Supplementary Material: Accelerating Score-based Generative Models with Preconditioned Diffusion Sampling

Hengyuan Ma[1], Li Zhang[1⋆], Xiatian Zhu[2], and Jianfeng Feng[1]

[1] Fudan University
[2] University of Surrey

1 Proofs of theorems

Theorem 1. *The steady-state distribution of*

$$d\mathbf{x} = \frac{\epsilon^2}{2} \bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) dt + \epsilon d\mathbf{w}, \tag{1}$$

and

$$d\mathbf{x} = \frac{\epsilon^2}{2}(MM^\intercal + S) \bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) dt + \epsilon M d\mathbf{w}, \tag{2}$$

are the same, as long as the linear operator M is invertible and the linear operator S is skew-symmetric.

Proof. The Fokker-Planck equation of Eq. (2) is

$$\frac{\partial p}{\partial t} = -\frac{\epsilon^2}{2} MM^\intercal \bigtriangledown_{\mathbf{x}} \cdot (\bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) p) + \bigtriangledown_{\mathbf{x}} \cdot S \bigtriangledown_{\mathbf{x}} (\log p^*(\mathbf{x}) p) + \frac{\epsilon^2}{2} MM^\intercal \Delta_{\mathbf{x}} p. \tag{3}$$

Since S is skew-symmetric, $\bigtriangledown_{\mathbf{x}} \cdot (S \bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) p) = 0$. Then, the probabilistic density function of the steady-state distribution satisfies

$$\bigtriangledown_{\mathbf{x}} \cdot (\bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) p) = \Delta_{\mathbf{x}} p, \tag{4}$$

where we make use of the invertibility of M. This is the same as the steady state function of Eq. (1)

$$\bigtriangledown_{\mathbf{x}} \cdot (\bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x}) p) = \Delta_{\mathbf{x}} p. \tag{5}$$

As a result, the theorem is proved.

⋆ Li Zhang (lizhangfd@fudan.edu.cn) is the corresponding author with School of Data Science, Fudan University. Xiatian Zhu is with Surrey Institute for People-Centred Artificial Intelligence, CVSSP, University of Surrey.

Theorem 2. *Consider the diffusion process*

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + G(t)d\mathbf{w}, \tag{6}$$

where $\mathbf{f} : \mathbb{R}^d \otimes \mathbb{R} \to \mathbb{R}^d$, $G : \mathbb{R} \to \mathbb{R}^{d\times d}$. *$M$ is an invertible $d \times d$ matrix and S is a skew-symmetric $d \times d$ matrix. Denote p^* as the steady-state distribution of Eq. (6), then the process*

$$d\mathbf{x} = MM^\intercal \mathbf{f}(\mathbf{x}, t)dt + S \bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x})dt + MG(t)d\mathbf{w}, \tag{7}$$

has the same steady-state distribution as Eq. (6), given $G(t)G(t)^\intercal$ and $M^\intercal$ are commutable $\forall t$.

Proof. The steady-state distribution of Eq. (6) satisfies the following equation

$$\bigtriangledown_{\mathbf{x}} \cdot [p\mathbf{f}(\mathbf{x}, t)] = \frac{G(t)G(t)^\intercal}{2} \Delta_{\mathbf{x}} p. \tag{8}$$

The Fokker-Planck equation of Eq. (7) is

$$\frac{\partial p}{\partial t} = -MM^\intercal \bigtriangledown_{\mathbf{x}} \cdot [p\mathbf{f}(\mathbf{x}, t)] + \frac{MG(t)G(t)^\intercal M^\intercal}{2} \Delta_{\mathbf{x}} p, \tag{9}$$

where we have used the skew symmetry of S so that $\bigtriangledown_{\mathbf{x}} \cdot (S \bigtriangledown_{\mathbf{x}} \log p(\mathbf{x})p) = 0$. Then, the steady-state distribution of Eq. (7) satisfies the following equation

$$MM^\intercal \bigtriangledown_{\mathbf{x}} \cdot [p\mathbf{f}(\mathbf{x}, t)] = \frac{MG(t)G(t)^\intercal M^\intercal}{2} \Delta_{\mathbf{x}} p, \tag{10}$$

which is equivalent to Eq. (8) due to the invertiblity of M and commutability of $G(t)G(t)^\intercal$ and $M^\intercal$. Therefore, the theorem is proved.

2 Preconditioning a diffusion process in the frequency domain

In this section, we will prove theoretically why we can *directly* regulate the frequency distribution of a diffusion process through the preconditioning strategy, and why it is necessary to do so.

We first show that a diffusion process can be directly transformed to another space (e.g., the frequency domain) via an orthogonal transform. To minimize ambiguity, we denote $p^*(\mathbf{x})$ as $p^*_{\mathbf{x}}(\mathbf{x})$.

Theorem 3. *The Langevin dynamics*

$$d\mathbf{x} = \frac{\epsilon^2}{2} \bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x})dt + \epsilon d\mathbf{w} \tag{11}$$

can be rewritten as

$$d\tilde{\mathbf{x}} = \frac{\epsilon^2}{2} \bigtriangledown_{\tilde{\mathbf{x}}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})dt + \epsilon d\mathbf{w}, \tag{12}$$

where $\tilde{\mathbf{x}} : s = B\mathbf{x}$, given B is an orthogonal transform.

Proof. Multiplying B on both sides of Eq. (11), we have:

$$d\tilde{\mathbf{x}} = \frac{\epsilon^2}{2} B \bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x}) dt + \epsilon B d\mathbf{w}. \tag{13}$$

We have $Bd\mathbf{w} = d\mathbf{w}$ by the rotational invariance of the standard Wiener process. Now we only need to verify

$$B \bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x}) = \bigtriangledown_{\tilde{\mathbf{x}}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}). \tag{14}$$

Given two d-dimensional random vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ with their respective differentiable density functions $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$, if $g(\mathbf{x}) = \mathbf{y}$, where $g \in \mathbb{R}^d \to \mathbb{R}^d$ is an invertible differentiable transformation, we have

$$p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{x}}(g^{-1}(\mathbf{y})) \left| \det \left[\frac{dg^{-1}(\mathbf{y})}{d\mathbf{y}} \right] \right|. \tag{15}$$

Therefore,

$$\bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x}) = \bigtriangledown_{\mathbf{x}} [\log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) - \log \left| \det \left[B^T \right] \right|] = \bigtriangledown_{\mathbf{x}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}). \tag{16}$$

Using the chain rule of the calculus, we have

$$\bigtriangledown_{\mathbf{x}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) = B^T \bigtriangledown_{\tilde{\mathbf{x}}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}). \tag{17}$$

Combining Eq. (16) and Eq. (17), we have

$$\bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x}) = B^T \bigtriangledown_{\tilde{\mathbf{x}}} \log p^*_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}), \tag{18}$$

which is equivalent to Eq. (14) using the orthogonality of B.

Remark 1. The above result is easy to be extended to a more general case where the drift term $\frac{\epsilon^2}{2} \bigtriangledown_{\mathbf{x}} \log p^*_{\mathbf{x}}(\mathbf{x})$ is replaced by $f(t)\mathbf{x} + \bigtriangledown_{\mathbf{x}} \log q(\mathbf{x}, t)$, if f is a scalar function of time and $q(\cdot, t)$ is a distribution function that may vary over time. Therefore, the theorem can be applied generally to all the diffusion processes adopted in NCSN [5], NCSNv2 [6], and NCSN++ [7].

Specially, when we set B as a two-dimensional discrete cosine transform [1,2], the whole diffusion process can be transformed to the frequency domain without changing its original form. *This explains why we can directly implement a preconditioning operator on the original diffusion process to regulate its frequency distribution.*

There exists a general observation that the amplitude of the high-frequency part of a natural image is dramatically lower than that in the low-frequency part [2]. This means the distribution of natural images exhibits huge gaps in quantity between different coordinates in the frequency domain, causing a severe ill-conditioned issue. *This explains the necessity to regulate the frequency distribution of a diffusion process, which is implemented by preconditioning in this paper.*

3 Parameter settings

We report the settings of the preconditioning operator

$$M[\cdot] = A \odot F^{-1}[R \odot F[\cdot]]. \tag{19}$$

The parameterized frequency filter R is calculated as follows

$$R(c,h,w) = \begin{cases} 1 \text{ , if } (h-0.5H)^2 + (w-0.5W)^2 \leq 2r^2 \\ \lambda \text{ , } \qquad\qquad \text{otherwise} \end{cases}, \tag{20}$$

where C is the channel number, H is the height, and W is the width of an image. $1 \leq c \leq C$, $1 \leq h \leq H$ and $1 \leq w \leq W$. An example of R is given in Fig. 1.

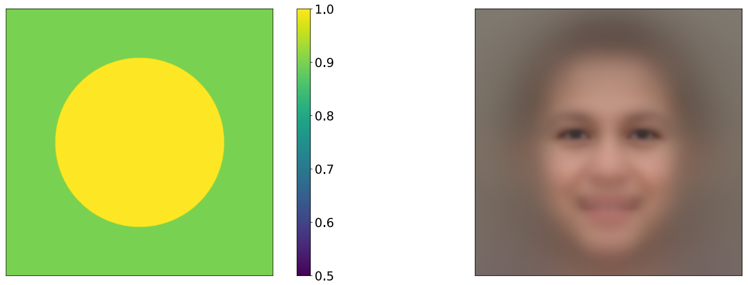

Fig. 1. Examples of (**Left**) frequency preconditioning R ($(r, \lambda) = (0.2H, 0.9)$) and (**Right**) mean of FFHQ [3] dataset used for constructing space preconditioning A used in proposed preconditioning operator M (Eq. (19)).

Empirically, we find that it is necessary to normalize both space and frequency filter that are calculated by the dataset statistics as follows

$$A(c,w,h) = \frac{A(c,w,h)}{\max A(c,w,h)}, \tag{21}$$

and

$$R(c,w,h) = \frac{1}{\alpha}\left(\frac{R(c,w,h)}{\max R(c,w,h)} + \alpha - 1\right), \tag{22}$$

where α is the normalization parameter. This allows us to adaptively scale the frequency coordinates according to the specific amplitudes.

We provide the parameter settings used in our experiments in Table. 1. For NCSN [5] and NCSNv2 [6], we construct the frequency filter R following Eq. (20). The two parameters r and λ used in each dataset is shown in Table. 2. For these two models, we do not apply the space preconditioning.

For NCSN++ [7], we we construct the frequency filter R using the statistics from the target dataset directly. We apply the space preconditioning calculated by the dataset statistics for FFHQ dataset, since there is a clear space structure priors (the layout of human faces), and we do not apply the space preconditioning for other datasets.

Table 1. Parameters of PDS used for constructing frequency filter on NCSN [5] and NCSNv2 [6] following Eq. (20).

Dataset	Resolution	Model	Iterations	r	λ	use space preconditioning?
MNIST	28×28	NCSN	20	$0.2H$	1.6	✗
LSUN (church)	96×96	NCSNv2	126	$0.2H$	1.6	✗
			157	$0.2H$	1.6	✗
			210	$0.2H$	1.6	✗
LSUN (tower)	128×128	NCSNv2	65	$0.2H$	1.1	✗
			81	$0.2H$	1.1	✗
			108	$0.2H$	1.1	✗

Table 2. Parameters of PDS used for constructing frequency filter on NCSN++ [7] following Eq. (22).

Dataset	Resolution	Iterations	α	use space preconditioning?
CIFAR-10	32×32	100	5	✗
		200	10	✗
LSUN (bedroom)	256×256	166	5	✗
LSUN (church)	256×256	166	5	✗
FFHQ	1024×1024	66	5	✓

4 Implementation details

We use the public released codebases of NCSN[3], NCSNv2[4] and NCSN++[5]. For facilitating the comparisons, we follow the same preprocessing as [5,6,7]. We conduct all the following experiments with PyTorch on NVIDIA RTX 3090 GPUs.

5 More quantitative results

In this section, we report more quantitative results using Clean-FID (Fréchet Inception Distance) metric [4] to verify that our PDS accelerates the vanilla diffusion process while generating images with high quality. It is observed in Table 3 that the Clean-FID scores of our PDS are all dramatically smaller than those by the original methods in all the cases, consistent with our visualization results (Fig. 6-8).

[3] https://github.com/ermongroup/ncsn
[4] https://github.com/ermongroup/ncsnv2
[5] https://github.com/yang-song/score_sde

Table 3. Quantitative evaluation with the Clean-FID (Fréchet Inception Distance) metric [4] for accelerated diffusion process. We generate $50k$ images for each method.

Model	NCSNv2 [6]		NCSN++ [7]		
Dataset	LSUN	LSUN	LSUN	LSUN	FFHQ
Class	Church	Tower	Bedroom	Church	Face
Resolution	96×96	128×128	256×256	256×256	1024×1024
Iterations	156	108	166	166	66
Vanilla	217.9	67.2	393.7	393.3	463.2
PDS	**65.7**	**43.8**	**16.9**	**15.0**	**61.2**

6 Solenoidal term analysis

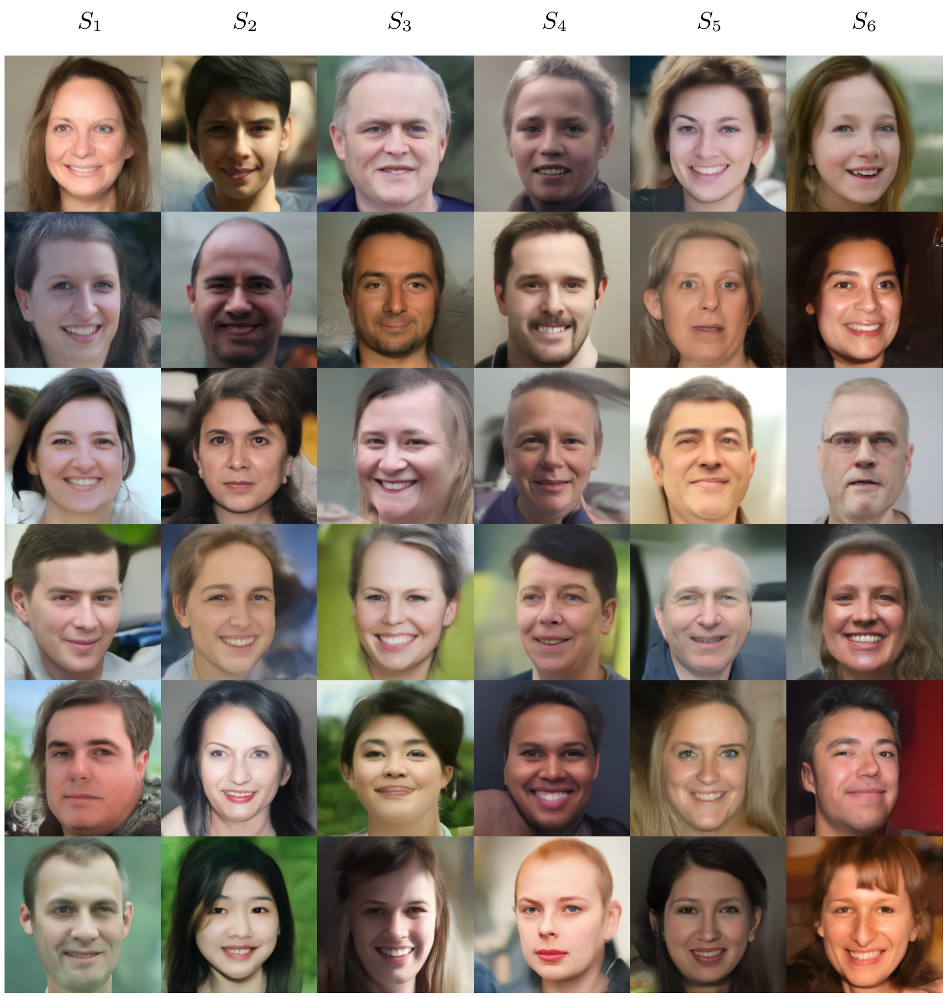

Fig. 2. Facial images at a resolution of 1024×1024 generated by NCSN++ [7] with our PDS using different solenoidal items. Sampling iterations: 66. Dataset: FFHQ [3].

In this section, we investigate the effect of the solenoidal term $S \log p^*(\mathbf{x})$ to the diffusion process

$$d\mathbf{x} = \frac{\epsilon^2}{2}(M^{-1}M^{-\mathbf{T}} + \omega S) \bigtriangledown_{\mathbf{x}} \log p^*(\mathbf{x})dt + \epsilon M^{-1}d\mathbf{w} \tag{23}$$

In Sec. 5 of the main paper, we have shown that using $S = Re[F - F^{\mathsf{T}}]$ has no obvious effect on the diffusion process. Now we study more cases. Denote $P_{m,n}$

as the shift operator that rolls the input image for m places along the height coordinate and rolls the input image for n places along the width coordinate. We then test how the skew-symmetric operators in Eq. (24) would affect the diffusion process.

$$\begin{aligned}
S_1 &= P_{1,1} - P_{1,1}^{\mathbf{T}} \\
S_2 &= P_{10,10} - P_{10,10}^{\mathbf{T}} \\
S_3 &= P_{100,100} - P_{100,100}^{\mathbf{T}} \\
S_4 &= Re[F[P_{1,1} - P_{1,1}^{\mathbf{T}}]F^{-1}] \\
S_5 &= Re[F[P_{10,10} - P_{10,10}^{\mathbf{T}}]F^{-1}] \\
S_6 &= Re[F[P_{100,100} - P_{100,100}^{\mathbf{T}}]F^{-1}].
\end{aligned} \tag{24}$$

The sampling results are shown in Fig. 2, where we set $\omega = 1000$. It is observed that again all these solenoidal terms do not impose an obvious effect on the sampling quality. Additionally, as displayed in Fig. 3, these solenoidal terms also do not make an obvious effect on acceleration. Nevertheless, we only study the effect of some special cases of the solenoidal terms, which does not mean there are no solenoidal terms that can accelerate the diffusion process, and the search for these solenoidal terms is in a further study.

7 Limitations

In general, there are several parameters in the preconditioning matrix of PDS need to be determined. A further study is needed to enable PDS find the best parameter settings automatically. Although DDPMs are a variant of SGMs, we find PDS can not directly used on DDPMs, since the diffusion process of DDPMs is not a Langevin dynamics. Nevertheless, we find that it is possible to rewrite this diffusion process to imitate the structure of Langevin dynamics, then use PDS for acceleration. We leave it for future study.

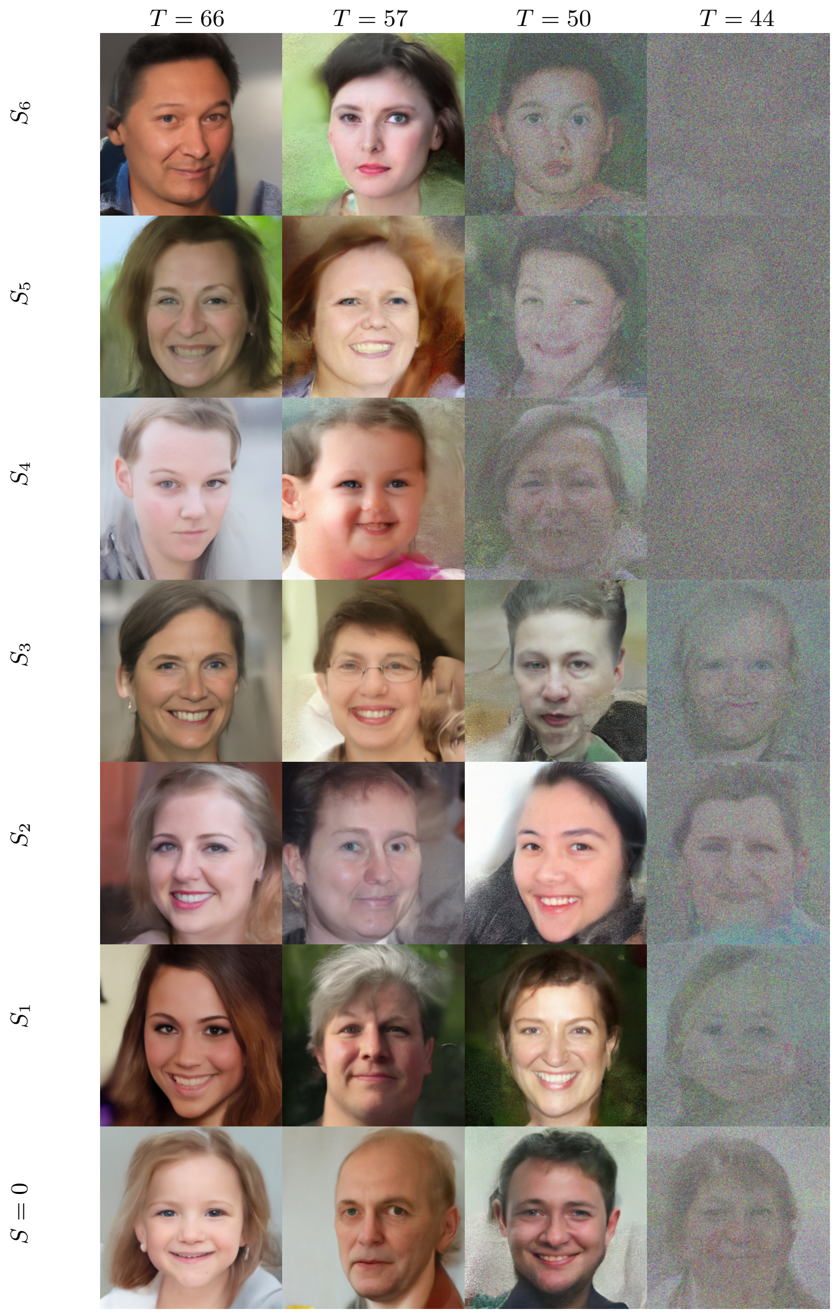

Fig. 3. Facial images at a resolution of 1024×1024 generated by NCSN++ [7] with our PDS under different sampling iterations and different solenoidal items described in Eq. (24). We set R following Eq. (22) and do not apply space preconditioning Dataset: FFHQ [3].

8 More examples

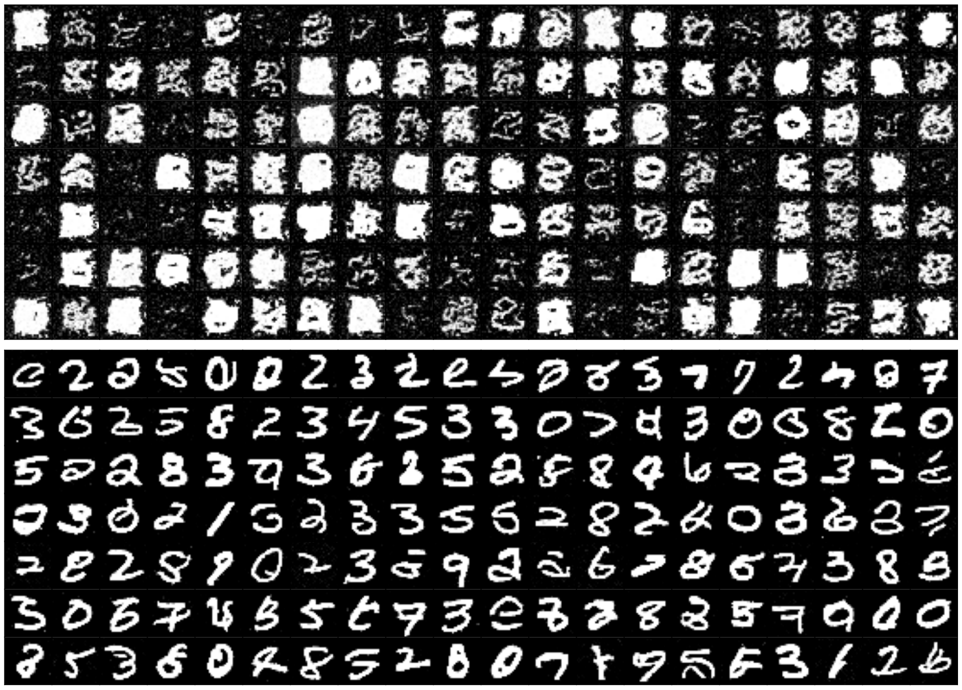

Fig. 4. Sampling using NCSN [5] on MNIST (28×28). **Top**: Results by the original sampling method with 20 sampling iterations. **Bottom**: Results by our PDS with 20 sampling iterations, where we set $(r, \lambda) = (0.2H, 1.6)$.

References

1. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE transactions on Computers (1974) 3
2. Bovik, A.C.: The Essential Guide to Image Processing (2009) 3
3. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019) 4, 7, 9, 13, 14
4. Parmar, G., Zhang, R., Zhu, J.Y.: On aliased resizing and surprising subtleties in gan evaluation. arXiv preprint (2021) 5, 6
5. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: NeurIPS (2019) 3, 4, 5, 10
6. Song, Y., Ermon, S.: Improved techniques for training score-based generative models. In: NeurIPS (2020) 3, 4, 5, 6, 11
7. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: ICLR (2021) 3, 4, 5, 6, 7, 9, 12, 13, 14

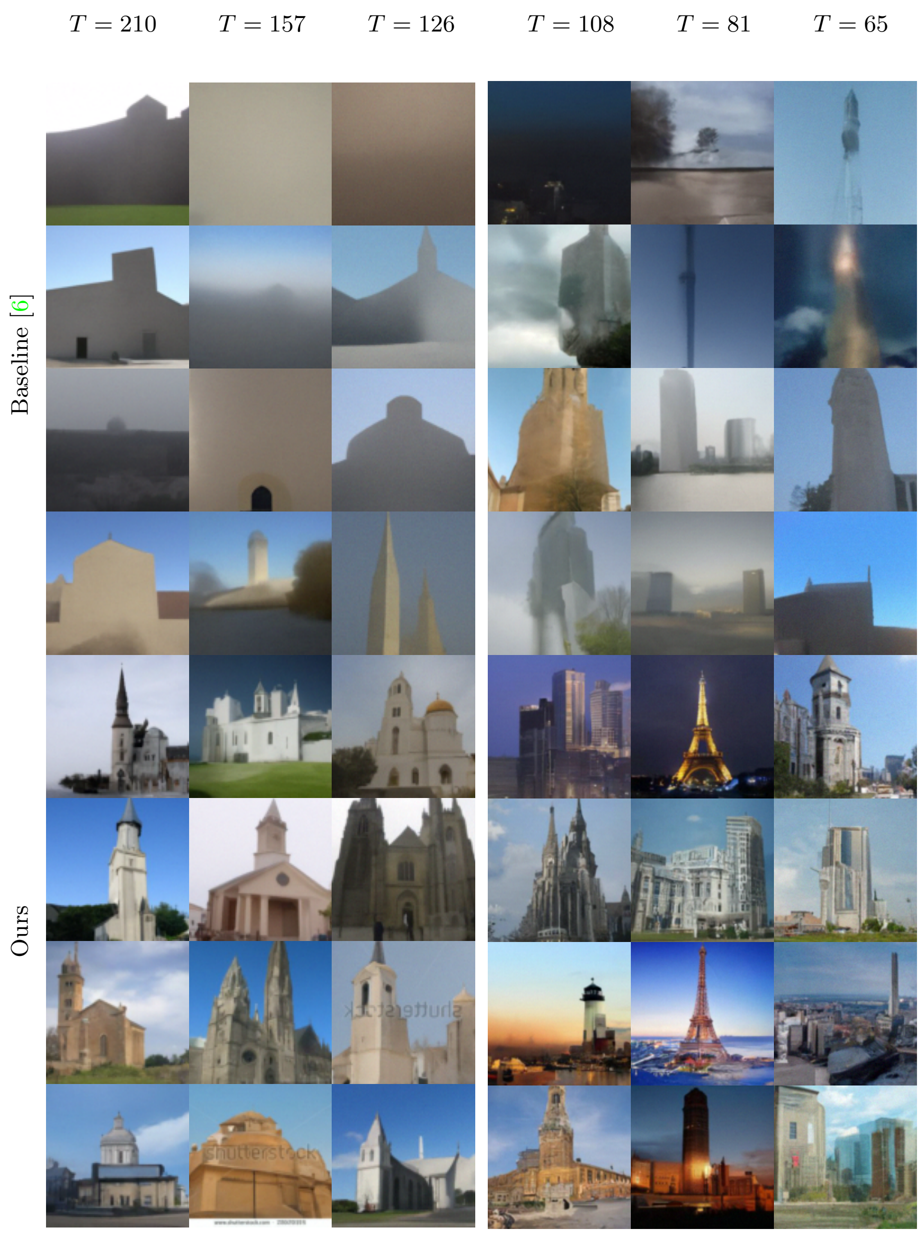

Fig. 5. Sampling using NCSNv2 [6] on LSUN (church 96×96 and tower 128×128) under different iteration numbers.

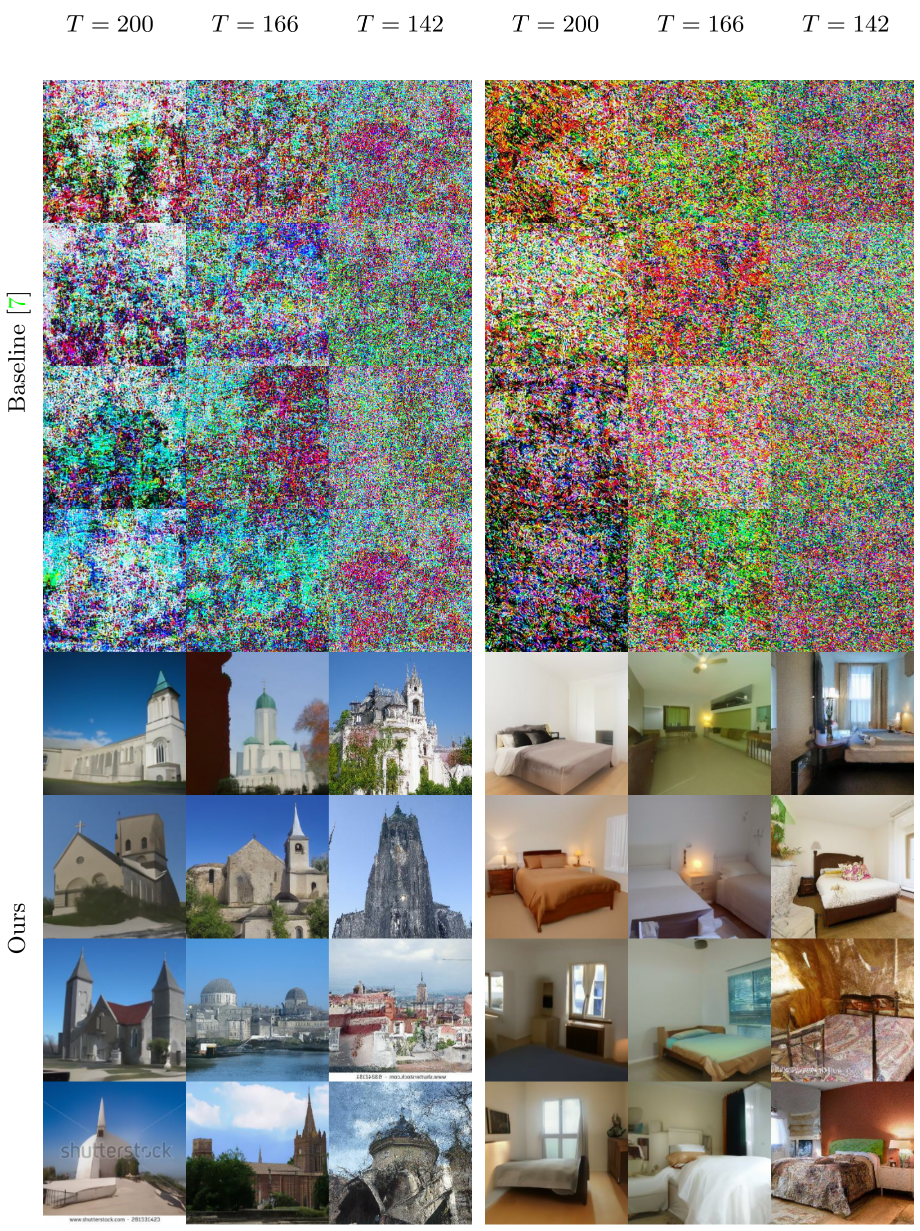

Fig. 6. Sampling using NCSN++ [7] on LSUN (church and bedroom) at a resolution of 256×256 under different iteration numbers. It is observed that when the iteration number decreases, both the original method and our PDS generate samples with high-frequency noise, but the quality of the samples produced by the original method drops much more dramatically.

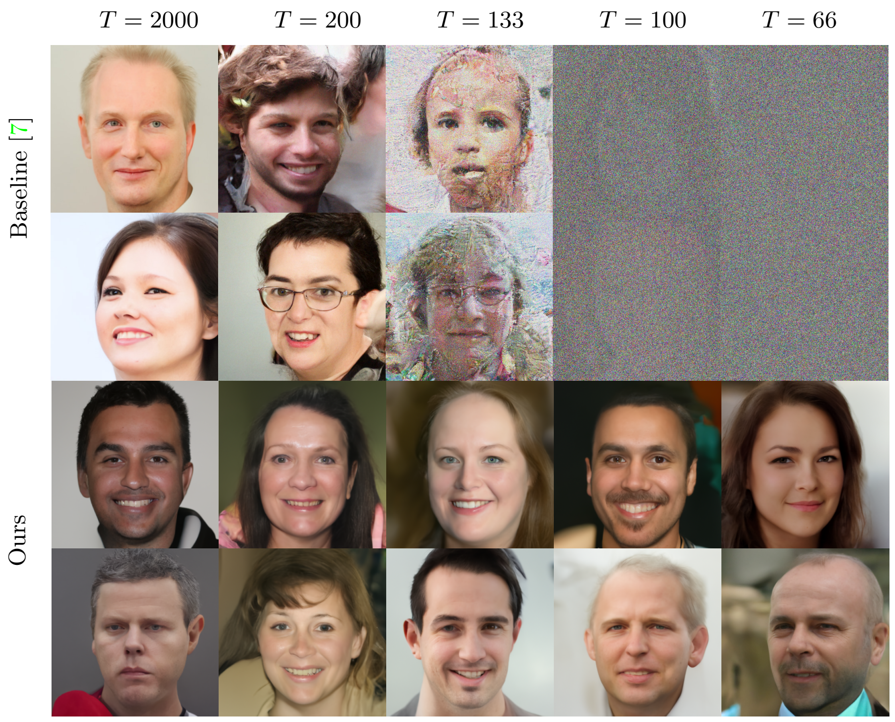

Fig. 7. FFHQ [3] (facial images) at a resolution of 1024×1024 generated by NCSN++ [7] under a variety of sampling iterations (top) without and (bottom) with our PDS. It is evident that NCSN++ decades quickly with increasingly reduced sampling iterations, which can be well solved with PDS.

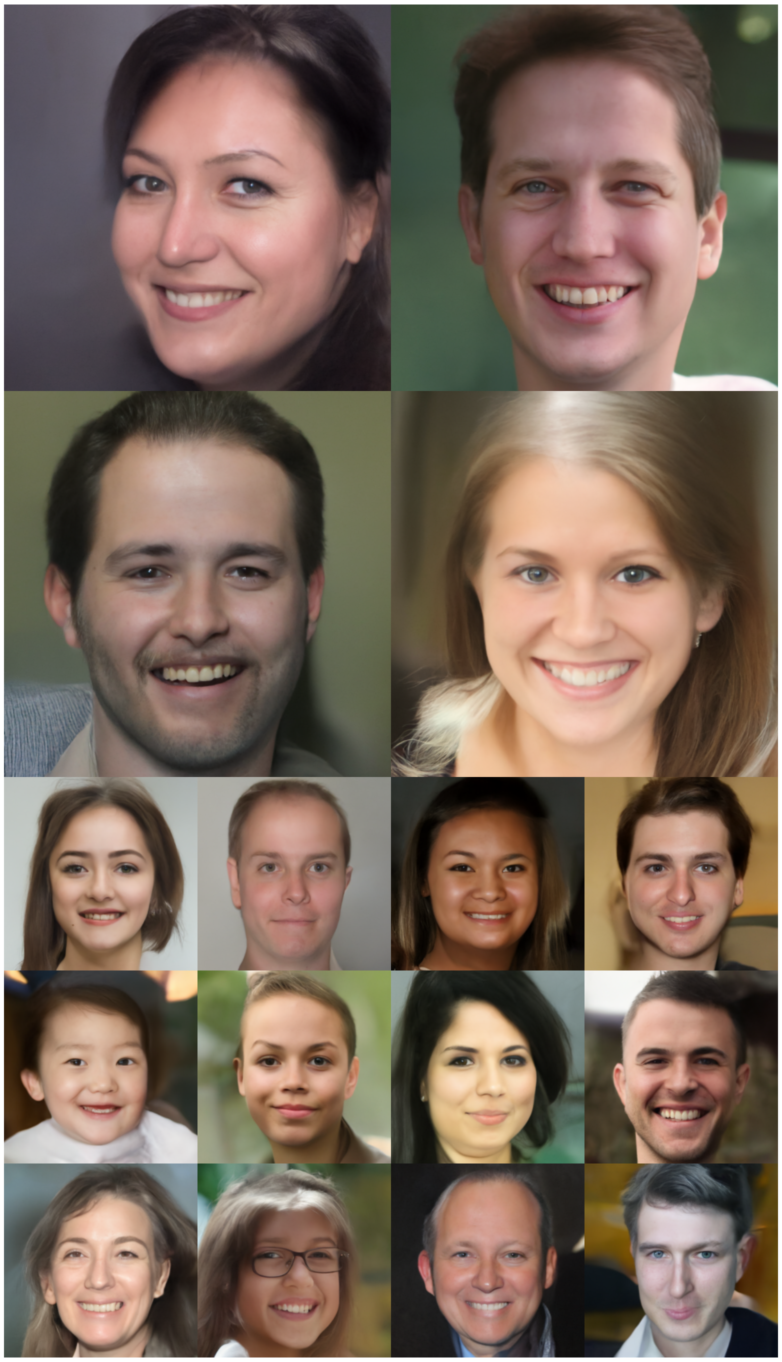

Fig. 8. Facial images at a resolution of 1024×1024 generated by NCSN++ [7] with our PDS. Sampling iterations: 66. Dataset: FFHQ [3].