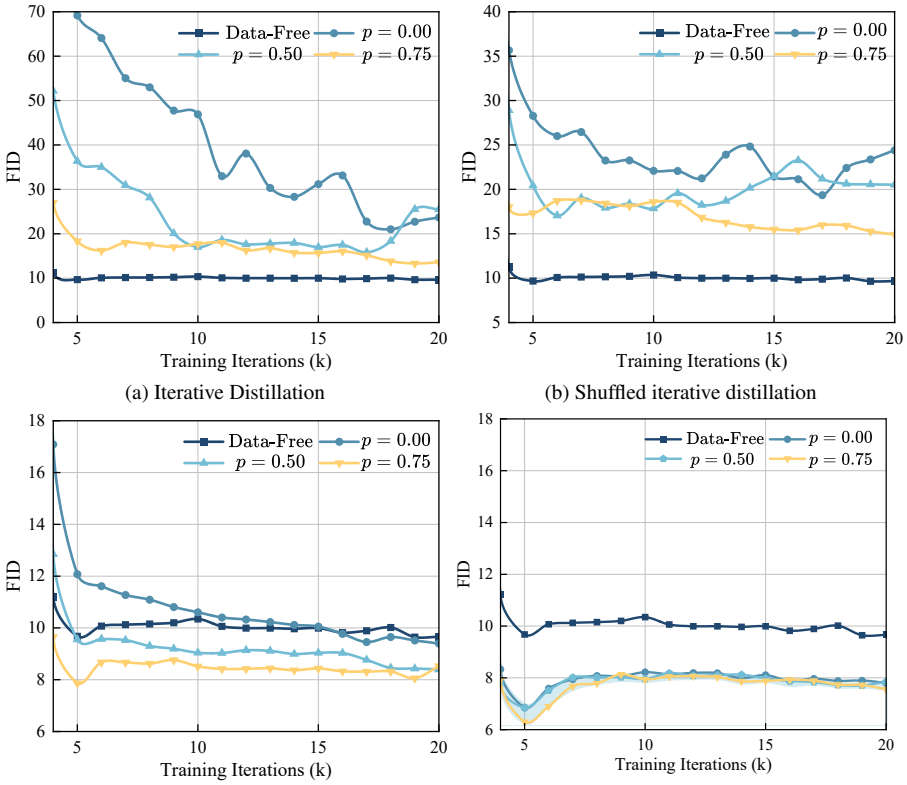
indicated in Tab. 9, our method demonstrates faster training speeds in latent space than data-based training.

11. Analysis: Random Discard

During our exploration, we discovered that the utilization of the **Random Discard** technique proves to be a straightforward yet highly effective approach for enhancing the distillation process. The idea behind it involves the random elimination of some batch of noisy samples generated by the teacher model during the iterative distillation. For instance, in iterative distillation during the initial five training iterations, batches $\hat{\mathcal{B}}_1, \hat{\mathcal{B}}_3, \hat{\mathcal{B}}_4$ may be discarded, while $\hat{\mathcal{B}}_2, \hat{\mathcal{B}}_5$ are utilized for the student's learning.

We present an analysis of the impact of random discarding in our devised methodologies. Specifically, we introduce the parameter p to denote the probability of discarding certain noisy samples. Subsequently, we apply varying discard probabilities to the iterative distillation, shuffled iterative distillation, and dynamic iterative distillation, and assess their respective performance alterations over a training duration of 200k iterations without early stop.

The outcomes are presented in Fig. 8. It is noteworthy that both iterative distillation and shuffled iterative distillation face limitations in constructing flexible batches, where random discard emerges as a noteworthy solution to enhance their efficacy. Conversely, for dynamic iterative distillation, when ρ attains a sufficiently large value, it be-



(a) Iterative Distillation (b) Shuffled iterative distillation

(c) Dynamic iterative distillation with $\rho = 0.05$ (d) Dynamic iterative distillation with $\rho = 0.4$

Figure 8. Effect of random discard on our iterative distillation, shuffled iterative distillation and dynamic iterative distillation without early stop. Random discard can not improve our dynamic iterative distillation with $\rho = 0.4$ and thus removed from our final method.

comes apparent that random discard fails to confer additional advantages. This observation underscores the inherent stability of our dynamic iterative distillation method and we ultimately omitted random discard from the final implementation. This is beneficial because of its inefficiency in requiring the teacher model to prepare a larger number of noisy samples.

12. Discussion and Future Work

The primary concept of our proposed DKDM paradigm is illustrated in Fig. 9. In this paradigm, the teacher DM $\boldsymbol{\theta}_T$, is trained on a real dataset $\mathcal{D}$, which follows the distribution $\mathcal{D}'$. There are two critical relationships between $\mathcal{D}$ and $\mathcal{D}'$. First, the distribution $\mathcal{D}'$ of a well-trained teacher closely approximates $\mathcal{D}$. Second, the FID scores, when computed using $\mathcal{D}$ as a reference, correlate with those using $\mathcal{D}'$, as demonstrated by the linear fitting in Fig. 9. This correlation underpins the effectiveness of the DKDM paradigm. By transferring the distribution $\mathcal{D}'$ from the teacher DM to a lighter student DM, DKDM enables the student to generate data whose distribution closely approximates $\mathcal{D}$.

However, in practice, there is invariably some discrepancy between $\mathcal{D}$ and $\mathcal{D}'$, limiting the performance of the

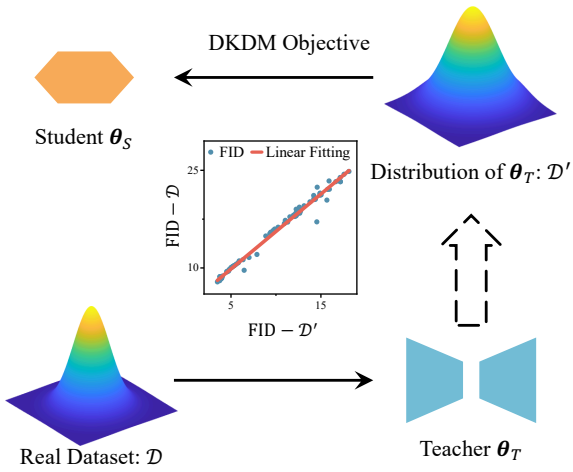


Figure 9. The DKDM paradigm learns distribution $\mathcal{D}'$ integrated in teacher DMs, which may deviate from the original dataset $\mathcal{D}$.

student model. We report these scores, denoted as FID$'$ and sFID$'$, calculated over the distribution $\mathcal{D}'$ instead of $\mathcal{D}$ in Tab. 11. The results indicate that the FID$'$ and sFID$'$ scores of the student closely mirror those of the teacher, suggesting

	FID	sFID	FID′	sFID′
Teacher	4.45	7.09	2.09	3.92
Data-Free Training	9.64	11.64	4.13	5.19
Dynamic Iterative Distillation	**6.85**	**8.01**	**2.60**	**4.14**

Table 11. Results on CIFAR10 calculated over the source $\mathcal{D}$ and that over distribution $\mathcal{D}'$ of $\boldsymbol{\theta}_T$. Here the architecture of the student is the same as the teacher.

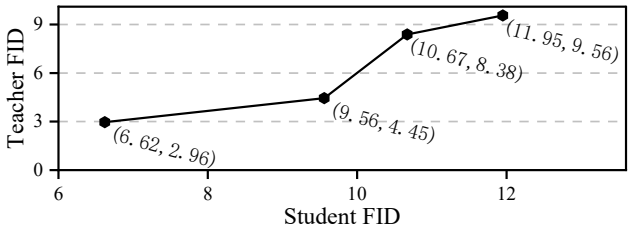


Figure 10. The relationship between the performances of teacher and student on CIFAR10.

effective optimization. Nevertheless, these scores are inferior to those of the teacher, primarily due to the gap between $\mathcal{D}$ and $\mathcal{D}'$. As illustrated by Fig. 10, the performance of student is strongly related to that of the teacher. Therefore, a potential solution to enhance DKDM involves improving the generative capabilities of the teacher, which we leave as a direction for future work.

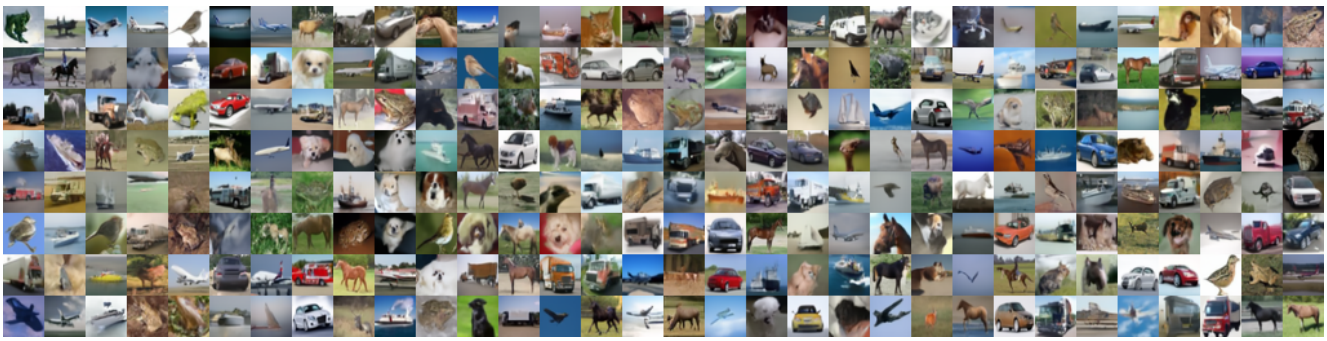


Figure 11. Selected samples generated by our student model on CIFAR10 32×32.

Figure 12. Selected samples generated by our student model on CelebA 64×64.

Figure 13. Selected samples generated by our student model on ImageNet 32×32.

Figure 14. Selected samples generated by our student model on CelebA-HQ 256×256.

Figure 15. Selected samples generated by our student model on FFHQ 256×256.