Diffusion Probabilistic Model Made Slim
-Supplementary Material-

Xingyi Yang[1] Daquan Zhou[2] Jiashi Feng[2] Xinchao Wang[1]
National University of Singapore[1] ByteDance Inc.[2]
xyang@u.nus.edu, {daquanzhou, jshfeng}@bytedance.com, xinchao@nus.edu.sg

This document presents supplementary materials that were not included in the main manuscript due to page limitations. Firstly, we present the complete solution deviation for the linear diffusion and the full architecture for Spectral Diffusion (SD). We also perform additional experiments to confirm the effectiveness of our proposed SD. Furthermore, we provide details on our implementation, including dataset settings, evaluation metrics, and hyper-parameter settings.

1. Linear Diffusion as Wiener Filters

Let us consider an additive white noise model given by $\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}$, where $\mathbf{x}$ is the clean signal and $\boldsymbol{\epsilon}$ is a Gaussian white noise with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. The optimal denoiser aims to estimate the recovered signal that can best recover the original clean signal $\mathbf{x}$. To align with the DPM objective, we adopt a different notation that recovers the noise $\boldsymbol{\epsilon}$ as an approximation $\boldsymbol{\epsilon} \approx \mathbf{s}(\mathbf{x}'; \boldsymbol{\theta})$.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta},\mathbf{x}} \mathbb{E}_{\boldsymbol{\theta}}[||\mathbf{s}(\mathbf{x} + \boldsymbol{\epsilon}; \boldsymbol{\theta}) - \boldsymbol{\epsilon}||_2^2] \tag{1}$$

Note that both predicting the signal or the noise does not make any difference in the additive model, since the reconstructed image is $\mathbf{x}' - \mathbf{s}(\mathbf{x}'; \boldsymbol{\theta})$.

To simplify the model and find a closed-form solution, we investigate a linear filter case in which $\mathbf{s}(\mathbf{x}'; \boldsymbol{\theta}) = h * \mathbf{x} + h * \boldsymbol{\epsilon}$, where $*$ denotes the convolution operator. Assuming that $\mathbf{x}$ is wide-sense stationary and $\boldsymbol{\epsilon}$ is white noise with a variance of σ^2, the optimal filter is known as the **Wiener filter** [5], and can be derived exactly

$$\mathcal{H}^*(f) = \frac{\sigma^2}{|\mathcal{X}(f)|^2 + \sigma^2} \tag{2}$$

where $|\mathcal{X}(f)|^2$ is the power spectrum of the signal $\mathbf{x}$ and $\mathcal{H}^*(f)$ is the frequency response of h^*.

Proof: Assume that $\mathcal{F}$ is the Fourier transformation. Since the convolution in the time domain is equivalent to the multiplication in the Fourier domain

$$\mathcal{H}(\mathcal{X} + \mathcal{F}[\epsilon]) = \mathcal{F}[h * (\mathbf{x} + \boldsymbol{\epsilon})] \tag{3}$$

The estimation error signal $E(f)$ is defined in Fourier domain as

$$J(f) = \mathcal{H}(\mathcal{X} + \mathcal{F}[\epsilon]) - \mathcal{F}[\epsilon] \tag{4}$$

We thus minimize the mean square error $E(f)$, which is, in essence, identical to Equation 1

$$\min_h \mathbb{E}[|J(f)|_2^2] \tag{5}$$

$$= \min_h \mathbb{E}[|\mathcal{H}(\mathcal{X} + \mathcal{F}[\epsilon]) - \mathcal{F}[\epsilon]|_2^2] \tag{6}$$

$$= \min_h \mathbb{E}[\big(\mathcal{H}(\mathcal{X} + \mathcal{F}[\epsilon]) - \mathcal{F}[\epsilon]\big)^* \big(\mathcal{H}(\mathcal{X} + \mathcal{F}[\epsilon]) - \mathcal{F}[\epsilon]\big)] \tag{7}$$

The symbol $*$ denotes the complex conjugate. To obtain the least mean square error filter,we set the complex derivative of Equation 7 with respect to filter $\mathcal{H}(f)$ to zero

$$\frac{\partial \mathbb{E}[|J(f)|_2^2]}{\partial \mathcal{H}(f)} = 2\mathcal{H}(f)|(\mathcal{X} + \mathcal{F}[\epsilon])|^2 - 2|\mathcal{F}[\epsilon]|^2 = 0 \tag{8}$$

where $|(\mathcal{X} + \mathcal{F}[\epsilon])|^2 = \mathbb{E}[(\mathcal{X} + \mathcal{F}[\epsilon])(\mathcal{X} + \mathcal{F}[\epsilon])^*]$ and $|\mathcal{F}[\epsilon]|^2 = E[\mathcal{F}[\epsilon]\mathcal{F}[\epsilon]^*]$ are the power spectrum of the input noisy image $\mathbf{x}'$ and the cross-power spectrum between $\mathbf{x}'$ and $\boldsymbol{\epsilon}$. Because $\boldsymbol{\epsilon}$ is probabilistically independent of $\mathbf{x}_0$, the second term boils down to $|\mathcal{F}[\epsilon]|^2$. Then, the solution is known as a Weiner solution

$$\mathcal{H}^*(f) = \frac{|\mathcal{F}[\epsilon]|^2}{|(\mathcal{X}(f) + \mathcal{F}[\epsilon])|^2} = \frac{\sigma^2}{|\mathcal{X}(f)|^2 + \sigma^2} \tag{9}$$

For white noise with known variance σ^2, $|\mathcal{F}[\epsilon]|^2$ is a constant of σ^2.

In DPMs' formulation, the signals and noises are scaled with $\bar{\alpha}$ such that $\mathbf{x}_t = \sqrt{\bar{\alpha}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}}\boldsymbol{\epsilon}$, where the linear optimal filter at step t could be written as

$$\mathcal{H}_t^*(f) = \frac{1}{\bar{\alpha}|\mathcal{X}_0(f)|^2 + 1 - \bar{\alpha}} \tag{10}$$

where $|\mathcal{X}_0(f)|^2$ is the power spectrum of $\mathbf{x}_0$ and $\mathcal{H}_t^*(f)$ is the frequency response of h_t^*.

Error Analysis for DPM. By plugging the solution back to the cost function in Eq. 4, we have the error as

$$|J^*(f)|^2 = |\mathcal{H}_t^*(f)(\mathcal{X} + \mathcal{F}[\epsilon]) - \mathcal{F}[\epsilon]|^2 \tag{11}$$

$$= |\mathcal{F}[\epsilon]|^2 \Big[1 - \frac{|\mathcal{F}[\epsilon]|^2}{|\mathcal{F}[\epsilon]|^2 + |\mathcal{X}(f)|^2}\Big] \tag{12}$$

$$= \frac{|\mathcal{X}(f)|^2 |\mathcal{F}[\epsilon]|^2}{|\mathcal{F}[\epsilon]|^2 + |\mathcal{X}(f)|^2} \tag{13}$$

$$= \frac{1}{1/\text{SNR} + \text{SNR}} \tag{14}$$

where SNR is the signal-to-noise ratio. For function $g(x) = \frac{1}{1/x+x}, x > 0$, it has a unique maximum at $x = 1$. When $x > 1$, $g(x)$ is decreasing; when $0 < x < 1$, $g(x)$ is increasing.

2. Improved DPMs through Frequency

In this section, we aim to describe several training techniques that have been observed to enhance the visual quality of DPMs. These techniques include cosine noise scheduling [4] and the Unet structure, and we demonstrate how they implicitly improve the generation of high-frequency components.

Cosine Scaling. Cosine scaling [4] is a noise scaling method for $\bar{\alpha}_t$ based on a cosine function. This method increases $\bar{\alpha}_t$ when t is small, resulting in more reverse steps for denoising nearly-clean images.

Our paper shows that DPMs initially reconstruct the low-frequency components of an image and gradually shift towards detail recovery during denoising. Consequently, cosine scaling prioritizes the restoration of high-frequency details by allocating more steps during the nearly-clean stage, compared to linear noise scaling.

UNet Architecture. Neural networks have a tendency to smooth out input information as it propagates through deep layers, which can be understood as a special case of Gaussian Process [3]. To counteract this effect and better preserve the textual and high-frequency details of images, Unet incorporates skip connections between shallow and deep layers, resulting in improved performance compared to plain encoder-decoder structures. As a result, using Unet has become a common practice when designing DPM architectures.

3. Additional Experiments

In this section, we provide more experiments to support our motivation and verify our `SD` models.

3.1. High-Frequency Loss and visual quality

Our paper is premised on the fundamental assumption that high-frequency components play a significant role in determining visual quality. We tested this assumption by measuring the FID score between real images and low-pass filtered images. We aimed to understand the impact of high-frequency absence on the FID score using a simple low-pass filter.

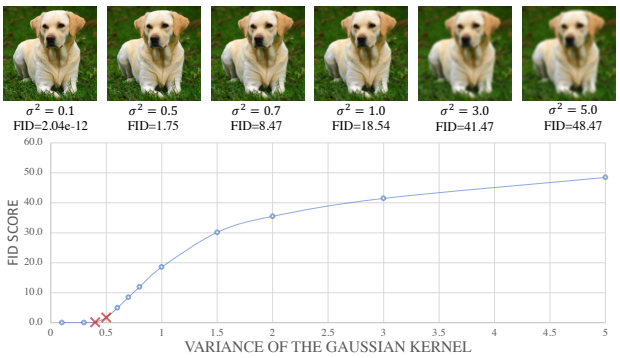

Figure 1. FID and σ^2 plot on FFHQ dataset

Specifically, we applied Gaussian filters with a kernel size of 9, zero mean, and $\sigma^2 = 0.1, 0.3, 0.4, 0.5, 0.7, 0.8, 0.9, 1.0, 1.5, 2.0, 3.0, 5.0$ to the 256×256 FFHQ dataset. For each σ^2, we calculated the FID score between 50k clean and 50k blurred images. As we increased σ^2, we removed more high-frequency components, resulting in heavily blurred images.

Figure 1 shows that a serious drop in FID score was observed with σ^2 around 0.5 (marked in red). Although the visual degradation with a small σ^2 was not apparent to human perception, the FID score significantly increased from 2e-12 to 18.54 when σ^2 increased from 0.1 to 1.0. This finding indicates that the FID score is highly sensitive to high-frequency recovery in images, supporting our motivation to enhance the high-frequency recovery of slim diffusion models to improve their performance.

3.2. Ablation Study on the Distillation Locations

For DPMs, we can distill knowledge from the teacher by selecting arbitrary outputs or feature positions. In this part, we apply distillation at multiple positions to identify the best option. Our candidate positions include the model's final output (`output`), feature map after the last up-sample operation (`feat-up1`), feature map after the second last up-sample module (`feat-up2`) and the feature map after the last down-sample layer (`feat-down4`). Those positions are formally defined in Sec. 4.4. Since the teacher model shares a similar structure as its student, we decide to distill the feature/output at the same location by default. We validate each option on FFHQ unconditional image generation task. We use the default distillation weight of $\lambda_s = \lambda_f = 0.1$.

We visualize the results in Figure 2. Our key observation is that, distilling the deep layers, especially the output layer, helps the most, while distilling the shallow layer of

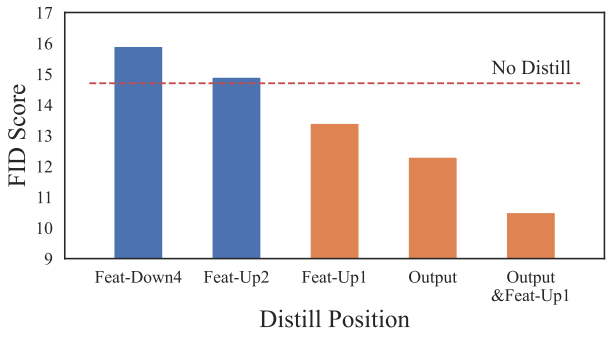

Figure 2. FID score with different distillation positions. Smaller the better. The red line is the our un-distilled baseline.

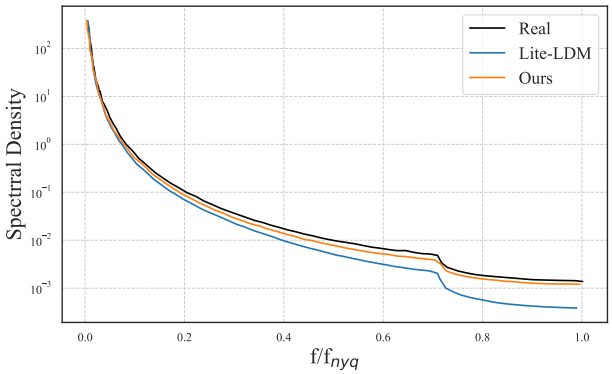

Figure 3. Mean reduced spectrum from real and generated images.

the DPMs even deteriorates the visual quality of the generated images. In our paper, we opt to distill the knowledge on the `output` and `feat-up1`, resulting in 10.5 FID.

3.3. Ablation Study on the Distillation Weights

Our `SD` is trained by mimicking the high-frequency predictions of a pre-trained large DPM. We would like to verify the weight selection for both spatial and frequency distillation terms. An ablation study is conducted by training our `SD` with different λ_s and $\lambda_f \in \{0, 0.01, 0.1, 0.5, 1.0\}$ on FFHQ dataset. The distillation is applied on `output` and `feat-up1`.

λ_s \ λ_f	0	0.01	0.1	0.5	1.0
0	14.7	13.6	11.4	12.8	14.2
0.01	14.3	13.4	11.7	12.3	13.9
0.1	12.3	11.6	10.5	10.9	13.3
0.5	11.6	12.3	11.7	13.0	14.6
1.0	12.8	12.5	12.8	14.3	16.1

Table 1. FID score on different distillation weights. Smaller the better.

We report the results in Table 1. It is noted that a moderate distillation weight of $\lambda_s = \lambda_f = 0.1$ gets the smallest FID score. Thus we apply them as our default hyper-parameters in the main paper.

3.4. Spectral Property OF `SD`

To support our claims that our `SD` is superior at the generation of the high-frequency component, we compare the reduced spectra of real images, Lite-LDM generated images, and images generated by `SD` in Figure 3. In the LSUN-Bedroom dataset, we compute 2D-coefficients for real or generated images and integrate them into 1D signals for visualization. The curves are averaged over 10k images.

Although both models achieve reasonable reconstruction performance on the low-frequency part of the real distribution, Lite-LDM largely falls off at the high end of the spectrum for image synthesis. In contrast, our `SD` achieves much better high-frequency generation quality than the baseline. It again highlights our core motivation to strengthen the high-frequency components for small DPMs.

3.5. Visualization for the Generated Images

To further support the effectiveness of our proposed `SD`, we visualize more generated high-resolution images on FFHQ, CelebA-HQ and class-conditioned ImageNet.

The synthesized results are demonstrated in Figure 5 for FFHQ, Figure 4 for CelebA-HQ and Figure 6 and Figure 7 for ImageNet dataset. With around 10% the computational cost of the standard LDM, our model still achieves superior visual quality and sample diversity. For example, `SD` is excellent at the restoration of animal feathers and human hair. On both face generation tasks, fluffy and glossy hair is clearly visible. Those results underpin that `SD` manages to generate realistic high-frequency image details.

However, we observe that `SD` still face difficulty in producing dense but connected curves, when synthesizing the `spaghetti` and cable on the `pirate ship`, as displayed in Figure 7.

4. Experimental Details

4.1. Evaluation Metrics

FID. We compute the FID [1] score using `torchmetrics.image.fid` by counting the Fréchet distance between 50K real training images and 50K synthesized images.

MACs. MACs measure the theoretical amount of Multiply-Add Operations in the neural networks for certain input sizes. To ensure a fair comparison, we re-estimate the MACs number using `flops-counter.pytorch`. For

Figure 4. Images generated on CelebA-HQ Dataset with 500 DDIM steps.

Figure 5. Images generated on FFHQ Dataset with 500 DDIM steps.

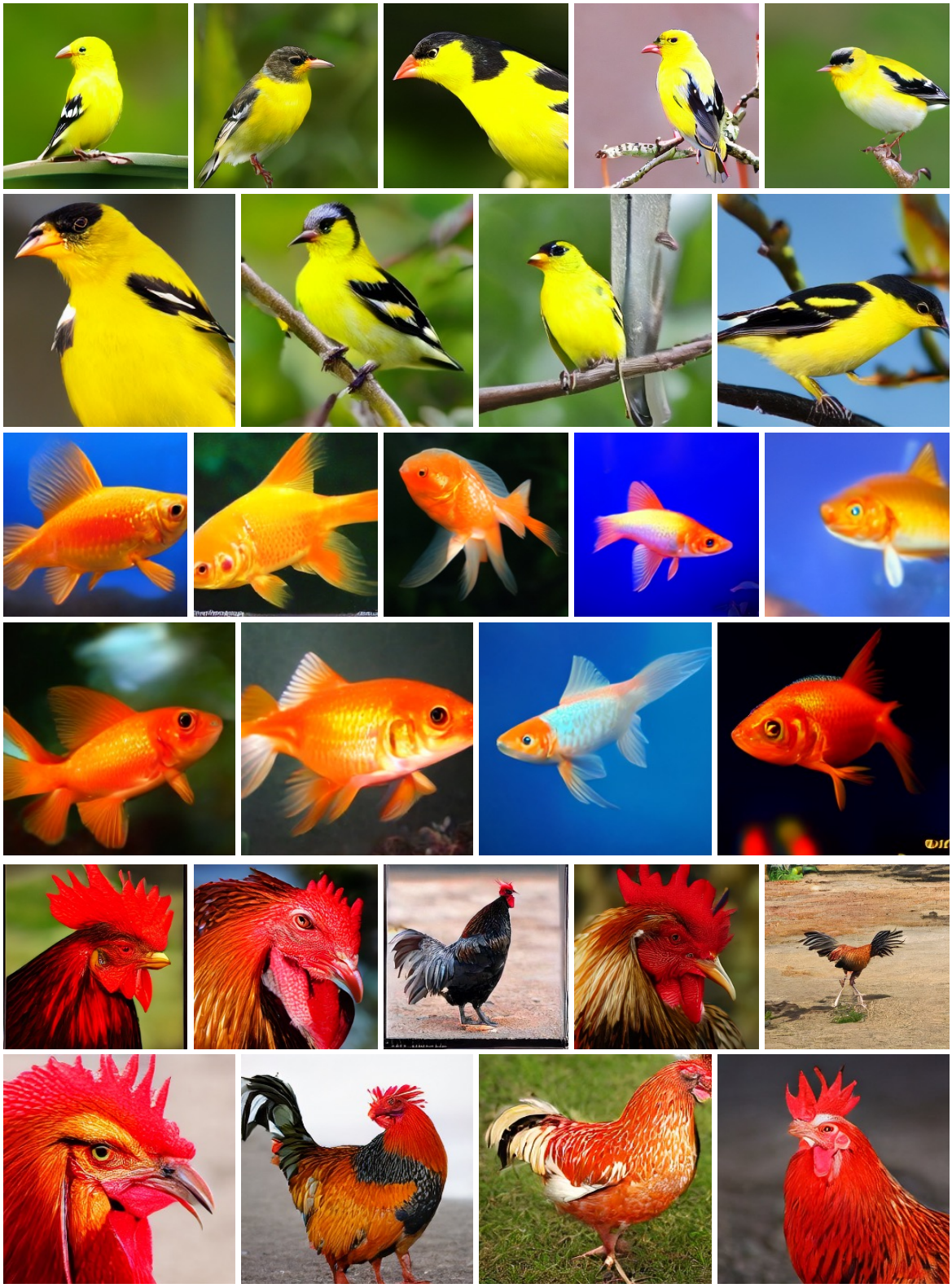

Figure 6. Randomly sampled images from class n01531178, n01443537 and n01514668.

Figure 7. Randomly sampled images from class n02106166, n07831146 and n03947888

	FFHQ 256×256	**CelebA-HQ** 256×256	**LSUN-Bedroom** 256×256	**LSUN-Church** 256×256
f	4	4	4	8
latent shape	$64 \times 64 \times 3$	$64 \times 64 \times 3$	$64 \times 64 \times 3$	$32 \times 32 \times 4$
Diffusion Steps	1000	1000	1000	1000
Noise Schedule	Linear	Linear	Linear	Linear
Batch Size×Num GPU	32×8	32×8	32×8	64×8
Iterations	(1k Warmup)150k	(1k Warmup)150k	(1k Warmup)150k	(1k Warmup)150k
Initial Learning Rate	5.12e-4	5.12e-4	5.12e-4	5.12e-4
Learning Rate Schedule	Linear Decay	Linear Decay	Linear Decay	Linear Decay

Table 2. Hyper-parameters for the unconditional image generation.

	ImageNet 256×256	**LAION** 256×256
f	4	8
latent shape	$64 \times 64 \times 3$	$32 \times 32 \times 4$
Diffusion Steps	1000	1000
Noise Schedule	Linear	Linear
Batch Size×Num GPU	32×8	64×32
Iterations	(1k Warmup)300k	(1k Warmup)300k
Initial Learning Rate	5.12e-4	4.096e-3
Learning Rate Schedule	Linear Decay	Linear Decay
w	3.0	2.0

Table 3. Hyper-parameters for the class-conditioned and text-conditioned image generation.

the designated models in the paper, we download the official source code, load the pre-trained models and perform the calculation.

We only count the inference MACs in the diffusion stage for a single time-step but exclude the computation cost for the decoder for latent diffusion models.

Throughput. Throughput measures how many times a system processes in a given amount of time. In our study, we refer to the number of time steps that the diffusion model runs within one second. It provides a fair comparison of the running speed of different DPM architectures since it precludes the influence of different sampling strategies.

4.2. Dataset Preprocessing

We follow the data preprocessing pipeline for VQ-GAN[1]. We resize training images to 256×256 and normalized to $[-1, 1]$, without any augmentations and flipping.

4.3. Hyper-Parameters

Toy experiments. In our toy experiment in Sec.4.2, each 40-dimensional signal is sampled from the $f(x) = cos(\alpha 2\pi x), x \in [0, 1]$. It is concatenated with the time step t and fed into the network with 41 input units, $M \in \{64, 1024\}$ hidden units, and 40 output units. We use the activation function of `tanh` in the hidden layer. The model's parameters are updated with an Adam optimizer [2] of a constant learning rate of 1e-3. We adopt a mini-batch size of 1,000 and train the network for 10,000 iterations.

[1] https://github.com/CompVis/taming-transformers

Hyper-Parameters for SD. We provide the hyper-parameters of all experiments in Table 2 and Table 3. In experiments, we use the /texttt[Haar] wavelet transform.

4.4. Network Architecture

The detailed network architectures of our proposed SD for all our experiments are specified in Table 4, Table 5, Table 6 and Table 7. We highlight the our designed down-sample and up-sample with **WG-Down** and **WG-Up**.

5. Source Code

All our source code is provided in the Supplementary Material `code.zip`. Please refer to `README.md` file for more information.

References

[1] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 3

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[3] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017. 2

[4] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 2

[5] Norbert Wiener, Norbert Wiener, Cyberneticist Mathematician, Norbert Wiener, Norbert Wiener, and Cybernéticien Mathématicien. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*, volume 113. MIT press Cambridge, MA, 1949. 1

Auto-Encoder	f=4,VQVAE (Z=8192, d=3)	
	Input Size	Configuration
Down1	64×64	3×Resblock, channel=64 **WG-Down**
Down2	32×32	3×Resblock, channel=128 Self-Attn, head = 4 **WG-Down**
Down3	16×16	3×Resblock, channel=192 Self-Attn, head = 6 **WG-Down**
Down4	8×8	3×Resblock, channel=256 Self-Attn, head = 8 **WG-Down**
Mid-Stage	4×4	Resblock, channel=256 Self-Attn, head = 8 Resblock, channel=256
Up4	4×4	3×Resblock, channel=256 Self-Attn, head = 8 **WG-Up**
Up3	8×8	3×Resblock, channel=192 Self-Attn, head = 6 **WG-Up**
Up2	16×16	3×Resblock, channel=128 Self-Attn, head = 4 **WG-Up**
Up1	32×32	3×Resblock, channel=64 **WG-Up**

Table 4. Network Architecture for on FFHQ, CelebA-HQ and LSUN-Bedroom dataset.

Auto-Encoder	f=8, KL-VAE	
	Input Size	Configuration
Down1	32×32	2×Resblock, channel=64 Self-Attn, head = 32 **WG-Down**
Down2	16×16	2×Resblock, channel=128 Self-Attn, head = 4 **WG-Down**
Down3	8×8	2×Resblock, channel=128 Self-Attn, head = 4 **WG-Down**
Down4	4×4	2×Resblock, channel=256 Self-Attn, head = 8 **WG-Down**
Down5	2×2	2×Resblock, channel=256 **WG-Down**
Mid-Stage	1×1	Resblock, channel=256 Self-Attn, head = 8 Resblock, channel=256
Up5	1×1	2×Resblock, channel=256 **WG-Up**
Up4	2×2	2×Resblock, channel=256 Self-Attn, head = 8 **WG-Up**
Up3	4×4	2×Resblock, channel=128 Self-Attn, head = 4 **WG-Up**
Up2	8×8	2×Resblock, channel=128 Self-Attn, head = 4 **WG-Up**
Up1	16×16	2×Resblock, channel=64 Self-Attn, head = 2 **WG-Up**

Table 5. Network Architecture for on LSUN-Church dataset.

Auto-Encoder	f=4,VQVAE (Z=8192, d=3)	
	Input Size	Configuration
Class Embedding	1000	`nn.Embedding`(1000, 512)
Down1	64×64	3×Resblock, channel=64 **WG-Down**
Down2	32×32 Cond: 512×1	3×Resblock, channel=128 Self-Attn+Cross-Attn, head = 4 **WG-Down**
Down3	16×16 Cond: 512×1	3×Resblock, channel=192 Self-Attn+Cross-Attn, head = 6 **WG-Down**
Down4	8×8 Cond: 512×1	3×Resblock, channel=256 Self-Attn+Cross-Attn, head = 8 **WG-Down**
Mid-Stage	4×4 Cond: 512×1	Resblock, channel=256 Self-Attn+Cross-Attn, head = 8 Resblock, channel=256
Up4	4×4 Cond: 512×1	3×Resblock, channel=256 Self-Attn+Cross-Attn, head = 8 **WG-Up**
Up3	8×8 Cond: 512×1	3×Resblock, channel=192 Self-Attn+Cross-Attn, head = 6 **WG-Up**
Up3	16×16 Cond: 512×1	3×Resblock, channel=128 Self-Attn+Cross-Attn, head = 4 **WG-Up**
Up1	32×32	3×Resblock, channel=64 **WG-Up**

Table 6. Network Architecture for on class-conditioned ImageNet dataset.

Auto-Encoder	f=8,KL-VAE	
	Input Size	Configuration
Text-encoder	-	`clip-vit-large-patch14`
Down1	32×32 Cond: 768×77	2×Resblock, channel=128 Self-Attn+Cross-Attn, head = 4 **WG-Down**
Down2	16×16 Cond: 768×77	2×Resblock, channel=256 Self-Attn+Cross-Attn, head = 8 **WG-Down**
Down3	8×8 Cond: 768×77	2×Resblock, channel=512 Self-Attn+Cross-Attn, head = 16 **WG-Down**
Down4	4×4	2×Resblock, channel=512 **WG-Down**
Mid-Stage	2×2 Cond: 768×77	Resblock, channel=512 Self-Attn+Cross-Attn, head = 16 Resblock, channel=256
Up4	2×2	2×Resblock, channel=512 **WG-Up**
Up3	4×4 Cond: 768×77	2×Resblock, channel=512 Self-Attn+Cross-Attn, head = 16 **WG-Up**
Up3	8×8 Cond: 768×77	2×Resblock, channel=256 Self-Attn+Cross-Attn, head = 8 **WG-Up**
Up1	16×16 Cond: 768×77	2×Resblock, channel=128 Self-Attn+Cross-Attn, head = 4 **WG-Up**

Table 7. Network Architecture for on text-conditioned LAION-400M and MS-COCO dataset.