Reviving ConvNeXt for Efficient Convolutional Diffusion Models

Supplementary Material

We provide the following supplementary sections:

- Section A: Overview of denoising diffusion probabilistic models (DDPMs).
- Section B: Hyperparameter and implementation details.
- Section C: Descriptions of evaluation metrics.
- Section D: Descriptions of baseline models.
- Section E: Additional results and analyses on model scalability.
- Section F: Frequency-based analysis of model behavior.
- Section G: Additional analyses of architectural variants.
- Section H: Experimental results on text-to-image generation.
- Section I: Additional quantitative results and analyses.
- Section J: Additional qualitative results and visual samples.

A. Overview of denoising diffusion probabilistic models

Diffusion models [18, 55] aim to model a target distribution $p(x)$ by learning a gradual denoising process from Gaussian noise $\mathcal{N}(0, I)$ to $p(x)$. Specifically, the model learns a *reverse* process $p_\theta(x_{t-1}|x_t)$ of a predefined *forward* diffusion process $q(x_t|x_{t-1})$, which progressively adds Gaussian noise over T timesteps.

For an initial sample $x_0 \sim p(x)$, the *forward* process is defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}\Big(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\Big)\,, \tag{1}$$

where $\beta_t \in (0, 1)$ is a variance schedule. A closed-form expression of $q(x_t|x_0)$ can also be derived as:

$$q(x_t|x_0) = \mathcal{N}\big(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t) I\big)\,, \quad \bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s). \tag{2}$$

The denoising diffusion probabilistic model (DDPM) [18] parameterizes the *reverse* transition as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\Big(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\, \epsilon_\theta(x_t, t)\Big), \sigma_t^2 I\Big), \tag{3}$$

where the noise predictor $\epsilon_\theta(x_t, t)$ is trained using the simple denoising autoencoder objective:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_t, x_0, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]. \tag{4}$$

Following DDPM, one can set $\sigma_t^2 = \beta_t$ for simplicity. Meanwhile, improved DDPM (iDDPM) [44] shows that performance can be improved by jointly learning the variance $\Sigma_\theta(x_t, t)$, which is parameterized as an interpolation between β_t and $\tilde{\beta}_t$ in the log domain:

$$\log \Sigma_\theta(x_t, t) = v \log \beta_t + (1-v)\, \log \tilde{\beta}_t, \tag{5}$$

where $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, and v denotes the interpolation weight predicted by the model in a dimension-wise manner. In this work, we adopt the iDDPM framework for both training and sampling, following the same design choice as DiT [46].

B. Hyperparameters and implementation details

We design Fully Convolutional Diffusion Models (FCDMs) at multiple scales, aligned by parameter counts with DiT. Thanks to their easy scalability, we adjust only two hyperparameters, L and C, to obtain these variants. Notably, when compared to DiT in terms of FLOPs, our FCDMs require only 50.8% to 59.9% of those consumed by DiT, demonstrating the state-of-the-art computational efficiency of our design. Our implementation is based on the original DiT codebase [46]. Detailed configurations of these hyperparameters, along with additional implementation details, are provided in Table 7. For the latent space, we adopt an off-the-shelf pre-trained variational autoencoder (VAE) [29, 30, 49] with a downsampling factor of 8. Accordingly, an input RGB image of shape $256\times256\times3$ is encoded to a latent tensor of $32\times32\times4$. All diffusion training operates in this latent space, and latents are decoded back to pixels by the VAE decoder.

Resolution	FCDM-S 256×256	FCDM-B 256×256	FCDM-L 256×256	FCDM-XL 256×256	FCDM-XL 512×512
Architecture					
Input dim.	32×32×4	32×32×4	32×32×4	32×32×4	64×64×4
Num. blocks (L)	2	2	2	3	3
Hidden channels (C)	128	256	512	512	512
1×1 conv. expansion ratio (r)	3	3	3	3	3
Training					
Training iteration	400K	400K	400K	2M	1M
Global batch size	256	256	256	256	256
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Learning rate schedule	constant	constant	constant	constant	constant
(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Weight decay	0	0	0	0	0
Numerical precision	fp32	fp32	fp32	fp32	fp32
Data augmentation	random flip	random flip	random flip	random flip	random flip
Sampling					
Sampler	iDDPM	iDDPM	iDDPM	iDDPM	iDDPM
Sampling steps	250	250	250	250	250

Table 7. **Hyperparameter setup of FCDM model scales.** For all scales of FCDM, we adopt the same experimental settings as DiT.

Computing resources. Thanks to the superior efficiency of our fully convolutional architecture, we are able to train ImageNet models at 256×256 resolution using *consumer*-grade GPUs such as the NVIDIA RTX 4090 (24GB). For 256×256, we train FCDM-XL with a batch size of 256 on 4× RTX 4090 GPUs, achieving a training throughput of approximately 0.9 steps/s with gradient checkpointing enabled. We also confirm that the same batch size 256 fits on a single NVIDIA A100 40GB GPU, further demonstrating the memory efficiency of our design. For 512×512, we use 4× NVIDIA H100 80GB GPUs and obtain a throughput of about 0.7 steps/s (also with gradient checkpointing) with the same batch size.

C. Evaluation metrics

For evaluation, we follow the setup of ADM [9] and use the same reference batches provided in their official implementation. Specifically, we generate 50K samples and compute the metrics using OpenAI's official TensorFlow evaluation toolkit. All evaluations are conducted on NVIDIA RTX 4090 or NVIDIA H100 GPUs, except for certain reported numbers that are taken from prior work.

The following gives a concise description of the evaluation metrics used in our experiments.

- **FID** [16] measures the distance between the feature distributions of real and generated images. It is computed using the Inception-v3 network [59], under the assumption that both feature distributions follow multivariate Gaussian distributions.
- **sFID** [43] computes FID using spatial feature maps from intermediate layers of Inception-v3, thereby better capturing the spatial structure of generated images.
- **IS** [53] evaluates only generated images using the Inception-v3 network. It assigns higher scores when the images are classifiable with high confidence (sharp and meaningful) and when the set of generated images is diverse across different categories.
- **Precision and Recall** [32] measure realism and diversity in feature space. Precision reflects the fraction of generated images that look realistic, while recall reflects how much of the real data distribution is covered by the generated samples.

We additionally report computational efficiency. FLOPs are computed using `torchprofile`, and throughput is evaluated under the sampling configurations of DiT [46] with a batch size of 64. FlashAttention-2 [8] and Flash Linear Attention [66] are activated in DiT and DiG, respectively.

D. Baseline models

The following summarizes the key ideas of the diffusion baselines used for the evaluation.

- **ADM** [9] improves hybrid U-Net architecture for diffusion models and introduces classifier guidance, which enables a trade-off between sample quality and diversity.
- **VDM++** [27] enhances training efficiency by proposing a simple adaptive noise schedule for diffusion models.
- **Simple diffusion** [20] proposes a diffusion model for high-resolution image generation by carefully redesigning the noise schedule and model architecture.
- **CDM** [19] adopts a cascaded framework in which a base model first generates a low-resolution image, and subsequent super-resolution diffusion models progressively refine it to higher fidelity.
- **LDM** [49] proposes latent diffusion models that operate in a compressed latent space, greatly improving training efficiency while retaining high generation quality.
- **U-ViT** [3] adapts Vision Transformers for latent diffusion by introducing long skip connections similar to those in U-Net.
- **MaskDiT** [70] proposes an asymmetric encoder–decoder architecture for diffusion transformers, trained with an auxiliary mask reconstruction task to improve efficiency.
- **SD-DiT** [72] reframes the mask modeling of MaskDiT as a self-supervised discrimination objective.
- **DiT** [46] replaces the hybrid U-Net architecture with a fully transformer-based backbone, introduces AdaIN-zero conditioning to stabilize training, and shows that diffusion transformers scale effectively.
- **SiT** [42] reformulates DiT training by transitioning from discrete diffusion to continuous flow matching.
- **DiG** [71] integrates Gated Linear Attention [66], enabling sub-quadratic complexity efficiency of diffusion transformers.
- **DiC** [62] re-examines purely convolutional denoisers by scaling standard 3×3 convolutional blocks in a U-Net design, introducing sparse skip connections.
- **DiCo** [1] proposes a 3×3 separable convolutional block in a U-Net design, introducing compact channel attention to activate more informative channels.

E. Additional Scaling Results

As shown in Figure 5, we clearly demonstrate the scalability of FCDM in terms of FID. We also observe consistent scalability across other metrics, including sFID, Inception Score, Precision, and Recall, as reported in Table 8.

In addition, we trained FCDMs using the original SiT implementation [42] to examine whether our proposed design also exhibits scalability under this framework. Following the original implementation details, we trained the model with the flow-matching objective [37, 42]. We used AdamW with a constant learning rate of 1×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and no weight decay. For sampling, we employed the Euler–Maruyama SDE sampler with 250 steps, setting the final step size to 0.04.

As shown in Table 9, we again observe clear scalability when training the proposed network with the flow-matching objective. Interestingly, under our framework, the flow-matching objective yields better performance at the Small (S) scale, while showing slightly worse results at the Base (B) through XLarge (XL) scales. Nevertheless, these results confirm that the proposed architecture possesses generalized scalability beyond a specific training objective.

Model	FLOPs (G)	Training Steps	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
FCDM-S	3.10	50K	103.93	15.03	12.01	0.3013	0.3513
		100K	77.17	12.15	17.11	0.3809	0.4473
		150K	66.85	11.02	20.68	0.4136	0.5106
		200K	60.33	10.70	23.71	0.4396	0.5537
		250K	56.08	10.54	26.30	0.4568	0.5609
		300K	53.01	10.59	28.10	0.4687	0.5806
		350K	50.44	10.29	30.08	0.4757	0.5729
		400K	**48.53**	**10.12**	**31.64**	**0.4836**	**0.5840**
FCDM-B	12.20	50K	76.61	9.75	16.45	0.3797	0.4569
		100K	50.83	8.24	26.90	0.4854	0.5543
		150K	40.60	7.53	35.35	0.5268	0.5894
		200K	35.00	7.22	42.42	0.5525	0.5981
		250K	31.77	7.11	47.22	0.5665	0.6117
		300K	29.26	7.03	51.62	0.5705	0.6141
		350K	27.34	6.94	55.10	0.5855	0.6127
		400K	**26.21**	**6.86**	**58.04**	**0.5908**	**0.6112**
FCDM-L	48.30	50K	55.15	8.33	22.74	0.4655	0.5403
		100K	32.03	7.10	43.20	0.5791	0.5857
		150K	23.88	6.47	58.51	0.6106	0.6052
		200K	19.97	6.17	69.19	0.6312	0.6128
		250K	17.33	5.99	77.79	0.6427	0.6233
		300K	15.98	5.82	84.28	0.6477	0.6245
		350K	14.95	5.82	87.55	0.6527	0.6257
		400K	**13.83**	**5.65**	**93.31**	**0.6612**	**0.6218**
FCDM-XL	64.60	50K	51.00	8.31	24.37	0.4940	0.5475
		100K	27.23	6.86	49.52	0.6108	0.5824
		150K	19.25	6.15	68.62	0.6492	0.5995
		200K	15.54	5.95	81.40	0.6690	0.6051
		250K	13.50	5.74	91.74	0.6785	0.6117
		300K	12.31	5.64	98.58	0.6829	0.6192
		350K	11.19	5.54	104.86	0.6914	0.6227
		400K	**10.72**	**5.47**	**108.04**	**0.6864**	**0.6273**

Table 8. **Performance of FCDMs across scales and training steps on ImageNet 256×256 (Diffusion).** Scaling FCDMs consistently leads to improved generative performance when trained with the diffusion objective.

Model	FLOPs (G)	Training Steps	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
FCDM-S	3.10	50K	103.10	13.10	12.23	0.2863	0.3111
		100K	76.95	11.12	16.96	0.3826	0.4547
		150K	66.97	10.18	20.56	0.4228	0.4953
		200K	60.53	9.62	23.90	0.4428	0.5372
		250K	55.94	9.53	26.16	0.4670	0.5515
		300K	52.43	9.22	28.78	0.4787	0.5632
		350K	49.76	9.05	31.06	0.4866	0.5730
		400K	**47.84**	**8.91**	**32.89**	**0.4944**	**0.5749**
FCDM-B	12.20	50K	80.10	18.48	15.46	0.3286	0.4072
		100K	52.23	8.34	25.95	0.4891	0.5450
		150K	42.58	7.85	33.83	0.5346	0.5780
		200K	37.01	7.51	40.72	0.5554	0.5819
		250K	33.14	7.24	46.39	0.5728	0.5855
		300K	30.16	7.07	51.60	0.5883	0.5953
		350K	28.40	6.99	55.07	0.5941	0.6066
		400K	**26.61**	**6.85**	**58.51**	**0.6050**	**0.6017**
FCDM-L	48.30	50K	57.32	15.82	21.61	0.4467	0.4872
		100K	33.71	7.74	40.77	0.5852	0.5615
		150K	26.10	6.94	54.79	0.6232	0.5865
		200K	21.91	6.63	65.12	0.6429	0.5909
		250K	19.39	6.47	73.55	0.6557	0.5940
		300K	17.59	6.31	80.47	0.6650	0.6075
		350K	16.27	6.18	85.62	0.6681	0.6064
		400K	**15.30**	**6.17**	**90.09**	**0.6751**	**0.6126**
FCDM-XL	64.60	50K	51.21	11.89	24.21	0.5010	0.5006
		100K	29.37	7.34	46.27	0.6172	0.5680
		150K	22.07	6.77	63.02	0.6572	0.5870
		200K	17.98	6.34	75.71	0.6747	0.5909
		250K	15.72	6.24	85.30	0.6853	0.5991
		300K	14.16	6.07	92.79	0.6952	0.6040
		350K	13.06	5.97	98.22	0.6976	0.6072
		400K	**12.11**	**5.96**	**103.07**	**0.7030**	**0.6070**

Table 9. **Performance of FCDMs across scales and training steps on ImageNet 256×256 (Flow-Matching).** Scaling FCDMs also demonstrates consistent improvements in generative performance when trained with the flow-matching objective.

F. Frequency-based analysis

To better highlight the differences between our fully convolutional diffusion model (FCDM) and the transformer-based DiT, we examine the evolution of the spectral energy, defined as the sum of the log-magnitude spectrum of the predicted noise, over the course of the diffusion process (using models trained for 400K iterations at 512×512 resolution). For each predicted noise sample, we compute the 2D Fourier transform, take the magnitude spectrum, and apply a logarithmic scaling $\log(1 + F)$ to compress the dynamic range. We then define the total spectral energy as the sum of all values in this log-magnitude spectrum, which reflects the overall distribution of frequency components. Figure 8 presents the total spectral energy, averaged over 128 validation samples, calculated at each of the 1,000 diffusion timesteps.

Across all diffusion steps, FCDM consistently exhibits higher spectral energy than DiT. This difference is most pronounced in the early-to-middle stages of the diffusion trajectory, where the model must simultaneously capture global structure and fine-grained detail. The elevated energy of FCDM indicates that its predicted noise retains stronger high-frequency components, which can be associated with sharper textures, edges, and local structures. By contrast, DiT produces lower spectral energy, suggesting smoother predictions with fewer high-frequency details. While this observation may provide a partial explanation for the performance gap between FCDM and DiT, further theoretical analysis is required.

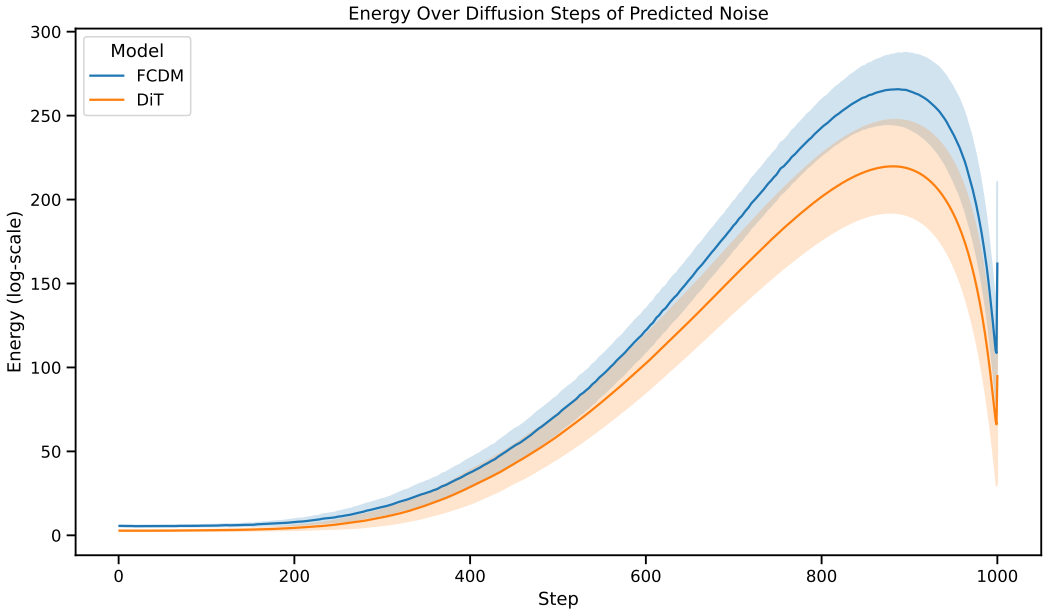


Figure 8. **Spectral energy of predicted noise across diffusion steps.** FCDM consistently exhibits higher spectral energy than DiT across the entire diffusion process, suggesting potential for better preservation of high-frequency components.

G. Analysis of additional architectural variants

This section provides further details on the ablations in the main manuscript and introduces additional architectural ablation results. Table 6 presents architectural ablations using the Large (L) model on ImageNet at 256×256. We analyze the effects of specific architectural elements, including kernel size, Global Response Normalization (GRN), the inverted bottleneck (channel expansion), the feedforward module, and FCDM blocks. In addition, we evaluate various other design choices to further investigate their impact on performance and efficiency. Interestingly, although ConvNeXt [40, 64] was developed for a different task, we observe similar ablation trends in our experiments.

Effect of Kernel Size. Vision Transformers employ non-local self-attention, enabling each layer to access a global receptive field. In contrast, ConvNets traditionally rely on stacking small 3×3 convolutions (popularized by VGGNet [54]), which are efficient on modern GPUs [34]. Our experiments show that reducing the kernel size from the default 7×7 consistently degrades performance, with FID increasing from 19.97 to 20.48 and 21.28. This suggests that larger kernels provide a larger effective receptive field, allowing the model to capture broader context.

Additionally, we explore kernel sizes larger than the default 7×7. Interestingly, as shown in Table 10, performance degrades for kernel sizes beyond 7×7, which may slow down convergence speed and leave the models under-trained within our 200K iteration ablation studies.

Taken together, these results suggest that the 7×7 kernel provides the best balance between expressiveness and convergence speed for our architecture.

Model Configuration	FLOPs (G) ↓	FID ↓	IS ↑
FCDM-L (Default: 7×7 DWConv)	48.3	**19.97**	**69.19**
7×7 → 9×9 DWConv	48.4	23.85	59.68
7×7 → 11×11 DWConv	48.6	23.93	60.18

Table 10. **Ablation study on larger kernels.** We analyze effects of larger kernel sizes. Training iterations are fixed to 200K.

Effect of Applying DiCo Design Choices. Following DiCo [1], we investigate whether the architectural design choices from DiCo improve performance in our framework. For a fair comparison, we adjust the channel dimensions (C) to match the total FLOPs across variants.

First, we replace the GRN layer with CCA [1], as both modules are designed to enhance channel diversity. As shown in Table 6, using CCA degrades performance compared to GRN, indicating that GRN is a more effective channel enhancement module in our architecture. Second, we remove channel expansion to align with DiCo, which does not apply channel expansion in its convolutional blocks. As shown in Table 6, disabling channel expansion leads to a clear drop in performance, demonstrating that channel expansion is a crucial component of our architecture for maintaining generative capacity. Lastly, we add the feedforward module after the convolutional block to follow the DiCo design, whereas our baseline does not include this component. As shown in Table 6, adding the feedforward module further degrades performance, suggesting that our design is better suited for efficient diffusion modeling.

Overall, these results indicate that our design choices consistently lead to superior performance at matched FLOPs, validating the effectiveness of our architecture.

Effect of FCDM Blocks. We replaced FCDM blocks with ResNet blocks [15] using standard 3×3 convolutions. To match FLOPs, the hidden channels were reduced from 512 to 336, given the higher computational cost of standard convolutions compared to separable convolutions. This substitution results in a substantial degradation, with FID increasing from 19.97 to 31.14, indicating that the FCDM block is better suited for this task than the ResNet block.

Effect of Autoencoders. Since FCDM operates in latent space, we tested whether performance persists under different VAEs. As shown in Table 11, FCDM consistently outperforms DiT under both SD-VAE [49] and EQ-VAE [30]. Similar to DiT, our model performs best with EQ-VAE, improving further over SD-VAE. These findings suggest that techniques originally proposed to enhance DiT (e.g., stronger VAEs) transfer equally well to FCDM, indicating the potential for further performance improvements.

Model Configuration	Training iterations	FLOPs (G) ↓	FID ↓
SD-VAE			
DiT-XL/2	400K	118.6	19.47
FCDM-XL	400K	**64.6**	**11.57**
EQ-VAE			
DiT-XL/2	400K	118.6	14.50
FCDM-XL	400K	**64.6**	**10.72**

Table 11. **Ablation study on autoencoders.** Across different latent spaces (SD-VAE and EQ-VAE), FCDM consistently outperforms DiT.

Effect of Isotropic Architecture. To further analyze our architecture, we evaluated an isotropic variant of our model. As shown in Table 12, despite using only ∼48% of the parameters of DiT-B/2 [46], our isotropic variant achieves a lower FID given the same number of training steps. This again validates the effectiveness and efficiency of our fully convolutional block design.

Model Configuration	Params (M)	FID ↓
DiT-B/2	130	55.00
FCDM (Iso.)	**62 (∼48%)**	**41.15**

Table 12. **Ablation study on isotropic architecture.** Comparison of our isotropic variant against DiT-B/2 [46]. All models are trained on ImageNet 256×256 for 200k iterations under identical settings.

Effect of Replacing Convolution with Local Attention. We conduct an ablation study to examine the impact of replacing depthwise convolution with local self-attention in our architecture. Specifically, we employ Neighborhood Attention (NA) [14], which implements sliding-window local attention with efficient C++ and CUDA kernels. For a fair comparison, we use a 7×7 attention window to match the receptive field of our 7×7 depthwise convolution. We also adjust the channel dimensions to ensure matched FLOPs across variants.

As shown in Table 13, replacing depthwise convolution with NA results in a significant performance drop in terms of both FID and IS. Moreover, the throughput is substantially reduced, indicating that local self-attention is considerably less efficient in practice.

These results suggest that depthwise convolution is more effective than local self-attention in terms of both performance and efficiency in our architecture, making it a better choice for efficient diffusion modeling.

Model Configuration	FLOPs (G) ↓	TP (it/s) ↑	FID ↓	IS ↑
FCDM-L (Default: DWConv)	48.3	**381.3**	**19.97**	**69.19**
DWConv → NA* [14]	45.9	122.8	29.81	50.92

Table 13. **Ablation study on replacing depthwise convolution with local self-attention.** Neighborhood Attention (NA) [14] with a 7×7 window is used as a replacement for depthwise convolution. All models are trained for 200K iterations at matched FLOPs. $*$ indicates that C is adjusted to match FLOPs to ensure a fair comparison.

Effect of Asymmetric Encoder–Decoder Allocation. Following [21], we investigated whether an asymmetric allocation of compute between the encoder and decoder could outperform the symmetric setup. Intuitively, assigning more compute to the decoder appears advantageous, since upsampling from low to high resolution is more demanding and additionally requires processing skip connections. As shown in Table 14, an asymmetric design slightly improves FID for the Large (L) model (19.97 → 19.55). However, for the XLarge (XL) model, the asymmetric setup performs on par with the symmetric variant. While asymmetric encoder–decoder architectures remain an interesting direction, we adopt the symmetric setup for its simplicity and more straightforward scalability.

Effect of Block Scaling across U-Net Levels. We investigated how block scaling across U-Net levels affects model behavior. We compared (1) a default scaling setup, where deeper levels have more blocks, and (2) a uniform setup, where all levels

Model Configuration	Hidden channel C	Depths	Params (M)	FLOPs (G) ↓	TP (it/s) ↑	FID ↓
Asymmetric U-Net Ablations						
FCDM-L (Default: Sym. U-Net)	512	[2, 4, 8, 4, 2]	504.5	48.3	381.3	19.97
Asym. U-Net	512	[2, 3, 8, 5, 2]	504.5	48.3	381.3	**19.55**
FCDM-XL (Default: Sym. U-Net)	512	[3, 6, 12, 6, 3]	698.8	64.6	272.7	**15.54**
Asym. U-Net	512	[3, 5, 12, 7, 3]	698.8	64.6	272.7	**15.54**
Asym. U-Net	512	[3, 3, 12, 9, 3]	698.8	64.6	272.7	15.55

Table 14. **Ablation study on asymmetric block allocation.** All models trained for 200K iterations under identical training settings.

use the same number of blocks. To ensure fairness, we kept the total number of blocks fixed and matched the total parameter count by adjusting channel sizes. As shown in Table 15, the uniform setup yields better FID for the Large (L) model (19.97 $\rightarrow$ 17.63). However, it significantly degrades efficiency: FLOPs increase from 48.3G to 62.8G, reaching a level similar to the XLarge (XL) model with the scaling setup (62.8G vs. 64.6G), while still yielding worse FID (17.63 vs. 15.54). Given this trade-off, we adopt the default scaling setup in our architecture, since efficiency is a core design goal of this work.

Model Configuration	Hidden channel C	Depths	Params (M)	FLOPs (G) ↓	TP (it/s) ↑	FID ↓
FCDM-L (Default: Scaling)	512	[2, 4, 8, 4, 2]	504.5	**48.3**	**381.3**	19.97
Uniform	600	[4, 4, 4, 4, 4]	496.6	62.8	261.6	17.63
FCDM-XL (Default: Scaling)	512	[3, 6, 12, 6, 3]	698.8	64.6	272.7	**15.54**

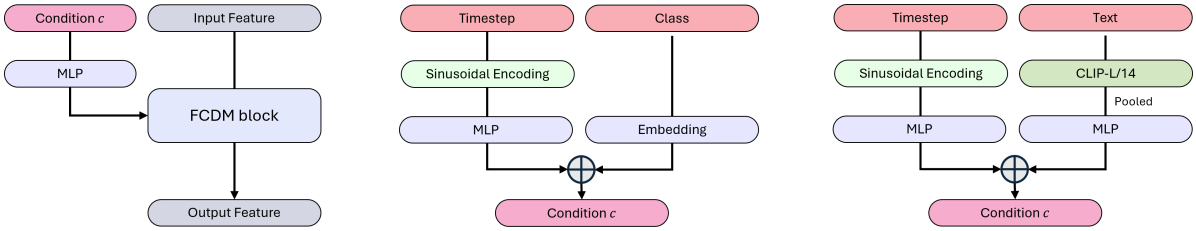
Table 15. **Ablation study on block scaling strategies.** All models are trained on ImageNet 256×256 for 200k iterations under identical settings. The channel size is adjusted to match the total number of parameters.

H. Text-to-image generation experiment

Originally, the FCDM block (Figure 9 (a)) is designed for class-conditioned generation, where the conditioning vector c is derived from the diffusion timestep and the class label using the conditioning module illustrated in Figure 9 (b). To adapt FCDM for text-to-image generation, we modify this conditioning module by replacing the class embedding layer with a CLIP text encoder [48] followed by MLP layers, as shown in Figure 9 (c). This design enables the network to be conditioned on both the diffusion timestep and the pooled text representation [11, 47]. By modifying only the conditioning module while preserving the rest of the architecture, we can reuse the same FCDM backbone for text-to-image generation.

We train FCDM-XL with the text-conditioning module from scratch on the MS-COCO [36] training split and evaluate it on the validation split. The model is trained for 100K iterations with a batch size of 256, using the CLIP text encoder to compute pooled text embeddings. Apart from the modified conditioning module, all other training hyperparameters and settings are kept identical to those of the original FCDM configuration.

As shown in Figure 10, FCDM successfully generates images corresponding to the given text descriptions, even when trained for only 100K iterations using pooled text embeddings alone. These findings indicate that text conditioning can be effectively incorporated into FCDM. Nevertheless, extending FCDM to support joint conditioning on full text embeddings, as in MMDiT [11], represents an important direction for future work toward learning richer representations and improving generation performance.



(a) FCDM block (b) Class conditioning (c) Text conditioning

Figure 9. **Conditioning modules for class and text in the FCDM architecture.** (a) FCDM block with conditioning vector c, (b) Conditioning module for class conditioning, (c) Conditioning module for text conditioning incorporating the CLIP text encoder.

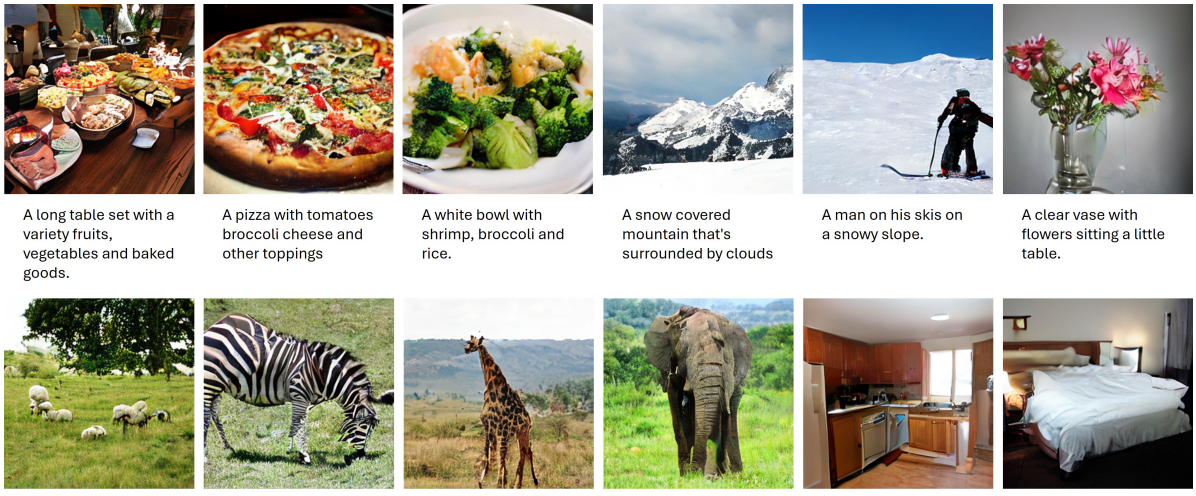


Figure 10. **Qualitative results on text-to-image generation (MS-COCO).** We use classifier-free guidance with scale = 4.0.

I. More quantitative results

While we position our architecture as a state-of-the-art model in terms of *efficiency*, it is also important to assess how its *performance* relates to other diffusion models. To this end, building upon the results in Table 3 and Table 4, we extend our evaluation to additional baselines [12, 67], focusing on both efficiency and performance.

As presented in Table 16 and Table 17, our findings indicate that FCDM-XL achieves generation quality closely aligned with FasterDiT [67]. Despite this similarity in performance, our architecture provides substantial efficiency benefits: it achieves 3.3× faster inference throughput and requires 1.8× fewer total training FLOPs.

These results reaffirm that our architecture remains highly competitive, offering competitive performance while maintaining state-of-the-art efficiency. We hope that our architecture will serve as a practical and efficient backbone for future research.

Model	Architecture Type	FLOPs (G) ↓	Throughput (it/s) ↑	FID ↓	IS ↑	Precision ↑	Recall ↑
MDT-XL/2 (400K) [12]	Transformer	118.7	83.9	16.42	-	-	-
FasterDiT-XL/2 (400K) [67]	Transformer	118.6	80.5	11.90	-	-	-
FCDM-XL (400K)	Conv	**64.6**	**272.7**	**10.72**	108.0	0.69	0.63
MDT-XL/2 (1.3M)	Transformer	118.7	83.9	9.60	-	-	-
FasterDiT-XL/2 (1M)	Transformer	118.6	80.5	8.72	121.17	0.68	**0.67**
FCDM-XL (1M)	Conv	**64.6**	**272.7**	**7.91**	**135.55**	**0.71**	0.64

Table 16. **Additional comparisons on ImageNet 256×256 without guidance.** We report efficiency metrics (FLOPs, throughput) and performance metrics (FID, IS, Precision, Recall) using 50K samples. The best results are shown in **bold**.

Model	Training epochs	FLOPs (G) ↓	Training FLOPs (Z) ↓	Throughput (it/s) ↑	FID ↓	IS ↑	Precision ↑	Recall ↑
MDT-XL/2 [12]	500	118.7	0.228	83.9	2.15	249.3	**0.82**	0.58
FasterDiT-XL/2 [67]	400	118.6	0.182	80.5	**2.03**	270.0	0.81	**0.60**
FCDM-XL	400	**64.6**	**0.099**	**272.7**	**2.03**	**285.7**	0.81	0.59

Table 17. **Additional comparisons on ImageNet 256×256 with guidance.** We report efficiency metrics (FLOPs, training FLOPs, throughput) and performance metrics (FID, IS, Precision, Recall) using 50K samples. The best results are shown in **bold**.

J. More qualitative results

Class label = “husky” (250)

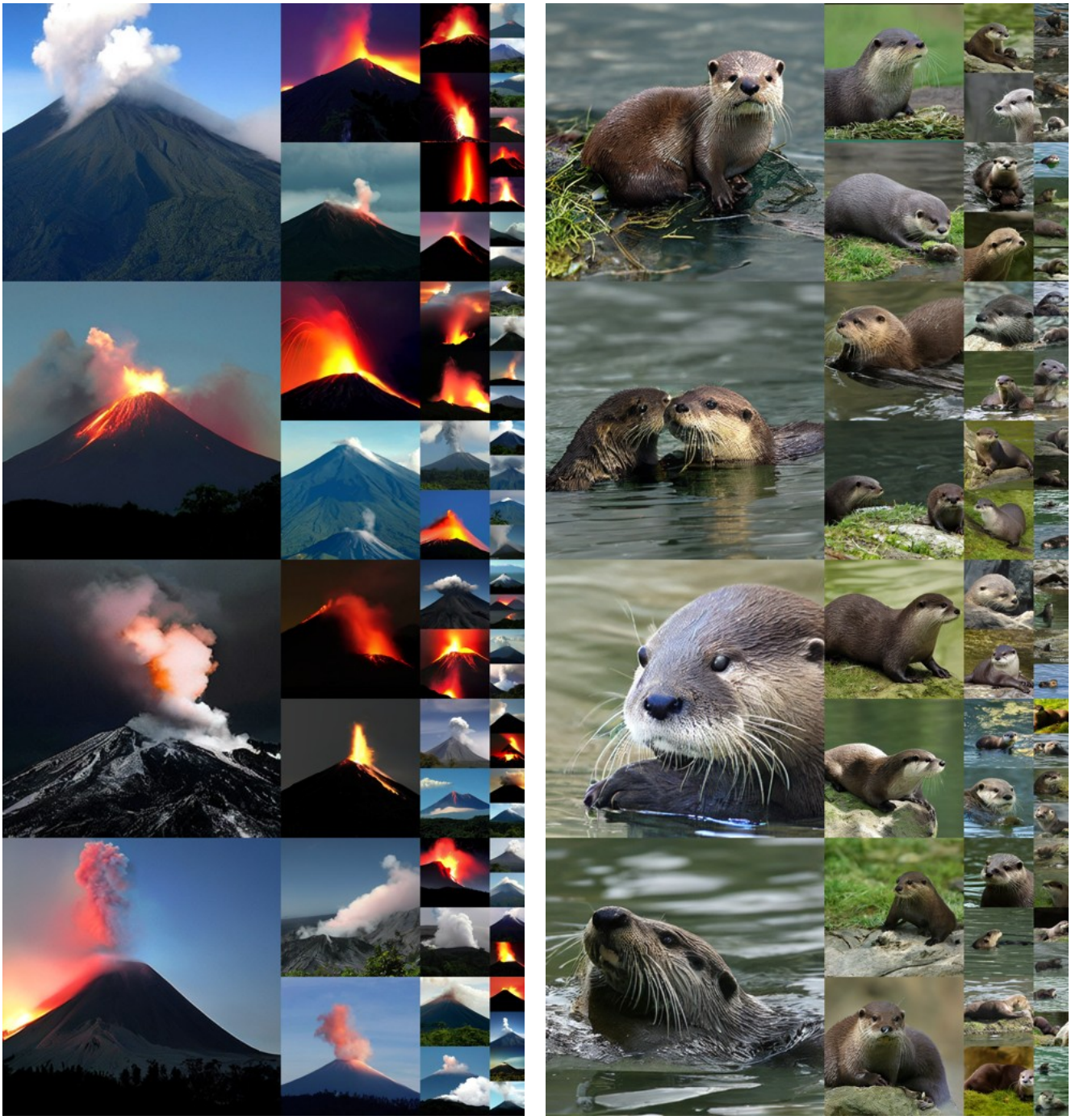
Class label = “sulphur-crested cockatoo” (89)

Figure 11. **Uncurated 512×512 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “lion” (291)

Class label = “arctic wolf” (270)

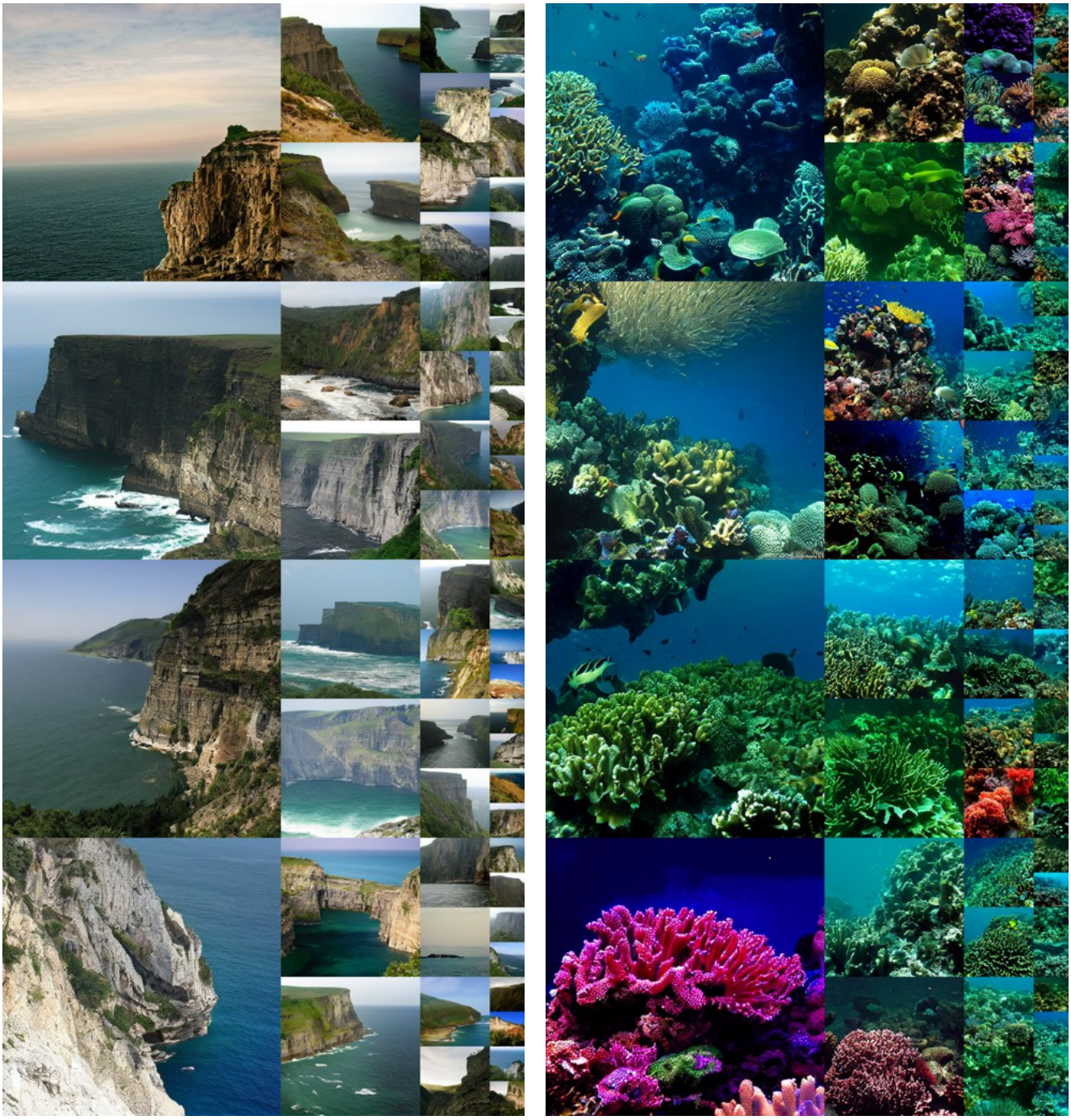
Figure 12. **Uncurated 512×512 FCDM-XL samples.** Classifier-free guidance scale = 4.0



Class label = “volcano” (980)

Class label = “otter” (360)

Figure 13. **Uncurated 512×512 FCDM-XL samples.** Classifier-free guidance scale = 4.0



Class label = “cliff drop-off” (972)

Class label = “coral reef” (973)

Figure 14. **Uncurated 512×512 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “panda” (388)

Class label = “red panda” (387)

Figure 15. **Uncurated 512×512 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “macaw” (88)

Class label = “arctic fox” (279)

Figure 16. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “golden retriever” (207)

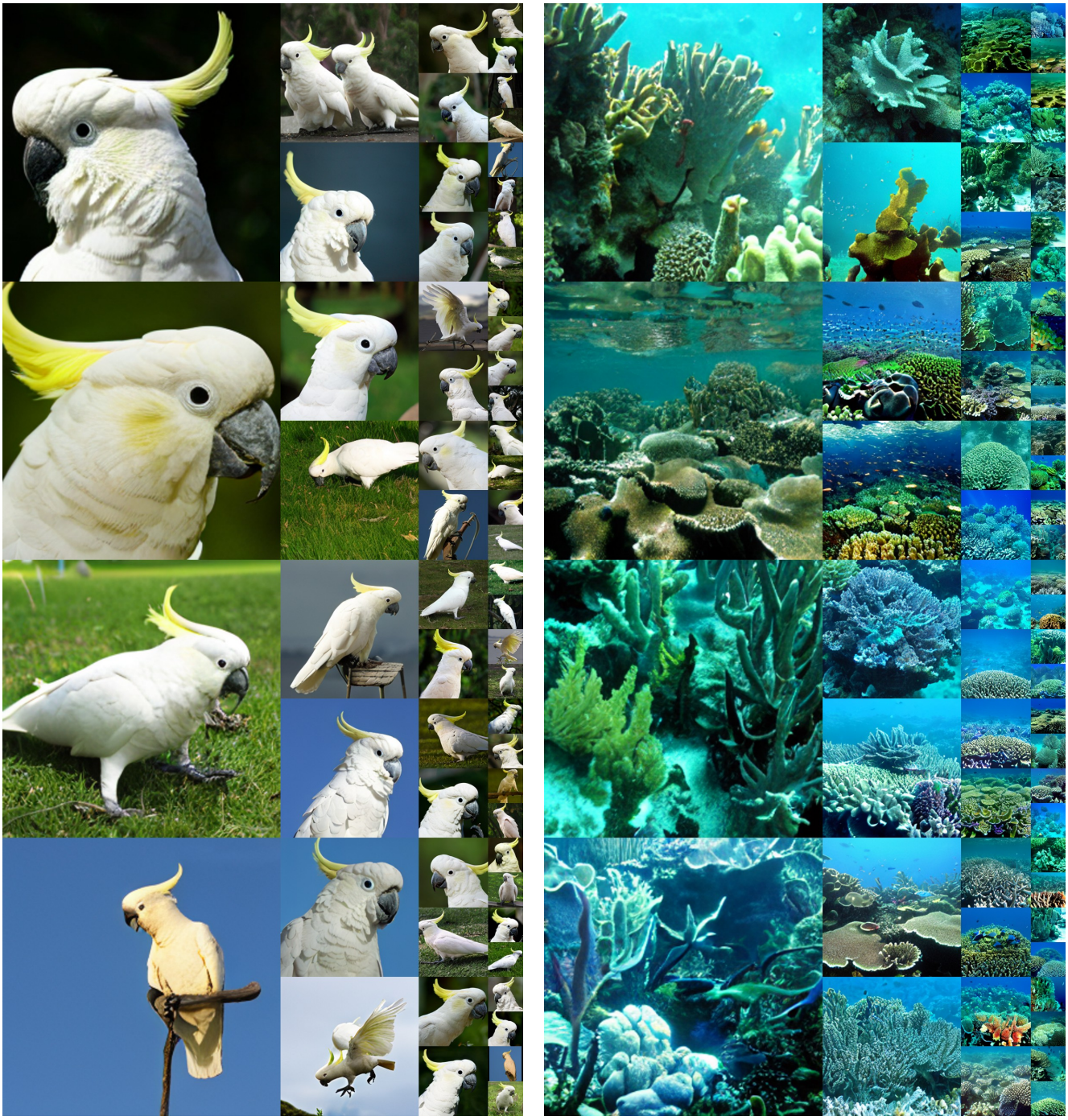
Class label = “loggerhead sea turtle” (33)

Figure 17. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “lake shore” (975)

Class label = “space shuttle” (812)

Figure 18. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0



Class label = “sulphur-crested cockatoo” (89)

Class label = “coral reef” (973)

Figure 19. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “otter” (360)

Class label = “geyser” (974)

Figure 20. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0

Class label = “cliff drop-off” (972)

Class label = “ice cream” (928)

Figure 21. **Uncurated 256×256 FCDM-XL samples.** Classifier-free guidance scale = 4.0