Observation-Guided Diffusion Probabilistic Models

Junoh Kang*[1] Jinyoung Choi*[1] Sungik Choi[3] Bohyung Han[1,2]
Computer Vision Laboratory, [1]ECE & [2]IPAI, Seoul National University [3]LG AI Research
{junoh.kang, jin0.choi, bhhan}@snu.ac.kr, sungik.choi@lgresearch.ai

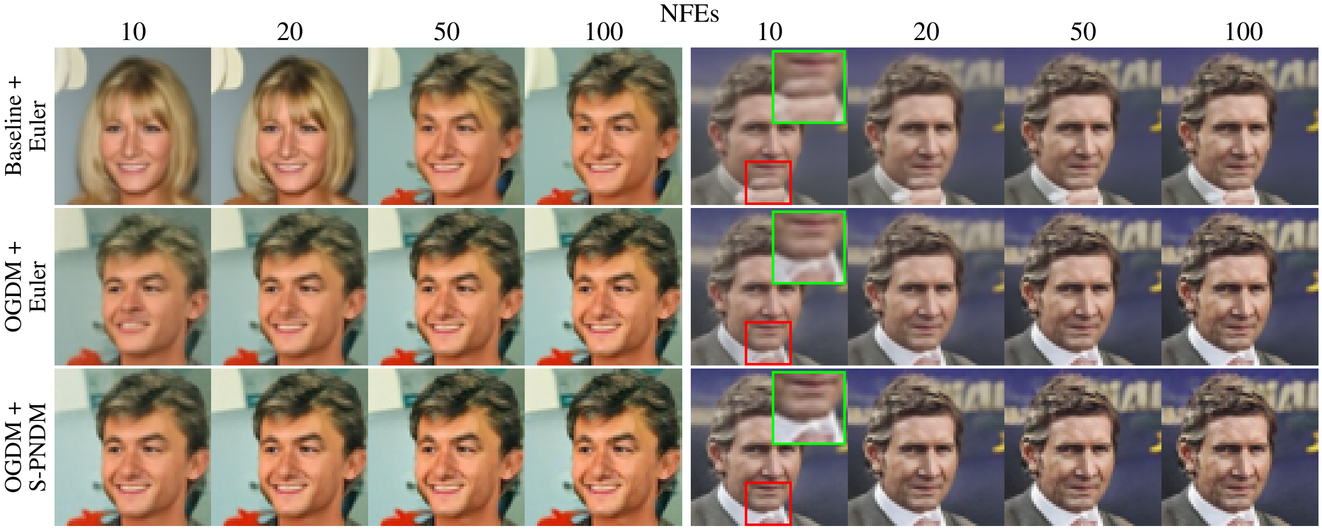


Figure 1. Comparisons of images generated by the ADM backbone on the CelebA dataset with deterministic samplers using the same initial noise but different NFEs. The entries on the leftmost column of the figure denote the combinations of the training and inference methods. (Left) The baseline model generates samples with inconsistent attributes, *e.g.*, gender, hair, *etc*., by varying NFEs while our approach preserves such properties. (Right) The samples generated by the baseline method with a small number of NFEs tend to be blurry and unrealistic. Also, they have unnaturally bright and textureless areas around the chin of the person.

Abstract

We propose a novel diffusion-based image generation method called the observation-guided diffusion probabilistic model (OGDM), which effectively addresses the trade-off between quality control and fast sampling. Our approach reestablishes the training objective by integrating the guidance of the observation process with the Markov chain in a principled way. This is achieved by introducing an additional loss term derived from the observation based on a conditional discriminator on noise level, which employs a Bernoulli distribution indicating whether its input lies on the (noisy) real manifold or not. This strategy allows us to optimize the more accurate negative log-likelihood induced in the inference stage especially when the number of function evaluations is limited. The proposed training scheme is also advantageous even when incorporated only into the fine-tuning process, and it is compatible with various fast inference strategies since our method yields better denoising networks using the exactly the same inference procedure without incurring extra computational cost. We demonstrate the effectiveness of our training algorithm using diverse inference techniques on strong diffusion model baselines. Our implementation is available at https://github.com/Junoh-Kang/OGDM_edm.

1. Introduction

Diffusion probabilistic models [7, 27] have shown impressive generation performance in various domains including image [3, 25], 3D shapes [38], point cloud [21], speech [9, 15], graph [8, 24], and many others. The key idea

*indicates equal contribution.

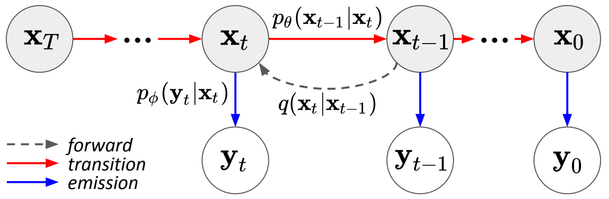


Figure 2. The graphical model of the proposed denoising process with observations.

behind these approaches is to formulate data generation as a series of denoising steps of the diffusion process, which sequentially corrupts training data towards a random sample drawn from a prior distribution, *e.g*., Gaussian distribution.

As diffusion models are trained with an explicit objective, *i.e*., maximizing log-likelihood, they are advantageous over Generative Adversarial Networks (GANs) [5] in terms of learning stability and sample diversity. Moreover, the iterative backward processes and accompanying sampling strategies further improve the quality of samples at the expense of computational efficiency; the tedious inference process involving thousands of network forwarding steps is a critical drawback of diffusion models.

The step size of diffusion models has a significant impact on the expressiveness of the models as the Gaussian assumption imposed on the reverse (denoising) sampling holds only when the step size is sufficiently small [27]. On the other hand, the backward distribution deviates from the Gaussian assumption as the step size grows, resulting in an inaccurate modeling. This exacerbates the discrepancy between the training objective and the negative log-likelihood at inference. Consequently, performance degradation is inevitable with coarse time steps. To alleviate the deviation from the true objective caused by large step sizes, our approach incorporates an observation of each state corresponding to the perturbed data. To be specific, we consider the data corruption and denoising processes to follow the transition probabilities that respectively align with the forward and backward distributions of DDPM [7] while an observation at each time step following the emission probability aids in achieving a more accurate backward prediction. Fig. 2 depicts the graphical model of our method.

Our approach offers a significant benefit in the sense that it precisely maximizes the log-likelihood at inference even when employing fast sampling strategies with large step sizes. The observation process plays an important role during training to adjust the denoising steps towards a more accurate data manifold especially when the reverse process deviates from the Gaussian distribution. When it comes to the inference stage, the observation process is no longer an accountable factor and hence incurs no additional computational overhead for sampling.

The main question in this approach is what is observable given a state at each time step. We define an observation following the Bernoulli distribution on the probability of whether noisy data lies on the manifold of real data with the corresponding noise level. From a practical point of view, we implement this observation with a score of a time-dependent discriminator, taking either true denoised samples or fake ones given by the learned denoising network.

Our main contributions are summarized below:

- We propose an observation-guided diffusion probabilistic model, which accelerates inference procedure while maintaining high sample quality. Our approach only affects the training procedures, resulting in no extra computational or memory overhead during inference.
- We derive a principled surrogate loss maximizing the log-likelihood in the observation-guided setting and show its effectiveness in minimizing the KL-divergence between temporally coarse forward and backward processes.
- Our training objective is applicable to various inference methods with proper adjustments, which allows us to utilize diverse fast sampling strategies that further improve sample quality.
- The proposed technique can be employed for training from scratch or for fine-tuning from pretrained models; compatibility with fine-tuning significantly enhances the practicality of our method.

The rest of this paper is organized as follows. Sec. 2 reviews related work and Sec. 3 describes our main algorithm with the justification of the proposed objective. We present experimental results and analyses in Sec. 4, discuss future work in Sec. 5, and conclude our paper in Sec. 6.

2. Related Work

There exists a series of studies on diffusion probabilistic models [7, 27, 29] that have contributed to accelerated sampling. A simple and intuitive method is to simply skip intermediate time steps and sample a subset of the predefined time steps used for training as suggested by DDIM [28]. By interpreting the diffusion model as solving a specific SDE [29], advanced numerical SDE or ODE solvers [4, 10, 12, 18, 29] are introduced to speed up the backward process. For instance, EDM [12] employs a second-order Heun's method [31] as its ODE solver, demonstrating that simply adopting existing numerical methods can improve performance. On the other hand, some papers further refine numerical solvers tailored for diffusion models. For example, PNDM [18] provides a pseudo-numerical solver by combining DDIM and high-order classical numerical methods such as Runge-Kutta [31] and linear multi-step [33], and GENIE [4] applies a higher-order solver to the DDIM ODE.

On the other hand, [1, 2, 23, 35] aim to find the better (optimal) parameters of the reverse process with or without training. For instance, Analytic-DPM [2] presents a training-free inference algorithm by estimating the optimal reverse variances under shortened inference steps and computing the KL-divergence between the corresponding forward and reverse processes in analytic forms. Knowledge distillation [20, 22, 26, 30] is another direction for better optimization, where a single time step in a student model learns to simulate the representations from multiple denoising steps in a teacher model. Note that our approach is orthogonal and complementary to the aforementioned studies since our goal is to train better denoising networks robust to inference with large step sizes.

There are a few of methods [13, 34, 36] that adopt time-dependent discriminators with diffusion process, yet their motivations and intentions differ significantly from ours. The time-dependent discriminator in DDGAN [36] is designed to guide the generator in approximating non-Gaussian reverse processes while Diffusion-GAN [34] employs diffusion process to mitigate overfitting of the discriminator. DG [13], on the other hand, utilize the discriminator during inference stages to adjust the score estimation additionally. In contrast, the discriminator in our approach serves as a means to provide observations to the diffusion models during the training phase, without being involved in the inference phase.

3. Observation-Guided Diffusion Probabilistic Models

This section describes the mathematical details of our algorithm and analyzes how to interpret and implement the derived objective function.

3.1. Properties

The proposed observation-guided diffusion probabilistic model, defined by the graphical model in Fig. 2, involves two stochastic processes: the state process $\{\mathbf{x}_t\}_{t=0}^{T}$ and the observation process $\{\mathbf{y}_t\}_{t=0}^{T}$. The transition and emission probabilities of the forward process, denoted by $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $q(\mathbf{y}_t|\mathbf{x}_t)$, respectively, are derived by utilizing the following properties given by the graphical model:

$$\mathbf{x}_{t+1}|\mathbf{x}_t \perp\!\!\!\perp \mathbf{x}_{0:t-1}, \mathbf{y}_{0:t} \quad \& \quad \mathbf{y}_t|\mathbf{x}_t \perp\!\!\!\perp \mathbf{x}_{0:t-1}, \mathbf{y}_{0:t-1}, \tag{1}$$

where $\perp\!\!\!\perp$ means statistical independence. In the reverse process, the transition and emission probabilities, $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and $p(\mathbf{y}_t|\mathbf{x}_t)$, are set using the similar properties as

$$\mathbf{x}_{t-1}|\mathbf{x}_t \perp\!\!\!\perp \mathbf{x}_{T:t+1}, \mathbf{y}_{T:t} \;\;\&\;\; \mathbf{y}_t|\mathbf{x}_t \perp\!\!\!\perp \mathbf{x}_{T:t+1}, \mathbf{y}_{T:t+1}. \tag{2}$$

3.2. New surrogate objective

Using Eq. (1) and Bayes' theorem, we derive the joint probability of the forward process as follows:

$$q(\mathbf{x}_{1:T}, \mathbf{y}_{0:T}|\mathbf{x}_0) = q(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t). \tag{3}$$

From Eq. (2), the joint probability of the reverse process is given by

$$p(\mathbf{x}_{T:0}, \mathbf{y}_{T:0}) = p(\mathbf{x}_T) \prod_{t=T}^{1} p(\mathbf{x}_{t-1}|\mathbf{x}_t) \prod_{t=T}^{0} p(\mathbf{y}_t|\mathbf{x}_t). \tag{4}$$

Therefore, we derive the upper bound of the expected negative log-likelihood as

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_0 \sim q}\left[-\log p(\mathbf{x}_0)\right] && (5)\\
&= \mathbb{E}_{\mathbf{x}_0 \sim q}\left[\log \mathbb{E}_{\mathbf{x}_{1:T}, \mathbf{y}_{0:T} \sim q} \frac{q(\mathbf{x}_{1:T}, \mathbf{y}_{0:T}|\mathbf{x}_0)}{p(\mathbf{x}_{0:T}, \mathbf{y}_{0:T})}\right] && (6)\\
&\leq \mathbb{E}_{\mathbf{x}_0 \sim q}\mathbb{E}_{\mathbf{x}_{1:T}, \mathbf{y}_{0:T} \sim q}\left[\log \frac{q(\mathbf{x}_{1:T}, \mathbf{y}_{0:T}|\mathbf{x}_0)}{p(\mathbf{x}_{0:T}, \mathbf{y}_{0:T})}\right] && (7)\\
&= \mathbb{E}_{\mathbf{x}_{0:T}, \mathbf{y}_{0:T} \sim q}\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)\prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p(\mathbf{x}_T)\prod_{t=T}^{1} p(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&\quad + \mathbb{E}_{\mathbf{x}_{0:T}, \mathbf{y}_{0:T} \sim q}\left[\log \frac{\prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t)}{\prod_{t=T}^{0} p(\mathbf{y}_t|\mathbf{x}_t)}\right] && (8)\\
&= D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) + E_q\left[-\log p(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&\quad + \sum_{t=2}^{T} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&\quad + \sum_{t=0}^{T} D_{\mathrm{KL}}(q(\mathbf{y}_t|\mathbf{x}_t)||p(\mathbf{y}_t|\mathbf{x}_t)). && (9)
\end{aligned}$$

where the first inequality is derived by the Jensen's inequality.

Transition probabilities From [7], the forward transition probabilities are given by

$$\begin{aligned}
q(\mathbf{x}_0) &:= \mathrm{P}_{\text{data}}(\mathbf{x}_0) \quad \text{and} && (10)\\
q(\mathbf{x}_t|\mathbf{x}_{t-1}) &:= \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), && (11)
\end{aligned}$$

where $\{\beta_t\}_{t=1}^{T}$ are predefined constants. The backward transition probabilities are defined as

$$p_\theta(\mathbf{x}_T) := \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \quad \text{and} \tag{12}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{1-\beta_t}}(\mathbf{x}_t + \beta_t s_\theta(\mathbf{x}_t, t)), \beta_t\mathbf{I}\right), \tag{13}$$

where $s_\theta(\cdot, \cdot)$ denotes a neural network parameterized by θ. Due to the following equation,

$$\begin{aligned}
&q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) && (14)\\
&= \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{1-\beta_t}}(\mathbf{x}_t + \beta_t \nabla \log q(\mathbf{x}_t|\mathbf{x}_0)), \bar{\beta}_t\mathbf{I}\right),
\end{aligned}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ and $\bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, the first three terms of Eq. (9) are optimized by the following loss function:

$$\sum_{t=1}^{T} \lambda_t ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||_2^2 + C, \tag{15}$$

where $\epsilon_\theta(\mathbf{x}_t, t) = \dfrac{s_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$ and C is a constant.

Emission probabilities We interpret the last term in Eq. (9) as an observation about whether the state, $\mathbf{x}_t$, is on the real data manifold or not. Then, the emission probability of the forward and backward processes are defined by Bernoulli distributions as

$$q(\mathbf{y}_t|\mathbf{x}_t) := \text{Ber}(1) \;\; \& \;\; p(\mathbf{y}_t|\mathbf{x}_t) := \text{Ber}(D(f(\mathbf{x}_t))), \tag{16}$$

where $f(\cdot)$ is an arbitrary function that projects an input onto a known manifold and $D(\cdot)$ indicates the probability that an input belongs to the manifold of real data. Hence, the KL-divergence of emission, *i.e.*, the last term of Eq. (9), is redefined via two different Bernoulli distributions in Eq. (16). Eventually, the KL-divergence between two emission distributions is replaced by a log-likelihood of the manifold embedding as follows:

$$\sum_{t=0}^{T} D_{\text{KL}}(q(\mathbf{y}_t|\mathbf{x}_t)||p(\mathbf{y}_t|\mathbf{x}_t)) = \sum_{t=0}^{T} -\log(D(f(\mathbf{x}_t))). \tag{17}$$

3.3. Manifold embedding and likelihood function

We now discuss a technically feasible way to implement Eq. (17). The only undecided components in Eq. (17) are the projection function to a known manifold, $f(\cdot)$, and the likelihood function, $D(\cdot)$. We define $f(\cdot)$ as a function projecting $\mathbf{x}_t$ onto a manifold of $\mathbf{x}_{t-s} \sim q_{t-s}$ $(t \geq s)$. With the diffusion model, this can be done by running one discretization step of a numerical ODE solver from the noise level of t to $t-s$, denoted by $\Phi(\mathbf{x}_t, t, s; \theta)$. We implement the projection function using the solver as follows:

$$f_\theta(\mathbf{x}_t) := \hat{\mathbf{x}}_{t-s}^{\theta} = \Phi(\mathbf{x}_t, t, s; \theta). \tag{18}$$

Note that s is a sample drawn from a uniform distribution, $\mathcal{U}(1, \min(t, \lfloor kT \rfloor))$, where $k \in [0, 1]$ is the hyperparameter that determines the lookahead range in the backward direction. We utilize a step of the Euler method[1] or the Heun's method[2] to realize the projection function.

In addition, we design $D(\cdot)$ as $D_\phi(\cdot)^\gamma$, a constant power of a discriminator function, $D_\phi(\cdot)$, which distinguishes between projected data from 1) the prediction of the denoising network and 2) real data. Such a design is motivated by the right-hand side of Eq. (17), which resembles the objective of the generator in non-saturating GAN [5]; we employ a time-dependent discriminator taking the projected data, t and s, as its inputs.

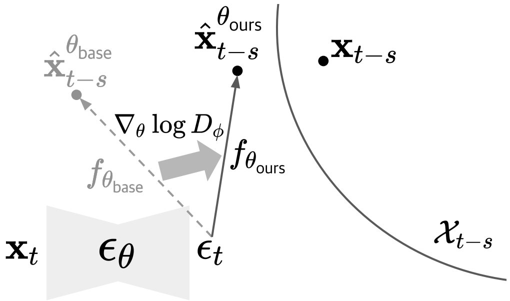


Figure 3. The role of the discriminator in our objective. θ_{ours} and θ_{base} denote the denoising parameters learned by the proposed method and the baseline, respectively. The proposed training method nudges the prediction of $\hat{\mathbf{x}}_{t-s}^{\theta}$ closer to the exact state space than the original.

3.4. Training objectives

By reformulating the transition and emission probabilities as discussed in Secs. 3.2 and 3.3, the first three terms of Eq. (9) become identical to Eq. (15) while the last term is set as $-\gamma \log(D_\phi(\hat{\mathbf{x}}_{t-s}^{\theta}, t, s))$. Therefore, the final training objective of the diffusion model with a network design parametrized by θ is given by

$$\min_\theta \mathbb{E}_{\mathbf{x}_0,\epsilon,t,s}\Big[\underbrace{\lambda_t ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||_2^2}_{\mathcal{L}_{\text{transition}}} - \gamma \underbrace{\log(D_\phi(\hat{\mathbf{x}}_{t-s}^{\theta}, t, s))}_{-\mathcal{L}_{\text{emission}}}\Big], \tag{19}$$

where $D_\phi(\cdot)$ denotes a discriminator and γ is a hyperparameter.

Besides optimizing θ, we need to train $D_\phi(\cdot)$ to distinguish real data from the predictions of the diffusion model. Following GANs [5], the training objective is given by

$$\max_\phi \mathbb{E}_{\mathbf{x}_0,\epsilon,t,s}[\log(D_\phi(\mathbf{x}_{t-s}, t, s)) + \log(1 - D_\phi(\hat{\mathbf{x}}_{t-s}^{\theta}, t, s))]. \tag{20}$$

We perform an alternating optimization, where the two objective functions in Eqs. (19) and (20) take turns until convergence.

For inference, the diffusion model $\epsilon_\theta(\cdot)$ is only taken into account and the discriminator $D_\phi(\cdot)$ is not required. Therefore, our approach incurs no extra computational overhead for sample generations.

[1] Refer to Algorithm 1 in Appendix B.
[2] Refer to Algorithm 2 in Appendix B.

3.5. Analysis of the observation-induced loss

We further analyze the surrogate of the negative log-likelihood of a generated sample and explain how the proposed observation-induced loss affects the surrogate.

3.5.1 Negative log-likelihood at inference

The negative log-likelihood of a generated sample $\mathbf{x}_0 \sim p_\theta$ is given by

$$\begin{aligned}&\mathbb{E}_{\mathbf{x}_0 \sim p_\theta}[-\log q(\mathbf{x}_0)] \qquad (21)\\ &\leq \sum_2^N D_{\text{KL}}\left(p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})||q(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})\right) + \mathbb{E}_{p_\theta}[q(\mathbf{x}_{\tau_0}|\mathbf{x}_{\tau_1})],\end{aligned}$$

where $\tau_0 = 0 < \tau_1 < \cdots < \tau_N = T$ is a subsequence of time steps selected for fast sampling and $\mathbf{x}_0$ is sampled from p_θ unlike Eq. (5). Here, $p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})$ is defined similarly to Eq. (13) by replacing β_t with $\tilde{\beta}_{\tau_i} = 1 - \frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i-1}}}$.

3.5.2 Approximation on true reverse distribution

While $p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})$ takes the tractable form of a Gaussian distribution, the true reverse distribution, $q(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})$, is still infeasible for estimating the KL-divergence in the right-hand side of Eq. (21). To simulate the true reverse density function with $\tilde{\beta}_{\tau_i}$, we use a weighted geometric mean of its asymptotic distributions corresponding to $\tilde{\beta}_{\tau_i} \approx 0$ or 1.

For notational simplicity, let $\mathbf{x}_{\tau_{i-1}} = \mathbf{u}$, $\mathbf{x}_{\tau_t} = \mathbf{v}$, and $\tilde{\beta}_{\tau_i} = \beta$. Then, we denote the true reverse density function as $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, for $\mathbf{u} \sim p_\mathbf{u}$ and $\mathbf{v}|\mathbf{u} \sim \mathcal{N}(\sqrt{1-\beta}\mathbf{u}, \beta\mathbf{I})$.

Lemma 1. *For $\mathbf{u} \sim p_\mathbf{u}$ and $\mathbf{v}|\mathbf{u} \sim \mathcal{N}(\sqrt{1-\beta}\mathbf{u}, \beta\mathbf{I})$, we obtain the following two asymptotic distributions of $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$:*

$$p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) \approx \mathcal{N}\left(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla\log p_\mathbf{u}(\mathbf{v})), \beta\mathbf{I}\right) \quad \textit{for } 0 < \beta \ll 1, \qquad (22)$$

$$\lim_{\beta\to 1^-} p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = p_\mathbf{u}(\mathbf{u}). \qquad (23)$$

Proof. Refer to Appendix A.3. ■

The weighted geometric mean of the two asymptotic distributions in Eqs. (22) and (23) is given by

$$\begin{aligned}&q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) \qquad (24)\\ &:= C_\xi\, \mathcal{N}\left(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla\log p_\mathbf{u}(\mathbf{v})), \beta\mathbf{I}\right)^{1-\xi} p_\mathbf{u}(\mathbf{u})^\xi,\end{aligned}$$

where C_ξ is the normalization constant and ξ determines the weight of each component. We further define a mapping function $\xi(\beta)$ that minimizes the difference between $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ and $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ as

$$\xi(\beta) := \underset{\xi\in[0,1]}{\arg\min} \int_{-\infty}^{\infty} \left(q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) - p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})\right)^2 d\mathbf{u}. \qquad (25)$$

The existence of $\xi(\beta)$ is clear under continuity of $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ with respect to ξ. For the rest of the analysis, we approximate $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ by $q^{(\xi(\beta))}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$; the validity of the approximation is discussed in Appendix A.4 with more detailed empirical study results. Finally, by substituting the variables back as $\mathbf{u} = \mathbf{x}_{\tau_{i-1}}$, $\mathbf{v} = \mathbf{x}_{\tau_i}$, and $\beta = \tilde{\beta}_{\tau_i}$, the true reverse density function is approximated by

$$\begin{aligned}&q(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i}) \qquad (26)\\ &\approx C_{\xi(\tilde{\beta}_{\tau_i})}\mathcal{N}(\mathbf{x}_{\tau_{i-1}}; \boldsymbol{\mu}_{\tau_i}, \tilde{\beta}_{\tau_i}\mathbf{I})^{1-\xi(\tilde{\beta}_{\tau_i})} q(\mathbf{x}_{\tau_{i-1}})^{\xi(\tilde{\beta}_{\tau_i})},\end{aligned}$$

where $\boldsymbol{\mu}_{\tau_i} = \frac{1}{\sqrt{1-\tilde{\beta}_{\tau_i}}}(\mathbf{x}_{\tau_i} + \tilde{\beta}_{\tau_i}\nabla\log q(\mathbf{x}_{\tau_i}))$.

3.5.3 Interpretation of Eq. (21)

By using Eq. (26), we factorize the KL-divergence term in Eq. (21) into a sum of two KL-divergences as follows:

$$\begin{aligned}&D_{\text{KL}}(p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})||q(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i}))\\ &\approx (1-\xi(\tilde{\beta}_{\tau_i}))D_{\text{KL}}(p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})||\mathcal{N}(\mathbf{x}_{\tau_{i-1}}; \boldsymbol{\mu}_{\tau_i}, \tilde{\beta}_{\tau_i}\mathbf{I}))\\ &\quad + \xi(\tilde{\beta}_{\tau_i})D_{\text{KL}}(p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})||q(\mathbf{x}_{\tau_{i-1}})) + \log C_{\xi(\tilde{\beta}_{\tau_i})}\\ &= (1-\xi(\tilde{\beta}_{\tau_i}))||s_\theta(\mathbf{x}_{\tau_i}, \tau_i) - \nabla\log q(\mathbf{x}_{\tau_i})||_2^2\\ &\quad + \xi(\tilde{\beta}_{\tau_i})D_{\text{KL}}(p_\theta(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})||q(\mathbf{x}_{\tau_{i-1}})) + C, \qquad (27)\end{aligned}$$

where C is a constant.

While $\mathcal{L}_{\text{transition}}$ in Eq. (19) minimizes the first term of the last equation in Eq. (27), $\mathcal{L}_{\text{emission}}$ in Eq. (19) minimizes the JS-divergence between two distributions [5]. Although the JS-divergence has different properties from the KL-divergence, both quantities are minimized when two distributions are equal; the minimization of the JS-divergence effectively reduces the second term of Eq. (27) in practice.

Note that the vanilla diffusion models neglect the second term of Eq. (27) while DDGAN [36] disregards the first term of Eq. (27). On the contrary, the proposed method considers both components, leading to effective optimization. In practice, both $\xi(\tilde{\beta}_{\tau_i})$ and $1 - \xi(\tilde{\beta}_{\tau_i})$ are expected to be non-trivial in fast sampling with a relatively large value of $\tilde{\beta}_{\tau_i}$. The behaviors of $\xi(\tilde{\beta}_{\tau_i})$ are demonstrated in Fig. 4 (a) and (b) in Appendix A.4.

4. Experiments

This section first describes the evaluation protocol in this work, and then presents quantitative and qualitative results of OGDM in comparison to other baselines.

Table 1. FID and recall scores for various NFEs when the projection function $f_\theta(\cdot)$ aligns to the sampler as Euler method[1]. 'EDM cond.' denotes that the model is trained with class labels. 'OGDM' represents that the models are trained from scratch while 'OGDM (ft)' indicates that the models are fine-tuned from the pretrained baseline models.

	NFEs	25		20		15		10	
Dataset (Backbone)	Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑
CIFAR-10 (ADM)	Baseline	7.08	0.583	8.05	0.582	9.93	0.567	15.20	0.527
	OGDM	**6.26**	**0.587**	**6.81**	**0.587**	**7.96**	**0.578**	11.63	0.546
	OGDM (ft)	6.69	0.582	7.26	0.581	8.15	0.571	**11.18**	**0.549**
CIFAR-10 (EDM)	Baseline	5.32	0.572	6.82	0.558	10.02	0.524	19.32	0.452
	OGDM (ft)	**3.21**	**0.603**	**3.53**	**0.600**	**4.64**	**0.587**	**9.28**	**0.546**
CIFAR-10 (EDM cond.)	Baseline	4.95	0.567	6.23	0.546	8.81	0.514	15.57	0.434
	OGDM (ft)	**2.56**	**0.599**	**2.83**	**0.589**	**3.67**	**0.572**	**6.85**	**0.528**
CelebA (ADM)	Baseline	7.20	0.441	7.88	0.429	9.34	0.392	11.92	0.315
	OGDM	**3.80**	0.541	**3.94**	0.534	5.06	0.502	7.91	0.451
	OGDM (ft)	4.61	**0.576**	4.61	**0.571**	**4.80**	**0.552**	**7.04**	**0.504**
LSUN Church (LDM)	Baseline	7.87	0.443	8.40	0.434	8.83	0.399	15.02	0.326
	OGDM (ft)	**7.46**	**0.449**	**7.92**	**0.444**	**8.76**	**0.402**	**14.84**	**0.331**

Table 2. FID and recall scores for various NFEs when the projection function $f_\theta(\cdot)$ aligns to the sampler as Heun's method[2]. 'OGDM (ft)' indicates that the models are fine-tuned from the pretrained baseline models.

	NFEs	35		25		19		15		11	
Dataset (Backbone)	Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑
CIFAR-10 (EDM)	Baseline	**2.07**	0.618	2.19	0.616	2.73	0.616	4.48	0.604	14.71	0.536
	OGDM (ft)	2.15	**0.620**	**2.17**	**0.622**	**2.56**	**0.620**	**4.21**	**0.619**	**13.54**	**0.589**

4.1. Evaluation protocol

The datasets, model architectures, and training and evaluation methods are as follows.

Datasets We perform the unconditional image generation experiment on several standard benchmarks with diverse resolutions—CIFAR-10 [16] (32×32), CelebA [19] (64×64) and LSUN Church [37] (256×256).

Architectures We apply our method to three strong baselines, ADM[3] [3] on CIFAR-10 and CelebA, EDM[4] [12] on CIFAR-10, and LDM[5] [25] on LSUN Church, using their official source codes. For the implementation of the time-dependent discriminator, we mostly follow the architecture proposed in Diffusion-GAN [34], which is based on the implementation of StyleGAN2[6] [11] while time indices are injected into the discriminator as in the conditional GAN. The only modification in our implementation is an additional time index, s in Eq. (18), which denotes the number of lookahead time steps from the current time index, t.

Training We use the default hyperparameters and optimization settings provided by the official codes of baseline algorithms for all experiments except for discriminator training. We consistently obtain favorable results with $k \in [0.1, 0.2]$ and $\gamma \in [0.005, 0.025]$ across all datasets and present our choices of the hyperparameters for reproducibility in Appendix C.

Evaluation We measure FID [6] and recall [17] using the implementation provided by ADM[3] for quantitative evaluation. To compute FID, we use the full training data as a reference set and 50K generated images as an evaluation set. For the recall metric, we utilize 50K images for both reference and generated sets.

4.2. Quantitative results

Tabs. 1 and 2 demonstrate quantitative comparison results when projection functions $f_\theta(\cdot)$ aligns with samplers well. They show that a proper combination of a projection function and a sampler substantially improves FID and recall in all cases with NFEs $\leq$ 25. This observation implies that the proposed method yields a robust denoising network for large step sizes. Tab. 1 also compares performance between our models trained from scratch and the ones fine-tuned on

[3] https://github.com/openai/guided-diffusion

[4] https://github.com/NVlabs/edm

[5] https://github.com/CompVis/latent-diffusion

[6] https://github.com/NVlabs/stylegan2-ada-pytorch

Table 3. FID and recall scores for various NFEs when the projection function $f_\theta(\cdot)$ is a step of Euler method[1] while two different PNDM [18] algorithms are used as samplers. 'EDM cond.' denotes that the model is trained with class labels. 'OGDM' represents that the models are trained from scratch while 'OGDM (ft)' indicates that the models are fine-tuned from the pretrained baseline models.

		NFEs	25		20		15		10	
Sampler	Dataset (Backbone)	Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑
S-PNDM	CIFAR-10 (ADM)	Baseline	5.31	0.601	5.95	0.596	7.09	0.577	10.32	0.548
		OGDM	**5.03**	**0.611**	**5.09**	**0.605**	**5.58**	**0.601**	**7.54**	**0.582**
	CIFAR-10 (EDM)	Baseline	**2.74**	**0.604**	**3.21**	0.597	4.48	0.586	9.51	0.531
		OGDM (ft)	3.75	**0.604**	3.62	**0.605**	**3.60**	**0.600**	**4.97**	**0.586**
	CIFAR-10 (EDM cond.)	Baseline	**2.50**	0.606	2.87	0.599	3.76	0.584	6.63	0.544
		OGDM (ft)	2.87	**0.610**	**2.75**	**0.607**	**2.69**	**0.601**	**3.53**	**0.573**
	CelebA (ADM)	Baseline	3.67	0.553	4.15	0.539	5.22	0.511	7.33	0.445
		OGDM	**2.62**	**0.607**	**2.70**	**0.604**	**2.96**	**0.585**	**4.35**	**0.545**
	LSUN Church (LDM)	Baseline	8.41	0.470	8.21	0.471	8.07	0.475	9.14	0.464
		OGDM (ft)	**7.69**	**0.480**	**7.48**	**0.489**	**7.48**	**0.481**	**8.68**	**0.478**
F-PNDM	CIFAR-10 (ADM)	Baseline	**5.17**	0.609	**6.19**	0.600	10.55	0.535	–	–
		OGDM	5.40	**0.609**	6.72	**0.600**	**8.84**	**0.557**	–	–
	CelebA (ADM)	Baseline	3.39	0.562	4.25	0.539	7.08	0.488	–	–
		OGDM	**2.81**	**0.615**	**3.10**	**0.609**	**5.14**	**0.576**	–	–
	LSUN Church (LDM)	Baseline	9.04	0.474	9.10	0.483	12.75	0.493	–	–
		OGDM (ft)	**8.24**	**0.481**	**8.39**	**0.495**	**11.78**	**0.505**	–	–

Table 4. Comparisons of FIDs with other methods on CIFAR-10 (32×32) and CelebA (64×64). '†' means the values are copied from other papers and '*' means the values are obtained by applying DDIM [28] as a sampler.

	CIFAR-10				CelebA			
Method \ NFEs	25	20	15	10	25	20	15	10
DDIM [28] †*	–	6.84	–	13.36	–	13.73	–	17.33
Analytic-DPM [2] †*	5.81	–	–	14.00	9.22	–	–	15.62
FastDPM [14] †*	–	5.05	–	9.90	–	10.69	–	15.31
GENIE [4] †	3.64	3.94	4.49	5.28	–	–	–	–
Watson *et al.* [35] †	4.25	4.72	5.90	7.86	–	–	–	–
ADM [3]	7.08	8.05	9.93	15.20	7.20	7.88	9.34	11.92
EDM [12]	2.19	–	4.48	–	–	–	–	–
S-PNDM [18]	2.74	**3.21**	4.48	9.51	3.67	4.15	5.22	7.33
F-PNDM [18]	5.17	6.19	10.55	–	3.39	4.25	7.08	–
CT [30]	6.94	6.63	6.36	6.20	–	–	–	–
OGDM	**2.17**	3.53	**3.60**	**4.97**	**2.62**	**2.70**	**2.96**	**4.35**

'CIFAR-10 (ADM)' and 'CelebA (ADM)'. We observe that the fine-tuned models exhibit competitive performance with a small fraction (5–10%) of the training iterations when compared to the models optimized through full training. Detailed analysis and comparisons are provided in Tab. 6 of Appendix C. These findings highlight the practicality and computational efficiency of the proposed method.

Tab. 3 presents FID and recall scores in the case that the projection function $f_\theta(\cdot)$ is a step of the Euler method while the samplers are either S-PNDM or F-PNDM. Unless the projection function and the sampler align properly, the benefit from our approach is not guaranteed because the observations may be inaccurately projected onto the manifold from the perspective of the sampler. Despite this reasonable concern, both S-PNDM and F-PNDM still achieve great performance gains especially when NFEs are small. Notably, the combination of the proposed method and S-PNDM shows consistent performance improvements; the models with the Euler projection and S-PNDM harness synergy because the steps of S-PNDM are similar to the Euler method except for the initial step. Moreover, 'CIFAR-10 (EDM cond.)' in Tabs. 1 and 3 shows that our method is still effective for conditional sampling. We also present results when stochastic sampling is employed, with our approach showing superior performance, in Appendix E.

Tab. 4 presents the FIDs of various algorithms combined with fast inference techniques in a wide range of NFEs. The results imply that the proposed approach is advantageous when the number of time steps for inference is small. We additionally compare our approach with other methods that incorporate time-dependent discriminators to diffusion process, *i.e*., DG [13] and DDGAN [36], in Section Appendix D.

4.3. Qualitative results

We discuss the qualitative results of our approach in the following two aspects.

Comparisons to baselines We provide qualitative results on CIFAR-10, CelebA, and LSUN Church obtained by a few sampling steps in comparison with the baseline methods in Appendix F. The baseline models often produce blurry samples when utilizing fast inference methods. In contrast, our models generate crispy and clear images as well as show more diverse colors and tones compared to the corresponding baselines. Moreover, as shown in the left-hand side of Fig. 1, the baseline model generates face images with inconsistent genders by varying NFEs. On the other hand, our model maintains the information accurately, which is desirable results because we use deterministic generative process with the same initial point. This is because the additional loss term of our method enables the model to approximate each backward step more accurately, even with coarse discretization.

Nearest neighborhoods Fig. 21 in Appendix F.5 illustrates the nearest neighbor examples in the training datasets with respect to the generated images by our approach. According to the results, the generated samples are sufficiently different from the training examples, confirming that our models do not simply memorize data but increase scores properly by improving diversity in output images.

4.4. Discussion on training cost

Regarding the trainig cost, our method increases the training time by approximately 80% compared to the baseline, given the same number of iterations. This is primarily due to the additional training cost incurred by the discriminator. However, we can achieve promising results by fine-tuning the pretrained baseline model for only a small number of iterations, as shown in Tabs. 1 to 3, which significantly alleviate the burden of training the discriminator.

5. Future work

Although not explored in this work, there is more room for amplifying the impact of the proposed approach. Using OGDM as a pretrained score model for consistency distillation [30] can be advantageous over using baseline models by enhancing the accuracy of one-step progress in the teacher model. Moreover, integrating the lookahead variable s as an additional input to diffusion models may further improve performance. We can also construct a specialized denoising network for specific sampling steps by focusing more on learning manifolds of noise levels corresponding to the time steps to be used in sampling. This can be realized by selecting s adaptively, rather than adopting a uniform sampling. Moreover, considering that $\xi(\beta)$ is positively correlated with β (see Fig. 4 (b) in Appendix A.4 of the supplementary document), it would make sense to set γ in proportion to $1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-s}}$. While we examine the observations following the Bernoulli distribution and implement them using the discriminators as in GANs, it is important to note that there are other options to define and implement the observation factors. For example, other formulations of the extended surrogate Eq. (9) can be explored, which may include the ones specific to target tasks, available resources, and measurement methods.

6. Conclusion

We presented a diffusion probabilistic model that introduces observations into the Markov chain of [7]. As a feasible and effective way, we have concretized the surrogate loss for negative log-likelihood using observations following the Bernoulli distribution, and integrated the adversarial training loss by adding a discriminator network that simulates the observation. Our strategy regulates the denoising network to minimize the accurate negative log-likelihood surrogate at inference, thereby increasing robustness in a few steps sampling. As a result, our method facilitates faster inference by mitigating quality degradation. We demonstrated the effectiveness of the proposed method on well-known baseline models and multiple datasets.

Acknowledgements This work was partly supported by LG AI Research, and the IITP grants [No. 2022-0-00959, (Part 2) Few-Shot Learning of Causal Inference in Vision and Language for Decision Making; No. 2021-0-02068, AI Innovation Hub (AI Institute, Seoul National University); No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)] funded by the Korea government (MSIT).

References

[1] Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. In *ICML*, 2022. 2

[2] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: An analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *ICLR*, 2022. 2, 7

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021. 1, 6, 7

[4] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In *NeurIPS*, 2022. 2, 7

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2, 4, 5

[6] 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. In *NeurIPS*, 2017. 6

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2, 3, 8

[8] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *ICML*, 2022. 1

[9] Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. Diff-TTS: A denoising diffusion model for text-to-speech. In *INTERSPEECH*, 2021. 1

[10] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021. 2

[11] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020. 6

[12] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022. 2, 6, 7, 13

[13] Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. In *ICML*, 2023. 3, 8, 14

[14] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. 7

[15] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021. 1

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 6

[17] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. 2019. 6

[18] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *ICLR*, 2022. 2, 7

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 6

[20] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 3

[21] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3D point cloud generation. In *CVPR*, 2021. 1

[22] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 3

[23] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021. 2

[24] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, 2020. 1

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 6

[26] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022. 3

[27] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 1, 2

[28] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2, 7

[29] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 2

[30] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 3, 7, 8

[31] Endre Süli and David Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 1 edition, 2003. 2

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 31

[33] Sauer Timothy. *Numerical Analysis*. Pearson, 3 edition, 2017. 2

[34] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-GAN: Training GANs with diffusion. 2023. 3, 6

[35] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *ICLR*, 2022. 2, 7

[36] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In *ICLR*, 2022. 3, 5, 8, 14

[37] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 6

[38] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3D shape generation. In *NeurIPS*, 2022. 1

Observation-Guided Diffusion Probabilistic Models
Supplementary Material

Appendix

A. Details of Sec. 3

A.1. Proof of Eq. (3)

$$
\begin{aligned}
q(\mathbf{x}_{1:T}, \mathbf{y}_{0:T}|\mathbf{x}_0) &= q(\mathbf{y}_0|\mathbf{x}_0) \prod_{t=0}^{T-1} q(\mathbf{x}_{t+1}|\mathbf{x}_{0:t}, \mathbf{y}_{0:t}) q(\mathbf{y}_{t+1}|\mathbf{x}_{0:t+1}, \mathbf{y}_{0:t}) \\
&= q(\mathbf{y}_0|\mathbf{x}_0) \prod_{t=0}^{T-1} q(\mathbf{x}_{t+1}|\mathbf{x}_t) q(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}) \qquad (\because \mathit{Eq.}\ (1)) \\
&= \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t) \prod_{t=0}^{T-1} q(\mathbf{x}_{t+1}|\mathbf{x}_t) \\
&= \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t) \left[q(\mathbf{x}_1|\mathbf{x}_0) \prod_{t=1}^{T-1} q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{x}_0) \right] \qquad (\because \mathit{Eq.}\ (1)) \\
&= \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t) \left[q(\mathbf{x}_1|\mathbf{x}_0) \prod_{t=1}^{T-1} \frac{q(\mathbf{x}_{t+1}, \mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \right] \\
&= \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t) \left[q(\mathbf{x}_1|\mathbf{x}_0) \prod_{t=1}^{T-1} \frac{q(\mathbf{x}_{t+1}|\mathbf{x}_0) q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \right] \\
&= q(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \prod_{t=0}^{T} q(\mathbf{y}_t|\mathbf{x}_t).
\end{aligned}
$$

A.2. Proof of Eq. (4)

$$
\begin{aligned}
p(\mathbf{x}_{T:0}, \mathbf{y}_{T:0}) &= p(\mathbf{x}_T) p(\mathbf{y}_T|\mathbf{x}_T) \prod_{t=T}^{1} p(\mathbf{x}_{t-1}|\mathbf{x}_{T:t}, \mathbf{y}_{T:t}) p(\mathbf{y}_{t-1}|\mathbf{x}_{T:t-1}, \mathbf{y}_{T:t}) \\
&= p(\mathbf{x}_T) p(\mathbf{y}_T|\mathbf{x}_T) \prod_{t=T}^{1} p(\mathbf{x}_{t-1}|\mathbf{x}_t) p(\mathbf{y}_{t-1}|\mathbf{x}_{t-1}) \qquad (\because \mathit{Eq.}\ (1)) \\
&= p(\mathbf{x}_T) \prod_{t=T}^{1} p(\mathbf{x}_{t-1}|\mathbf{x}_t) \prod_{t=T}^{0} p(\mathbf{y}_t|\mathbf{x}_t).
\end{aligned}
$$

A.3. Proof of Lemma 1

Lemma 1. *For* $\mathbf{u} \sim p_{\mathbf{u}}$ *and* $\mathbf{v}|\mathbf{u} \sim \mathcal{N}(\sqrt{1-\beta}\mathbf{u}, \beta\mathbf{I})$*, we obtain the following two asymptotic distributions of* $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$:

$$
p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) \approx \mathcal{N}\left(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla \log p_{\mathbf{u}}(\mathbf{v})), \beta\mathbf{I}\right) \quad \text{for } 0 < \beta \ll 1, \tag{22}
$$

$$
\lim_{\beta \to 1^-} p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = p_{\mathbf{u}}(\mathbf{u}). \tag{23}
$$

Proof of Eq. (22). Let $\mathbf{u}' = \sqrt{1-\beta}\mathbf{u}$. Then,

$$
p_{\mathbf{v}|\mathbf{u}'}(\mathbf{v}|\mathbf{u}') = \mathcal{N}(\mathbf{v}; \mathbf{u}', \beta\mathbf{I}) = (2\pi\beta)^{-d/2} \exp(-\frac{1}{2\beta}||\mathbf{v} - \mathbf{u}'||^2) = \mathcal{N}(\mathbf{u}'; \mathbf{v}, \beta\mathbf{I}) = q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}).
$$

By talyor expansion of $p_{\mathbf{u}'}(\mathbf{u}')$ at $\mathbf{v}$,

$$p_{\mathbf{u}'}(\mathbf{u}') = p_{\mathbf{u}'}(\mathbf{v}) + < \nabla p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} > + O(||\mathbf{u}' - \mathbf{v}||^2).$$

Then,

$$\begin{aligned} \int q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) p_{\mathbf{u}'}(\mathbf{u}') d\mathbf{u}' &= \mathbb{E}_{\mathbf{u}' \sim \mathcal{N}(\mathbf{v}, \beta\mathbf{I})}[p_{\mathbf{u}'}(\mathbf{v}) + < \nabla p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} > + O(||\mathbf{u}' - \mathbf{v}||^2)] \\ &= p_{\mathbf{u}'}(\mathbf{v}) + O(\beta). \end{aligned}$$

By Bayes' rule and above result,

$$\begin{aligned} p_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) &= \frac{p_{\mathbf{v}|\mathbf{u}'}(\mathbf{v}|\mathbf{u}') p_{\mathbf{u}'}(\mathbf{u}')}{\int p_{\mathbf{v}|\mathbf{u}'}(\mathbf{v}|\mathbf{u}') p_{\mathbf{u}'}(\mathbf{u}') d\mathbf{u}'} \\ &= \frac{q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) p_{\mathbf{u}'}(\mathbf{u}')}{\int q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) p_{\mathbf{u}'}(\mathbf{u}') d\mathbf{u}'} \\ &= q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) \frac{p_{\mathbf{u}'}(\mathbf{v}) + < \nabla p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} > + O(||\mathbf{u}' - \mathbf{v}||^2)}{p_{\mathbf{u}'}(\mathbf{v}) + O(\beta)} \\ &= q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v})(1 + < \frac{\nabla p_{\mathbf{u}'}(\mathbf{v})}{p_{\mathbf{u}'}(\mathbf{v})}, \mathbf{u}' - \mathbf{v} > + O(||\mathbf{u}' - \mathbf{v}||^2))(1 + O(\beta)) \\ &= q_{\mathbf{u}'|\mathbf{v}}(\mathbf{u}'|\mathbf{v}) \exp(< \nabla \log p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} >) + O(\beta) \\ &= (2\pi\beta)^{-d/2} \exp(-\frac{1}{2\beta}||\mathbf{v} - \mathbf{u}'||^2) \exp(< \nabla \log p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} >) + O(\beta) \\ &= (2\pi\beta)^{-d/2} \exp(-\frac{1}{2\beta}(||\mathbf{v} - \mathbf{u}'||^2 - 2\beta < \nabla \log p_{\mathbf{u}'}(\mathbf{v}), \mathbf{u}' - \mathbf{v} >)) + O(\beta) \\ &= (2\pi\beta)^{-d/2} \exp(-\frac{1}{2\beta}||\mathbf{u}' - \mathbf{v} - \beta \nabla \log p_{\mathbf{u}'}(\mathbf{v})||^2 + O(\beta)) + O(\beta) \\ &\approx \mathcal{N}(\mathbf{u}'; \mathbf{v} + \beta \nabla \log p_{\mathbf{u}'}(\mathbf{v}), \beta\mathbf{I}) \text{ for } \beta \ll 1. \end{aligned}$$

From $\mathbf{u}' = \sqrt{1-\beta}\mathbf{u}$,

$$\begin{aligned} p_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = \mathcal{N}(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta \nabla \log p_{\mathbf{u}}(\mathbf{v})), \frac{\beta}{1-\beta}\mathbf{I}) \\ \approx \mathcal{N}(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta \nabla \log p_{\mathbf{u}}(\mathbf{v})), \beta\mathbf{I}) \text{ for } \beta \ll 1 \end{aligned} \qquad \blacksquare$$

Proof of Eq. (23).

$$\begin{aligned} \lim_{\beta \to 1^-} p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u}) &= \lim_{\beta \to 1^-} (2\pi\beta)^{-d/2} \exp(-\frac{1}{2\beta}||\mathbf{v} - \sqrt{1-\beta}\mathbf{u}||^2) \\ &= (2\pi)^{-d/2} \exp(-\frac{1}{2}||\mathbf{v}||^2) := f(\mathbf{v}). \end{aligned}$$

From Bayes' rule and above results,

$$\begin{aligned} \therefore \lim_{\beta \to 1^-} p_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) &= \lim_{\beta \to 1^-} \frac{p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u}) p_{\mathbf{u}}(\mathbf{u})}{\int p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u}) p_{\mathbf{u}}(\mathbf{u}) d\mathbf{u}} \\ &= \frac{\lim_{\beta \to 1^-} p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u}) p_{\mathbf{u}}(\mathbf{u})}{\int \lim_{\beta \to 1^-} p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u}) p_{\mathbf{u}}(\mathbf{u}) d\mathbf{u}} \\ &= \frac{f(\mathbf{v}) p_{\mathbf{u}}(\mathbf{u})}{\int f(\mathbf{v}) p_{\mathbf{u}}(\mathbf{u}) d\mathbf{u}} \\ &= \frac{p_{\mathbf{u}}(\mathbf{u})}{\int p_{\mathbf{u}}(\mathbf{u}) d\mathbf{u}} = p_{\mathbf{u}}(\mathbf{u}) \end{aligned} \qquad \blacksquare$$

A.4. Behaviors of $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, and $\xi(\beta)$

In this section, we justify the approximation on the density function of reverse distribution by showing behaviors of $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, and $\xi(\beta)$ on toy examples.

Followings are the definitions in Sec. 3.5. For $\mathbf{u} \sim p_{\mathbf{u}}$ and $\mathbf{v}|\mathbf{u} \sim \mathcal{N}(\sqrt{1-\beta}\mathbf{u}, \beta\mathbf{I})$, $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ is a real backward density function,

$$q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = C(\mathcal{N}(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla \log p_{\mathbf{u}}(\mathbf{v})), \beta\mathbf{I}))^{1-\xi} p(\mathbf{u})^{\xi},$$

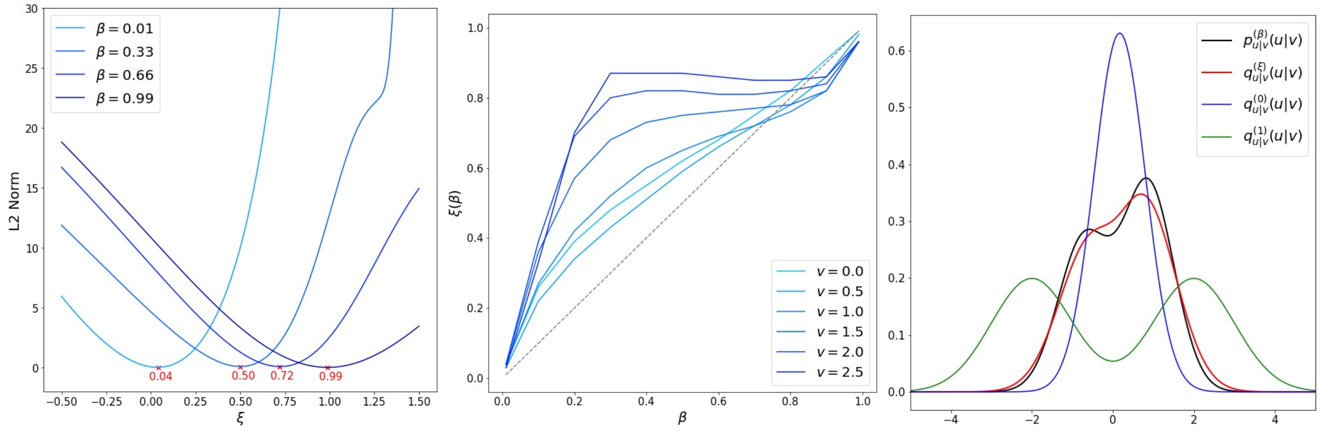
and

$$\xi(\beta) \in \operatorname*{arg\,min}_{\xi \in [0,1]} \int_{-\infty}^{\infty} (q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) - p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}))^2 d\mathbf{u}.$$

For arbitrary $\mu > 0$, let

$$\begin{aligned} p_{\mathbf{u}}(\mathbf{u}) &= \frac{1}{2}(\mathcal{N}(\mathbf{u}; \mu, 1) + \mathcal{N}(\mathbf{u}; -\mu, 1)) \\ &= \frac{1}{2}(2\pi)^{-1/2}(\exp(-(\mathbf{u}-\mu)^2/2) + \exp(-(\mathbf{u}+\mu)^2/2)). \end{aligned}$$

Then, $\mathcal{N}(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla \log p_{\mathbf{u}}(\mathbf{v})), \beta)$, and $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = \dfrac{p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u})p_{\mathbf{u}}(\mathbf{u})}{\int p_{\mathbf{v}|\mathbf{u}}(\mathbf{v}|\mathbf{u})p_{\mathbf{u}}(\mathbf{u})d\mathbf{u}}$ can be explicitly expressed. Moreover, $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ can be calculated numerically. Therefore, we can numerically calculate ℓ^2-norm between $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ and $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$. Finally, we can obtain $\xi(\beta)$ for each $\mathbf{v}$ and β.



(a)

(b)

(c)

Figure 4. Simulations when $\mu = 2$. (a) ℓ^2-norm between $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$, and $q^{(\xi)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ with respect to ξ when $\mathbf{v} = 0.1$. (b) The graph of $\xi(\beta)$ with respect to β for various $\mathbf{v}$. (c) Pdfs of four distributions when $\mathbf{v} = 0.1$, $\beta = 0.4$, and $\xi = \xi(\beta) = 0.55$.

Fig. 4(a) shows the that $q^{(\xi(\beta))}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ better approximates $p^{(\beta)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v})$ than $q^{(1)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = p_{\mathbf{u}}(\mathbf{u})$ and $q^{(0)}_{\mathbf{u}|\mathbf{v}}(\mathbf{u}|\mathbf{v}) = \mathcal{N}(\mathbf{u}; \frac{1}{\sqrt{1-\beta}}(\mathbf{v} + \beta\nabla \log p_{\mathbf{u}}(\mathbf{v})), \beta)$. We can observe that $0 < \xi(\beta) < 1$ for $\forall\beta \in (0, 1)$ from Fig. 4(b). In Fig. 4(c), the black line is the precise reverse distribution while the blue line is asymptotic function when $\beta \to 0^+$ and the green line is the limit when $\beta \to 1^-$. Note that the blue line is the approximation used in vanilla diffusion models. The red line which is a normalized weighted geometric mean of blue and green lines better approximates the real distribution.

B. Numerical ODE solvers

Given discretization steps of $T = t_N > t_{k-1} > \cdots > t_0 = 0$, Algorithms 1 and 2 caculate the numerical solution at $t = 0$ with initial condition $\mathbf{x}_T$ for the following ODE:

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt. \tag{28}$$

Algorithm 1: Euler Method

```
for $i = N, \cdots, 1$ do
    $\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} + (t_{i-1} - t_i)f(\mathbf{x}_{t_i}, t_i)$
end
return $\mathbf{x}_0$
```

Algorithm 2: Heun's Method

```
for $i = N, \cdots, 1$ do
    $\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} + (t_{i-1} - t_i)f(\mathbf{x}_{t_i}, t_i)$
    if $i > 1$ then
        $\mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} + (t_{i-1} - t_i)(\frac{1}{2}f(\mathbf{x}_{t_i}, t_i) + \frac{1}{2}f(\mathbf{x}_{t_{i-1}}, t_{i-1}))$
    end
end
return $\mathbf{x}_0$
```

C. Hyperparameters for experiments

Table 5. Hyperparameters for Training

Dataset	Backbone	Training	Projection	k	γ	Batch size	Seen Images
CIFAR-10	ADM	Baseline	–	–	–	128	38M
		OGDM	Euler	0.1	0.01	128	38M
		OGDM (ft)	Euler	0.2	0.025	128	2M
	EDM	Baseline	–	–	–	512	200M
		OGDM (ft)	Euler	0.2	0.025	512	20M
		OGDM (ft)	Heun's	0.2	0.005	512	20M
CelebA	ADM	Baseline	–	–	–	128	38M
		OGDM	Euler	0.1	0.01	128	38M
		OGDM (ft)	Euler	0.2	0.025	128	2M
LSUN Church	LDM	Baseline	–	–	–	96	48M
		OGDM (ft)	Euler	0.1	0.01	96	1.5M

Table 6. Hyperparameters for sampling

Dataset	Backbone	Sampler	Discretization
CIFAR-10	ADM	Euler	quadratic
		S-PNDM	linear
		F-PNDM	linear
	EDM	Euler	default of [12]
		Heun's	default of [12]
		S-PNDM	default of [12]
CelebA	ADM	Euler	linear
		S-PNDM	linear
		F-PNDM	linear
LSUN Church	LDM	Euler	linear
		S-PNDM	quadratic
		F-PNDM	quadratic

D. Comparisons with DG and DDGAN

In this section, we compare OGDM with DG [13] and DDGAN [36]; they adopt time-dependent discriminators in either training or inference.

D.1. Comparison with DG

Table 7. FID and recall scores of two samplers: Heun's method, DG sampler. Two samplers are applied to vanilla diffusion models and OGDM for variuos discretization steps. For each step, diffusion models are evaluated twice and the gradient of a discriminator is calculated once for DG sampler. The time to calculate the gradient of the discriminator requires about 180% time of evaluating the diffusion models, but it is regarded as NFEs = 1 in the table. The number of steps (n) are chosen by $\arg\min_n 3n - 2 \geq (35, 25, 20, 15)$.

	# steps (n)	13		9		8		6		
Dataset (Backbone)	Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	NFEs
CIFAR-10 (EDM)	Baseline	2.19	0.616	3.33	0.615	4.48	0.604	15.69	0.535	$2n-1$
	Baseline + DG [13]	**1.99**	0.630	4.62	0.613	7.39	0.586	24.78	0.465	$3n-2$
	OGDM (ft)	2.17	0.622	**2.99**	0.622	**4.21**	**0.619**	**13.59**	**0.591**	$2n-1$
	OGDM (ft) + DG [13]	2.00	**0.633**	3.58	**0.624**	5.77	0.616	19.25	0.560	$3n-2$

DG utilizes discriminator to improve the sample quality without taking sampling efficiency into account. Tab. 7 demonstrates quantitative comparison results between DG and OGDM. DG does not work well for fast sampling, rather it deteriorates the sample quality of a few-step-sampler. Moreover, DG requires more computational cost for each update since it incorporates gradient calculation.

D.2. Comparison with DDGAN

Table 8. FID and recall scores of DDGAN [36] and OGDM on CIFAR-10. '†' means the values are copied from the literature.

NFEs	20		8		4		2		1	
Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑
DDGAN [36]†	–	–	**4.36**	0.56	3.75	0.57	4.08	0.54	14.60	0.19
OGDM	3.53	0.60	6.16	**0.58**	–	–	–	–	–	–

DDGAN parametrizes the reverse distribution by neural network without Gaussian assumption. Its objective is not log-likelihood driven and therefore it is more a variant of GANs rather than a diffusion model. Tab. 8 displays quantitative comparison results between DDGAN and OGDM. OGDM achieves better FID and recall scores although DDGAN requires fewer steps. Note that the performance of DDGAN peaks at NFEs = 4 and then drops as the number of time steps increases, which limits the improvement of the model at the expense of increased sampling cost. Moreover, the low recall scores of DDGAN imply it may share the limitations of GANs such as low diversity and mode collapse. Also, DDGAN does not support deterministic sampling which makes it hard to solve problems such as inversion.

E. Stochastic sampling

Table 9. FID and recall scores for various NFEs of stochastic sampler where the projection function is Euler method[1].

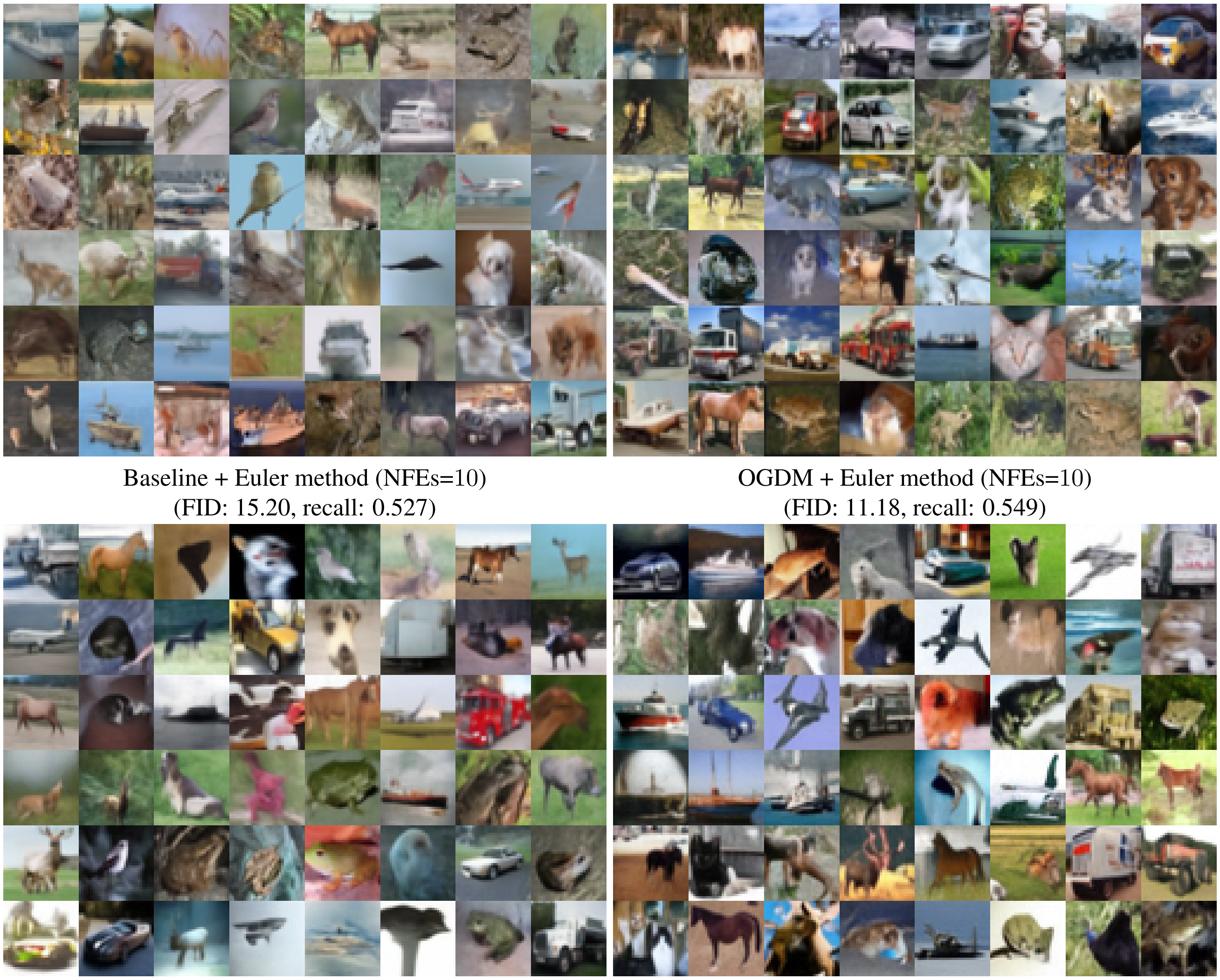
	NFEs	50		20		10	
Dataset (Backbone)	Method	FID↓	Rec.↑	FID↓	Rec.↑	FID↓	Rec.↑
CIFAR-10 (ADM)	Baseline	14.28	0.491	25.42	0.388	44.37	0.278
	OGDM	**9.94**	**0.524**	**16.84**	**0.450**	**29.70**	**0.359**
CelebA (ADM)	Baseline	13.51	0.312	21.00	0.181	31.09	0.079
	OGDM	**9.62**	**0.400**	**15.74**	**0.277**	**24.22**	**0.158**

Tab. 9 presents the quantitative results of stochastic sampling when the projection function is a step of Euler method. We observe that the FID and recall are improved for all cases.

F. Qualitative comparisons

We compare the generated images between the baseline and our method using few number of NFEs in Figs. 5 to 20. While we use Euler method and PNDM for sampling in common, for CIFAR-10, we further compare the results on EDM backbone sampled by Heun's method. The images generated by our method have more vivid color and clearer and less prone to produce unrealistic samples compared to the baselines. Also, our method complements advanced samplers, other than Euler method, effectively. In addition, Fig. 21 in Appendix F.5 illustrates the nearest neighbor examples of our generated examples in training data of CelebA and LSUN Church.

F.1. CIFAR-10 samples with ADM baseline



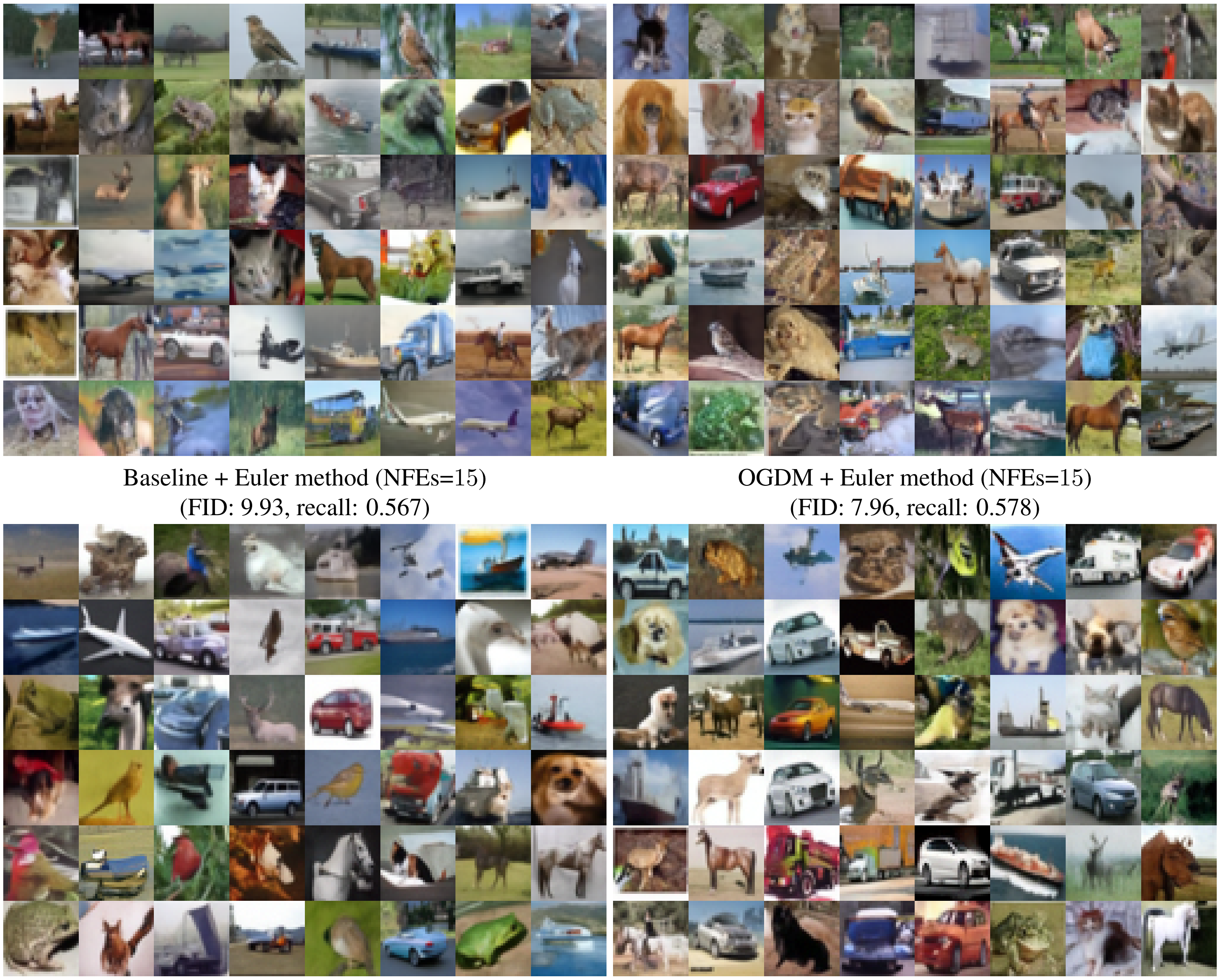
Baseline + Euler method (NFEs=10)
(FID: 15.20, recall: 0.527)

OGDM + Euler method (NFEs=10)
(FID: 11.18, recall: 0.549)

Baseline + S-PNDM (NFEs= 10)
(FID: 10.32, recall: 0.548)

OGDM + S-PNDM (NFEs= 10)
(FID: 7.54, recall: 0.582)

Figure 5. Qualitative results on CIFAR-10 dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=10.



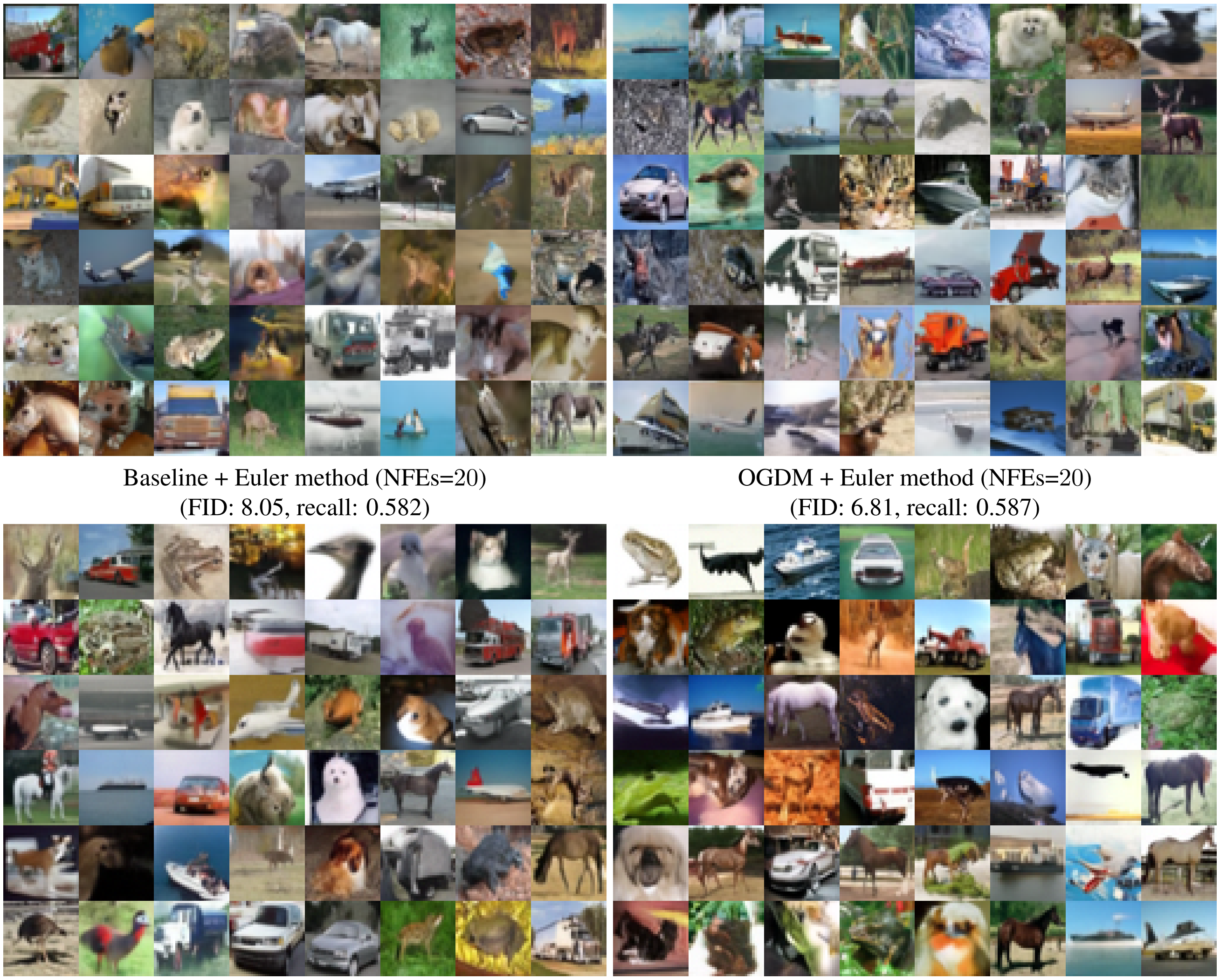
Baseline + Euler method (NFEs=15)
(FID: 9.93, recall: 0.567)

OGDM + Euler method (NFEs=15)
(FID: 7.96, recall: 0.578)

Baseline + S-PNDM (NFEs= 15)
(FID: 7.09, recall: 0.577)

OGDM + S-PNDM (NFEs= 15)
(FID: 5.58, recall: 0.601)

Figure 6. Qualitative results on CIFAR-10 dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=15.



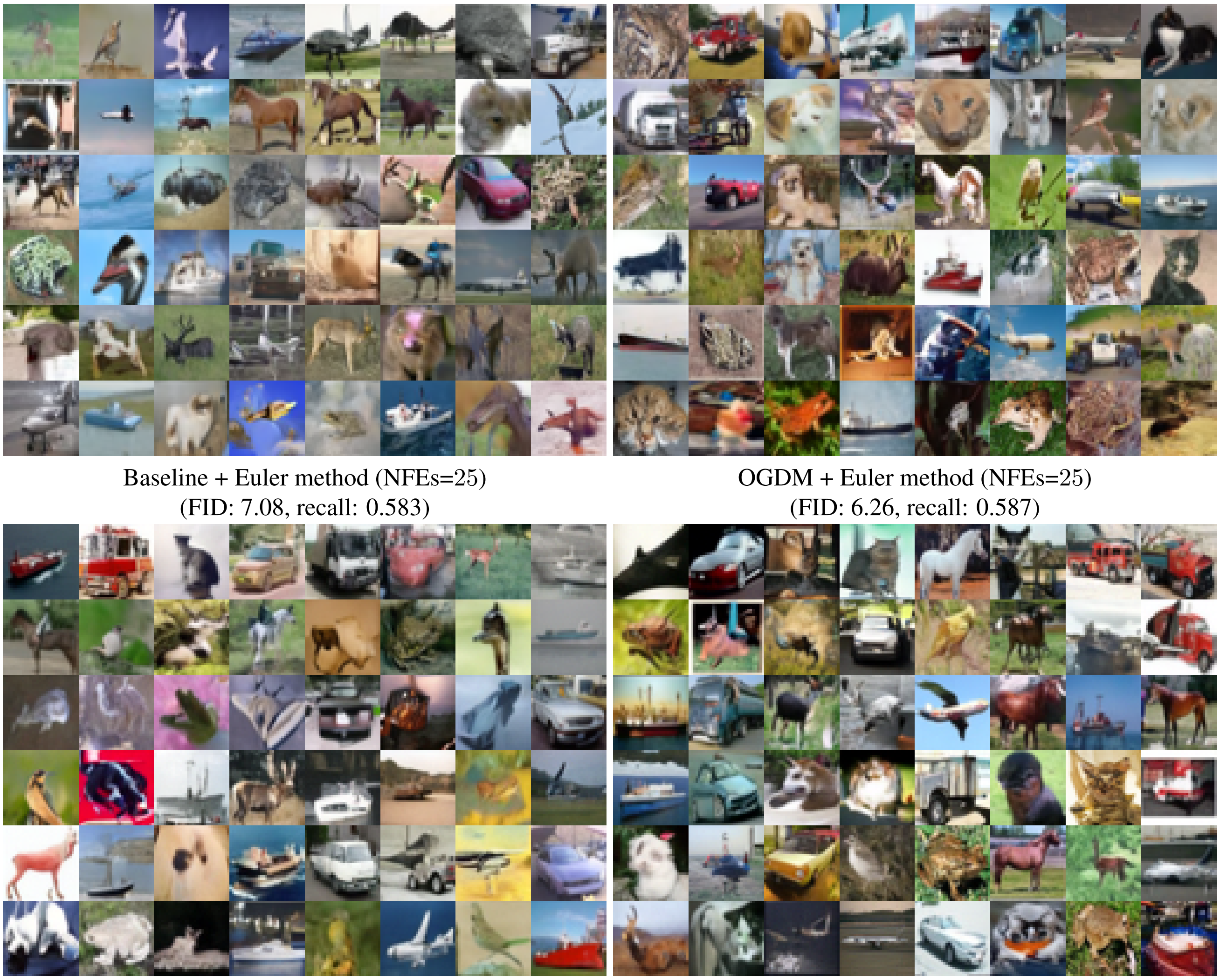
Baseline + Euler method (NFEs=20)
(FID: 8.05, recall: 0.582)

OGDM + Euler method (NFEs=20)
(FID: 6.81, recall: 0.587)

Baseline + S-PNDM (NFEs= 20)
(FID: 5.95, recall: 0.596)

OGDM + S-PNDM (NFEs= 20)
(FID: 5.09, recall: 0.605)

Figure 7. Qualitative results on CIFAR-10 dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=20.



Baseline + Euler method (NFEs=25)
(FID: 7.08, recall: 0.583)

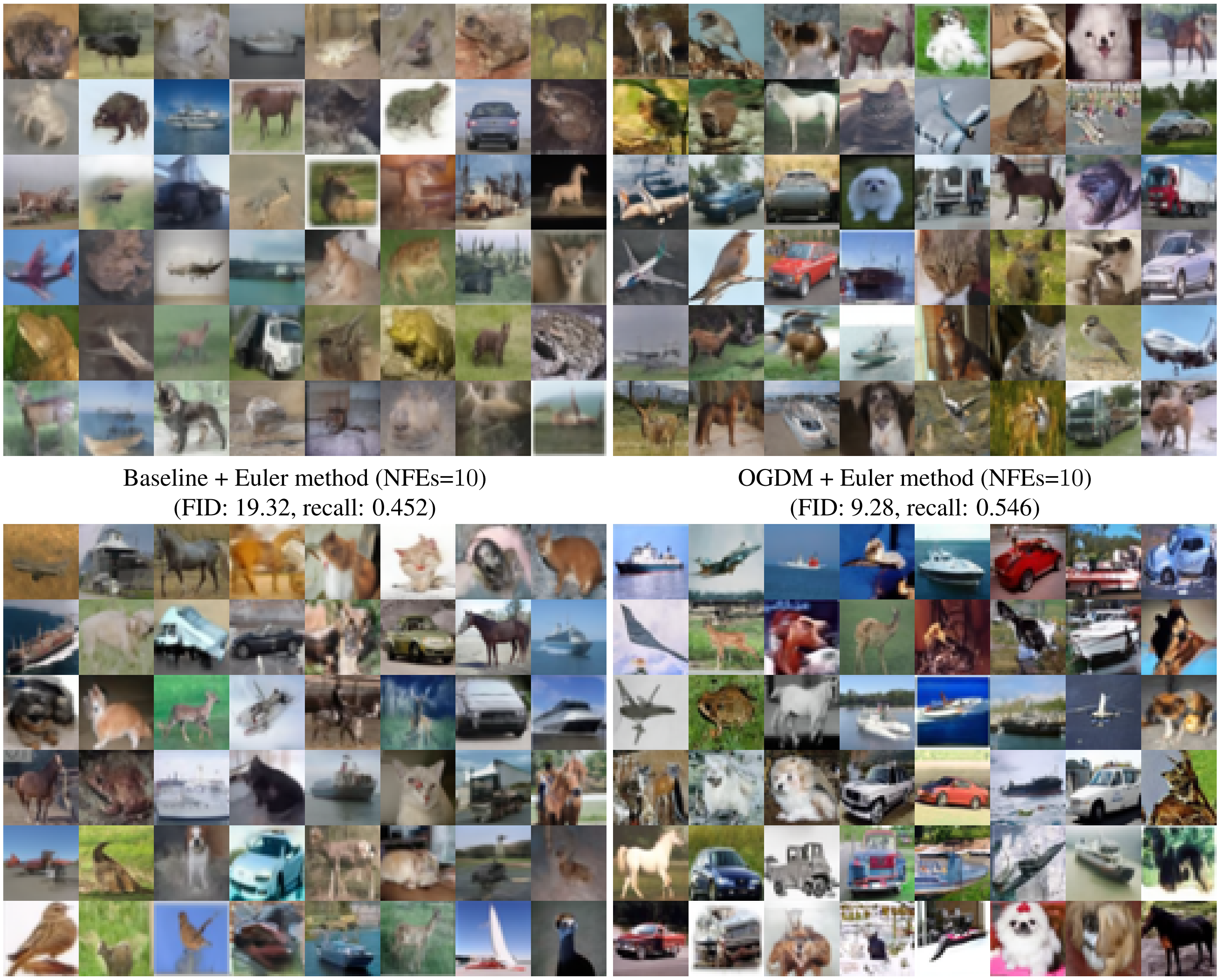
OGDM + Euler method (NFEs=25)
(FID: 6.26, recall: 0.587)

Baseline + S-PNDM (NFEs= 25)
(FID: 5.31, recall: 0.601)

OGDM + S-PNDM (NFEs= 25)
(FID: 5.03, recall: 0.611)

Figure 8. Qualitative results on CIFAR-10 dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=25.

F.2. CIFAR-10 samples with EDM baseline



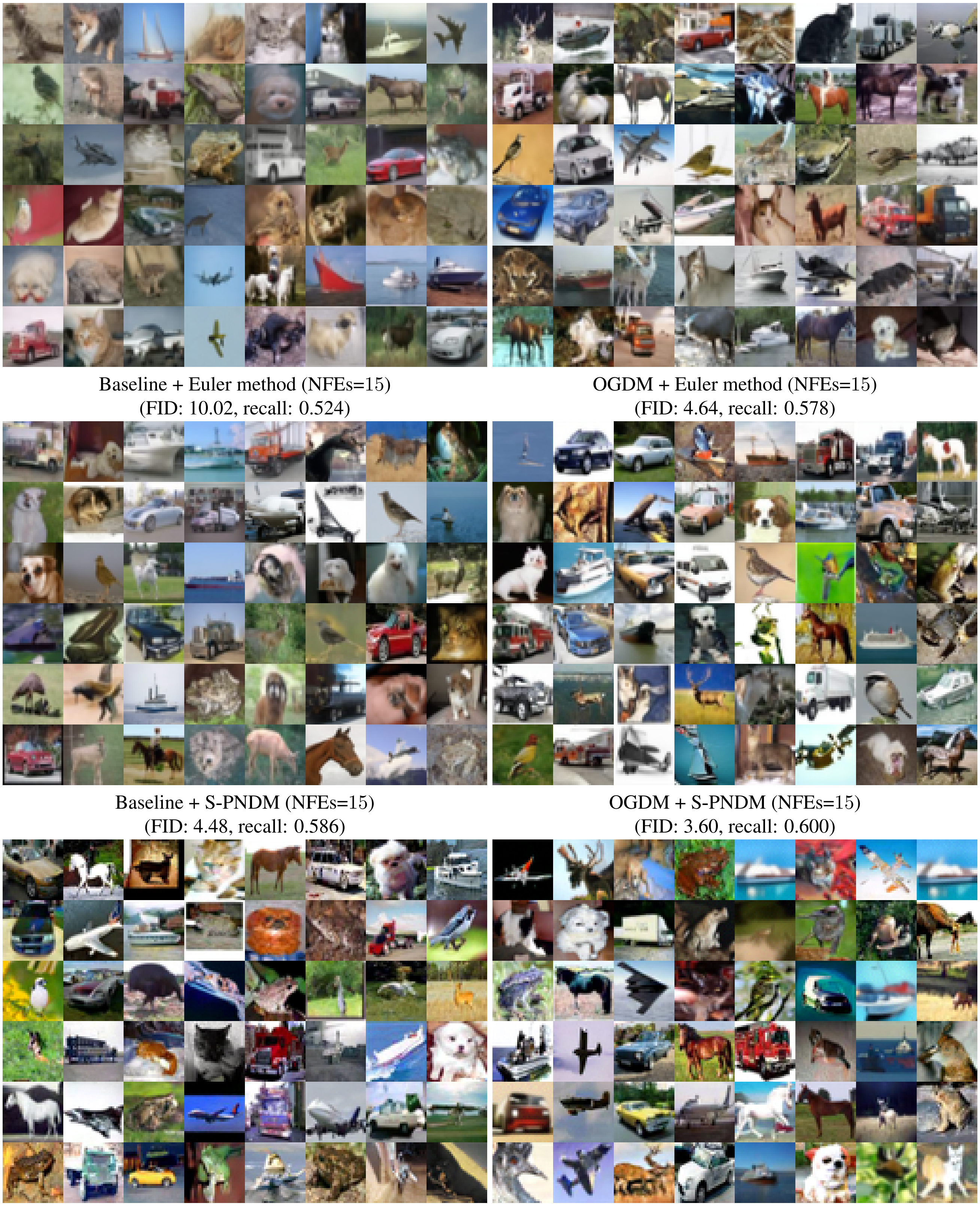
Baseline + Euler method (NFEs=10)
(FID: 19.32, recall: 0.452)

OGDM + Euler method (NFEs=10)
(FID: 9.28, recall: 0.546)

Baseline + S-PNDM (NFEs=10)
(FID: 9.51, recall: 0.531)

OGDM + S-PNDM (NFEs=10)
(FID: 4.97, recall: 0.586)

Figure 9. Qualitative results on CIFAR-10 dataset with the EDM backbone using Euler method (top), and S-PNDM (bottom) with NFEs=10.



Baseline + Euler method (NFEs=15)
(FID: 10.02, recall: 0.524)

OGDM + Euler method (NFEs=15)
(FID: 4.64, recall: 0.578)

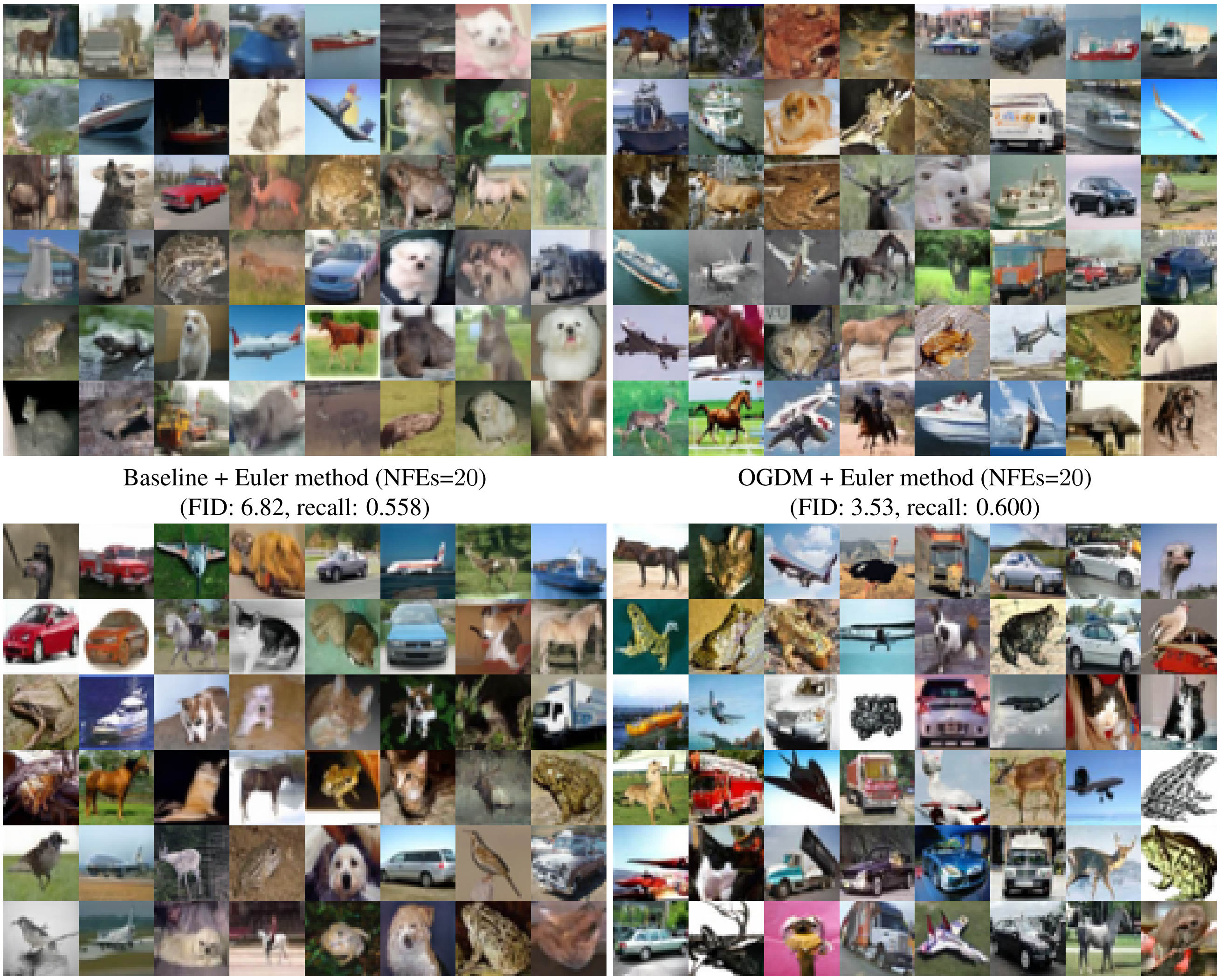
Baseline + S-PNDM (NFEs=15)
(FID: 4.48, recall: 0.586)

OGDM + S-PNDM (NFEs=15)
(FID: 3.60, recall: 0.600)

Baseline + Heun's method (NFEs=15)
(FID: 4.48, recall: 0.604)

OGDM + Heun's method (NFEs=15)
(FID: 4.21, recall: 0.619)

Figure 10. Qualitative results on CIFAR-10 dataset with the EDM backbone using Euler method (top), S-PNDM (middle) and Heun's method (bottom) with NFEs=15.



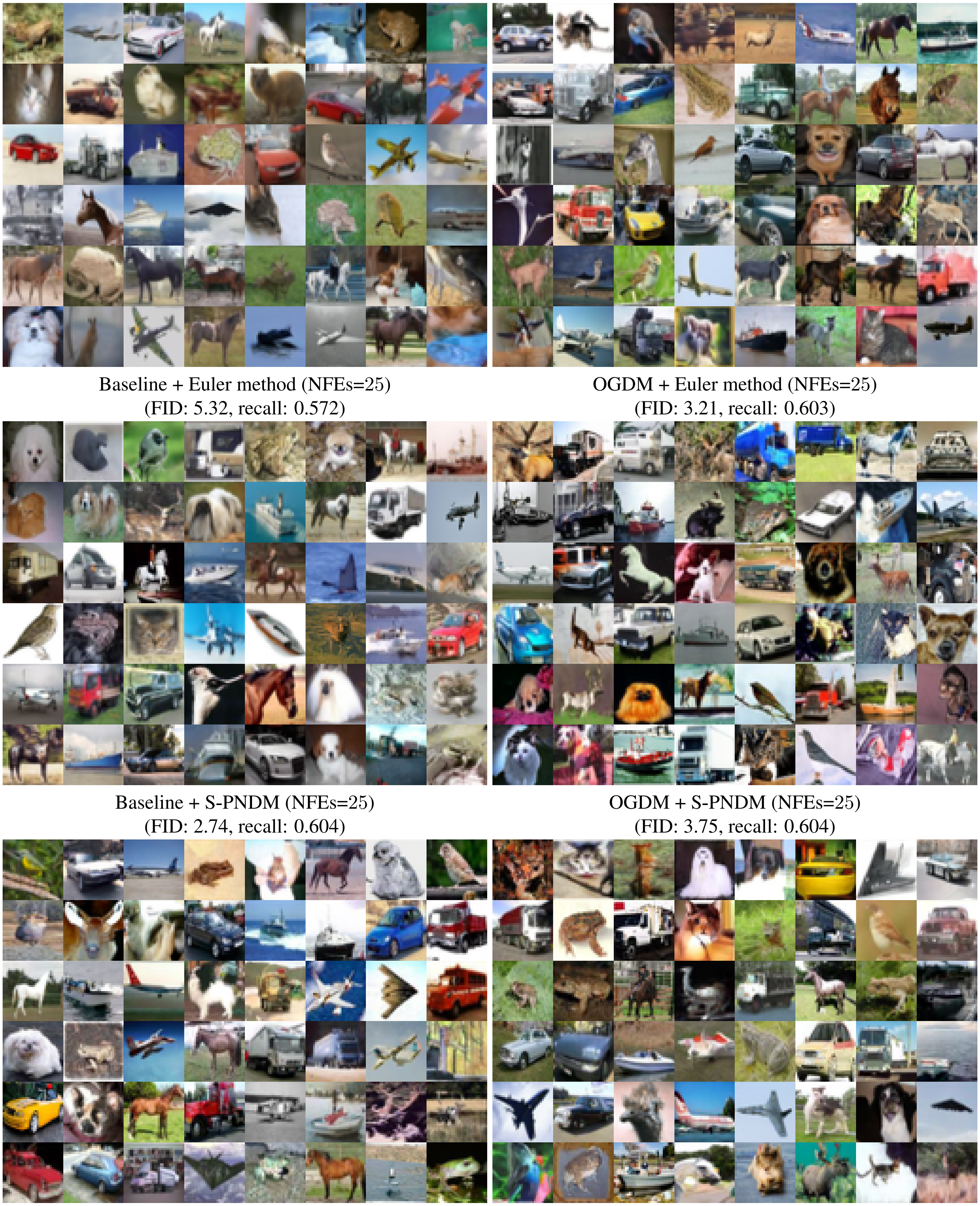
Baseline + Euler method (NFEs=20)
(FID: 6.82, recall: 0.558)

OGDM + Euler method (NFEs=20)
(FID: 3.53, recall: 0.600)

Baseline + S-PNDM (NFEs=20)
(FID: 3.21, recall: 0.597)

OGDM + S-PNDM (NFEs=20)
(FID: 3.62, recall: 0.605)

Figure 11. Qualitative results on CIFAR-10 dataset with the EDM backbone using Euler method (top), and S-PNDM (bottom) with NFEs=20.



Baseline + Euler method (NFEs=25)
(FID: 5.32, recall: 0.572)

OGDM + Euler method (NFEs=25)
(FID: 3.21, recall: 0.603)

Baseline + S-PNDM (NFEs=25)
(FID: 2.74, recall: 0.604)

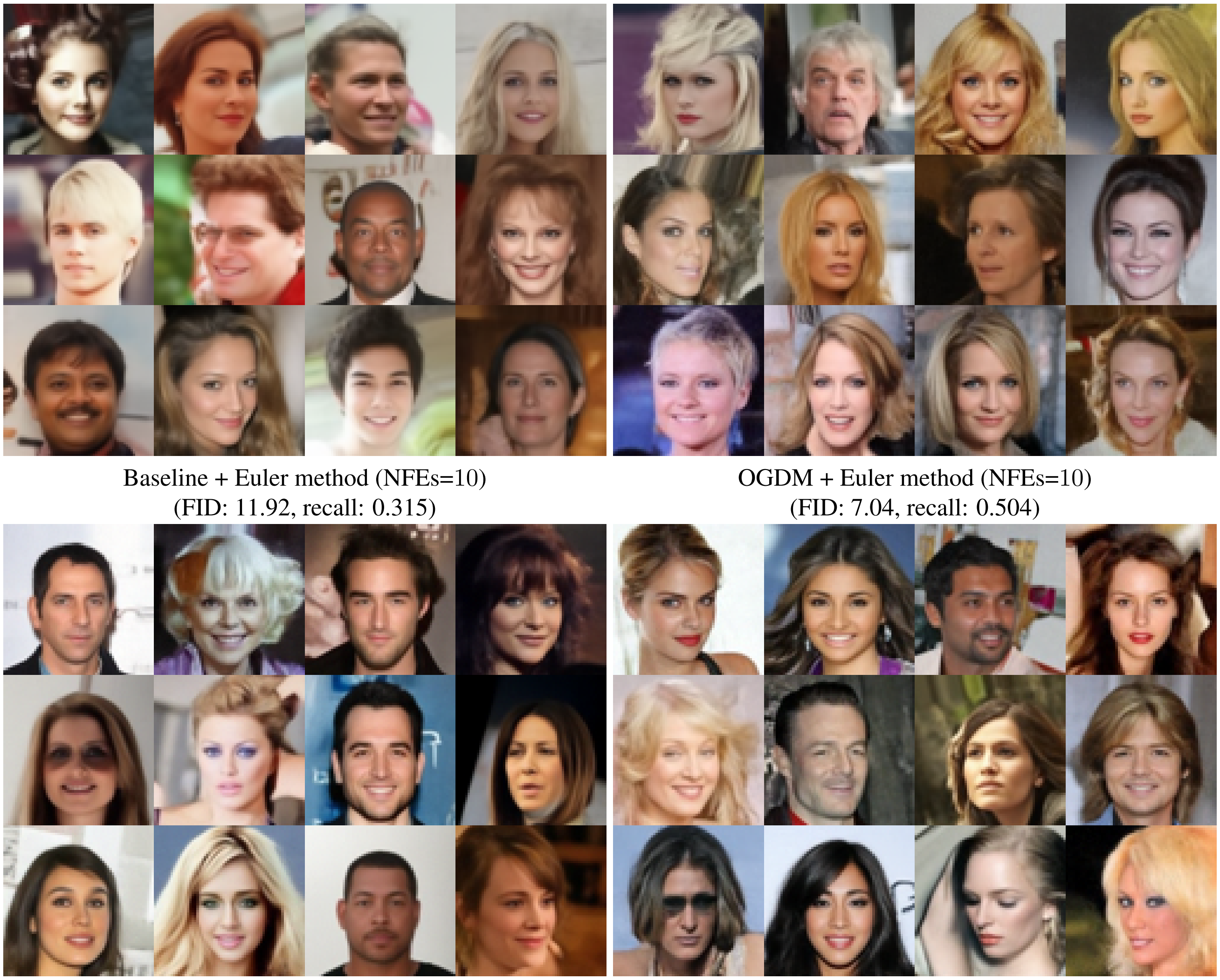
OGDM + S-PNDM (NFEs=25)
(FID: 3.75, recall: 0.604)

Baseline + Heun's method (NFEs=25)
(FID: 2.19, recall: 0.616)

OGDM + Heun's method (NFEs=25)
(FID: 2.17, recall: 0.622)

Figure 12. Qualitative results on CIFAR-10 dataset with the EDM backbone using Euler method (top), S-PNDM (middle) and Heun's method (bottom) with NFEs=25.

F.3. CelebA samples with ADM baseline



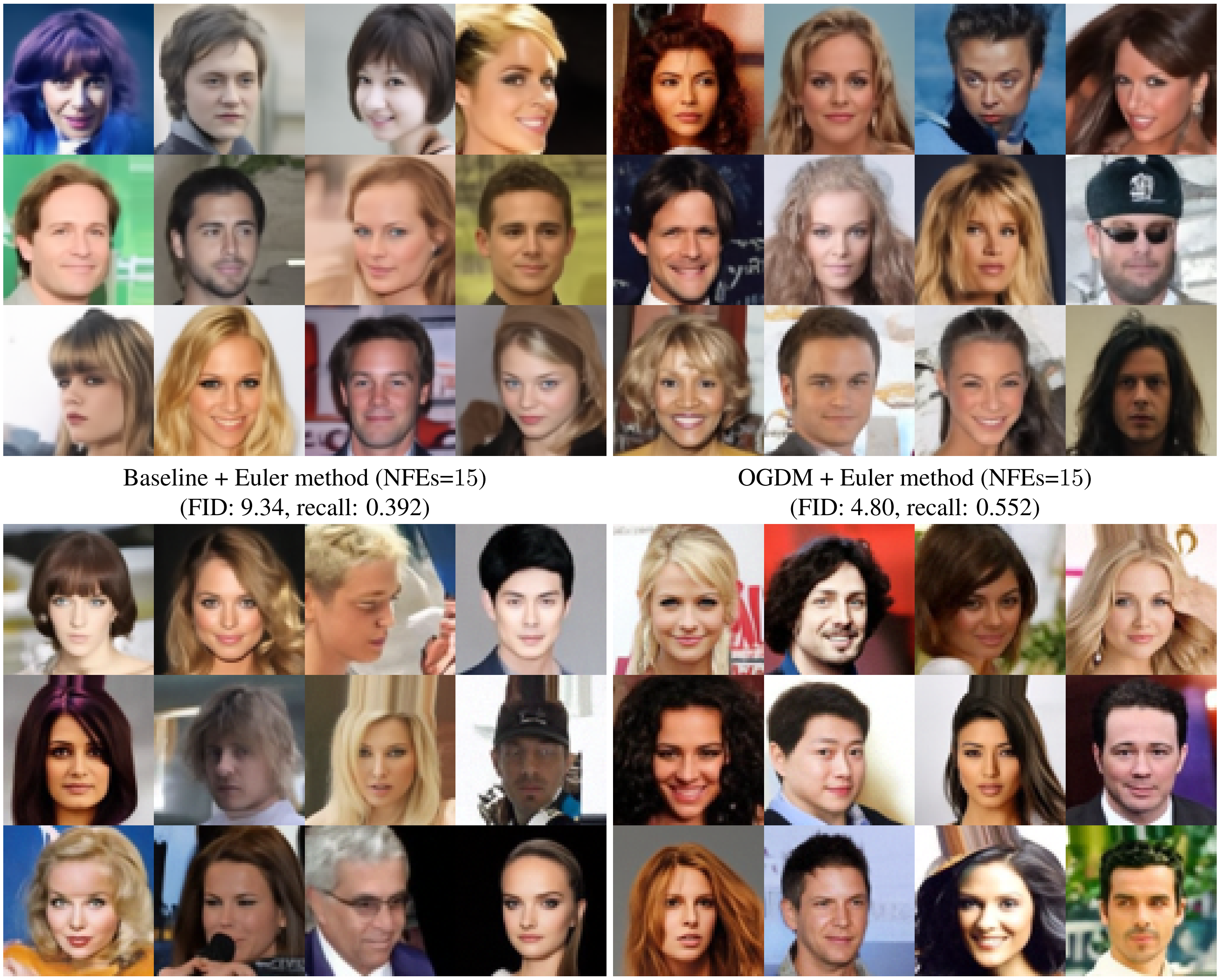
Baseline + Euler method (NFEs=10)
(FID: 11.92, recall: 0.315)

OGDM + Euler method (NFEs=10)
(FID: 7.04, recall: 0.504)

Baseline + S-PNDM (NFEs=10)
(FID: 7.33, recall: 0.445)

OGDM + S-PNDM (NFEs=10)
(FID: 4.35, recall: 0.545)

Figure 13. Qualitative results on CelebA dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=10.



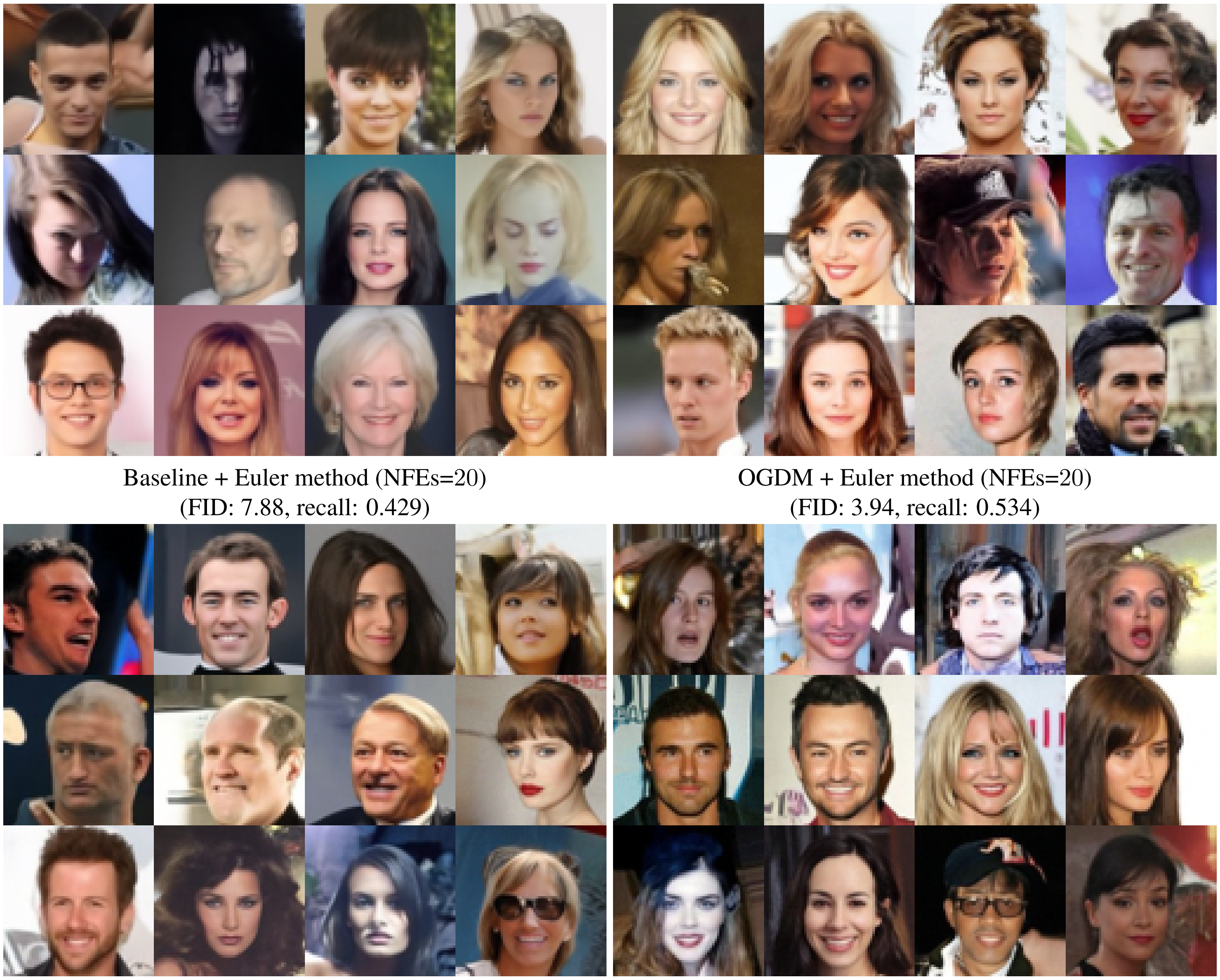
Baseline + Euler method (NFEs=15)
(FID: 9.34, recall: 0.392)

OGDM + Euler method (NFEs=15)
(FID: 4.80, recall: 0.552)

Baseline + S-PNDM (NFEs=15)
(FID: 5.22, recall: 0.511)

OGDM + S-PNDM (NFEs=15)
(FID: 2.96, recall: 0.585)

Figure 14. Qualitative results on CelebA dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=15.



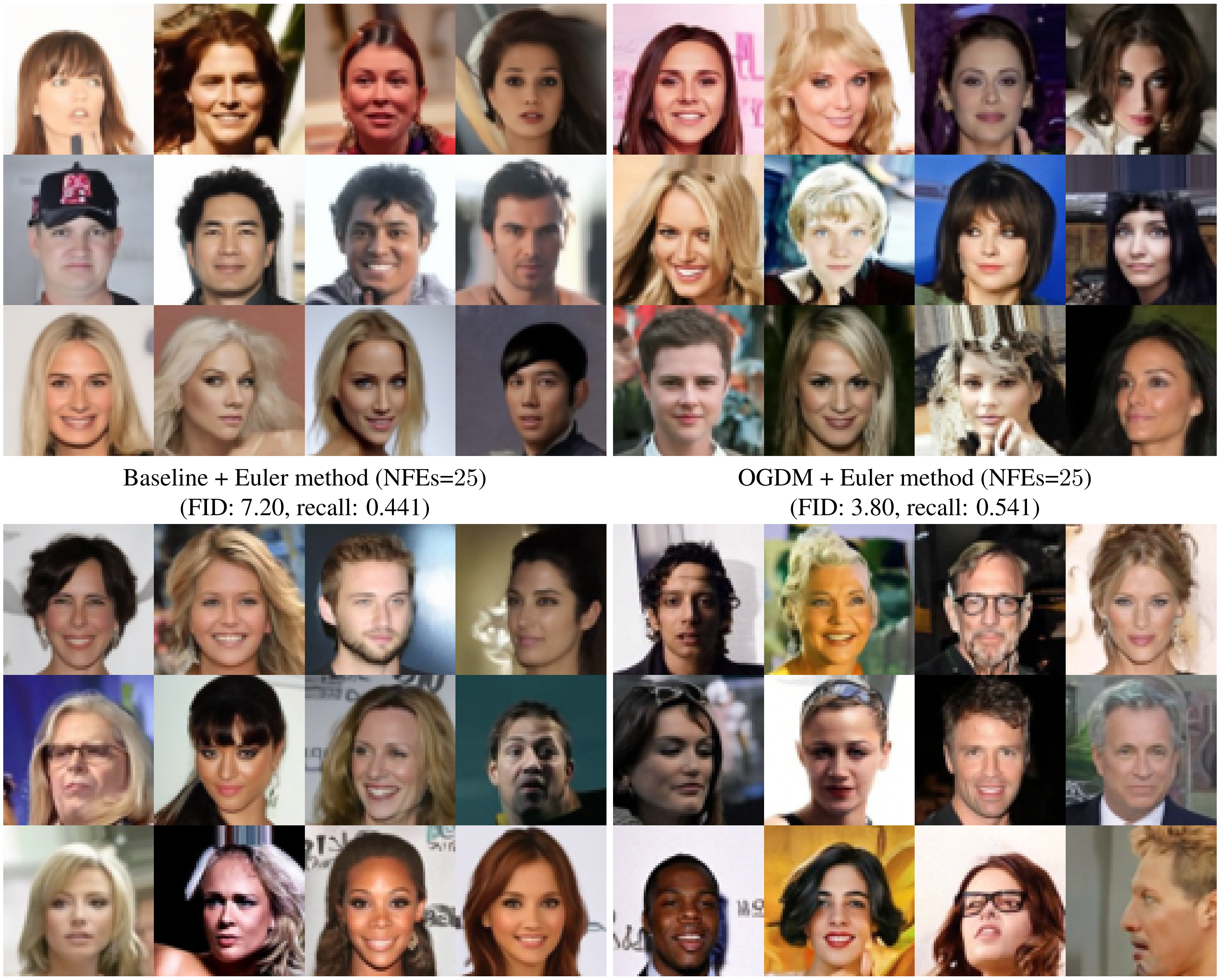
Baseline + Euler method (NFEs=20)
(FID: 7.88, recall: 0.429)

OGDM + Euler method (NFEs=20)
(FID: 3.94, recall: 0.534)

Baseline + S-PNDM (NFEs=20)
(FID: 4.15, recall: 0.540)

OGDM + S-PNDM (NFEs=20)
(FID: 2.70, recall: 0.604)

Figure 15. Qualitative results on CelebA dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=20.



Baseline + Euler method (NFEs=25)
(FID: 7.20, recall: 0.441)

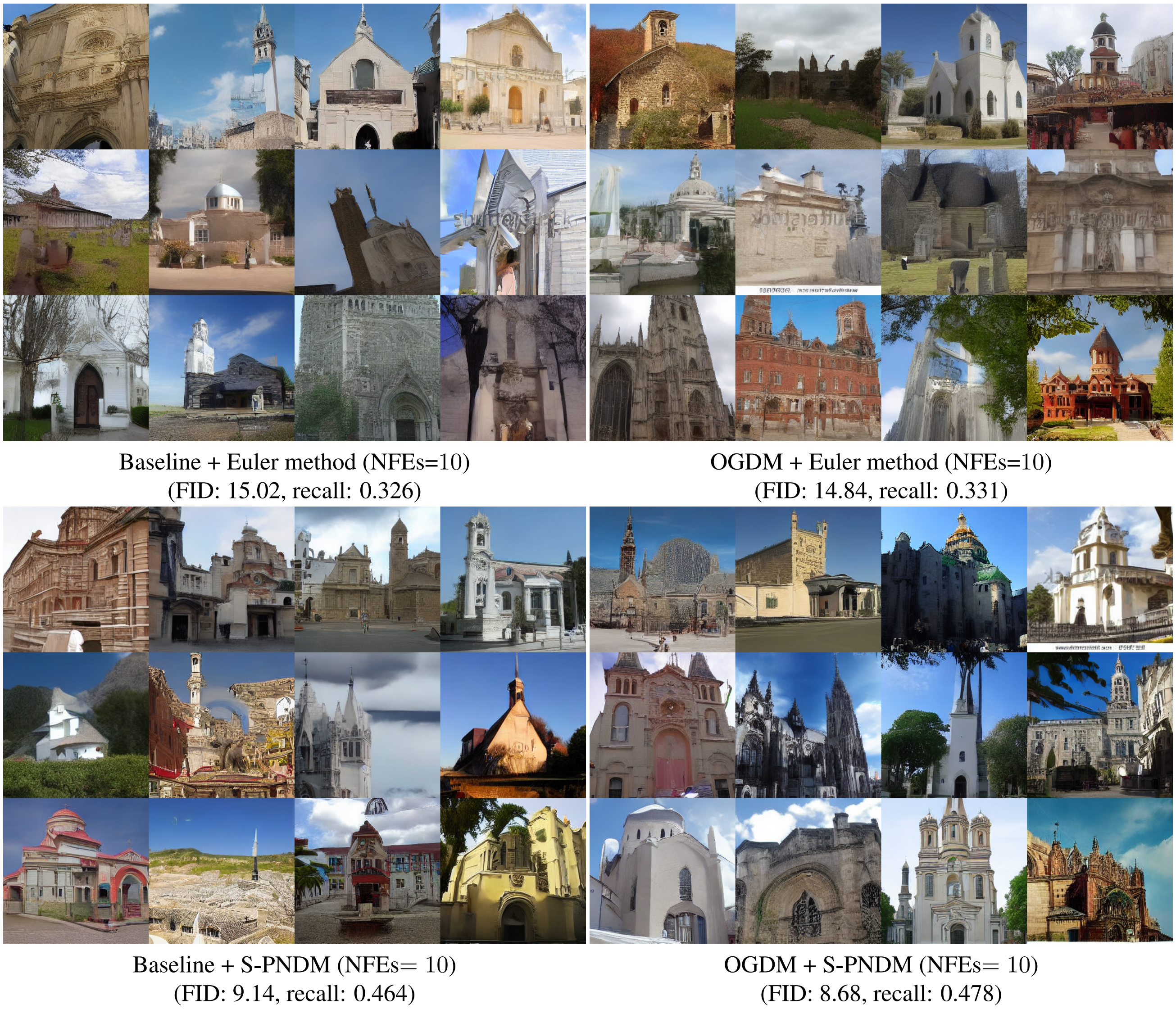
OGDM + Euler method (NFEs=25)
(FID: 3.80, recall: 0.541)

Baseline + S-PNDM (NFEs=25)
(FID: 3.67, recall: 0.553)

OGDM + S-PNDM (NFEs=25)
(FID: 2.62, recall: 0.607)

Figure 16. Qualitative results on CelebA dataset with the ADM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=25.

F.4. LSUN Church samples with LDM baseline



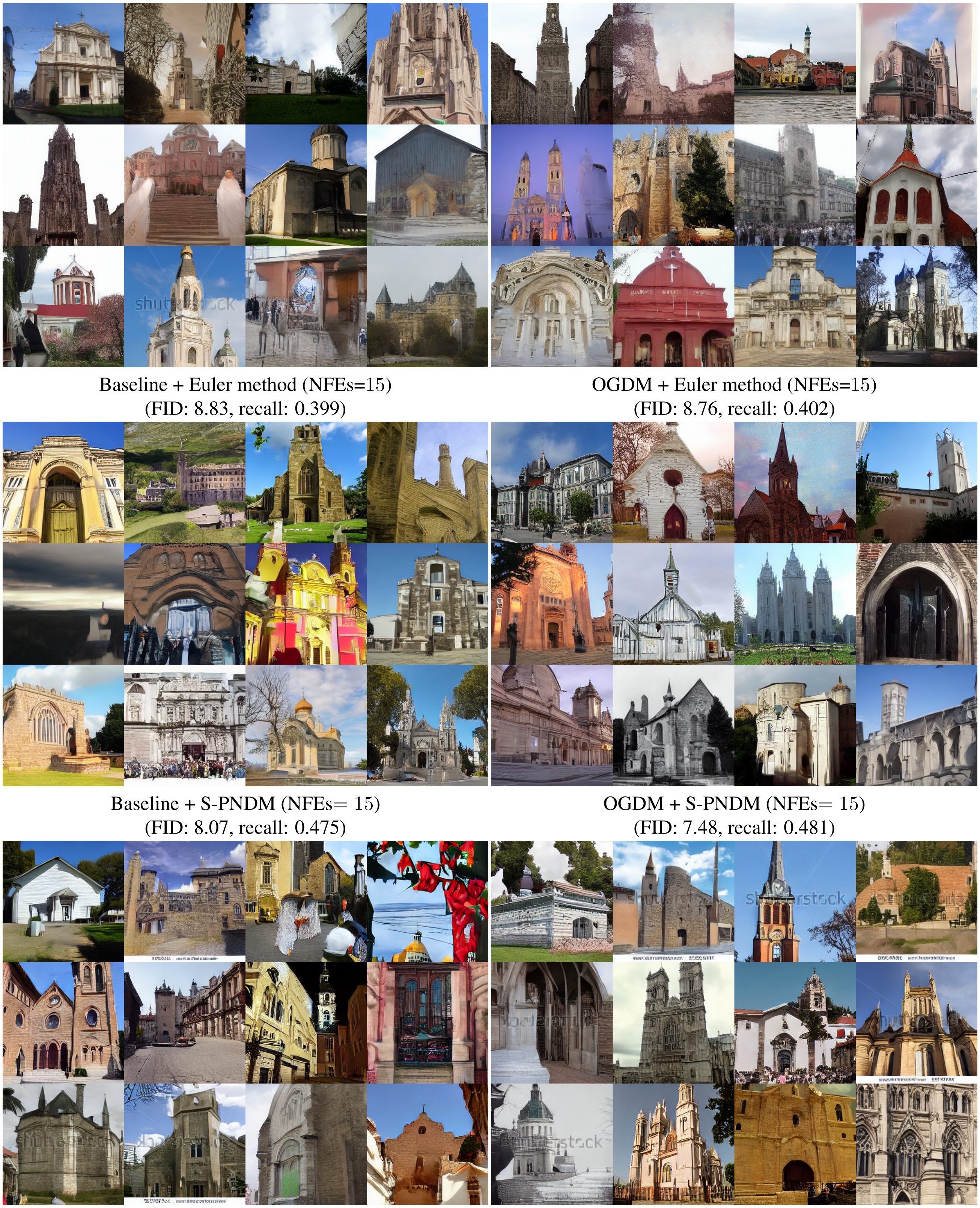
Baseline + Euler method (NFEs=10)
(FID: 15.02, recall: 0.326)

OGDM + Euler method (NFEs=10)
(FID: 14.84, recall: 0.331)

Baseline + S-PNDM (NFEs= 10)
(FID: 9.14, recall: 0.464)

OGDM + S-PNDM (NFEs= 10)
(FID: 8.68, recall: 0.478)

Figure 17. Qualitative results on LSUN Church dataset with the LDM backbone using Euler method (top) and S-PNDM (bottom) with NFEs=10.



Baseline + Euler method (NFEs=15)
(FID: 8.83, recall: 0.399)

OGDM + Euler method (NFEs=15)
(FID: 8.76, recall: 0.402)

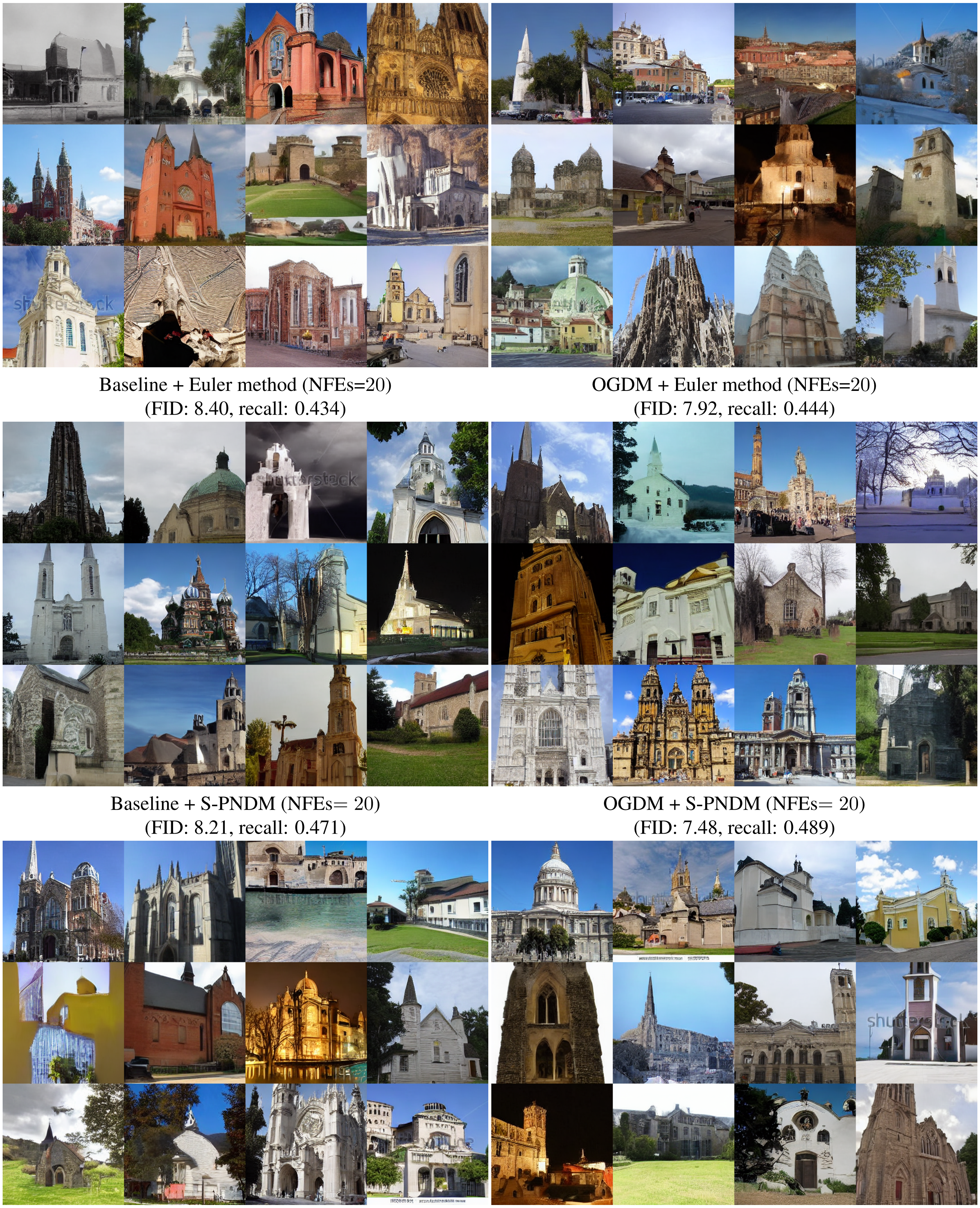
Baseline + S-PNDM (NFEs= 15)
(FID: 8.07, recall: 0.475)

OGDM + S-PNDM (NFEs= 15)
(FID: 7.48, recall: 0.481)

Baseline + F-PNDM (NFEs= 15)
(FID: 12.75, recall: 0.493)

OGDM + F-PNDM (NFEs= 15)
(FID: 11.78, recall: 0.505)

Figure 18. Qualitative results on LSUN Church dataset with the LDM backbone using Euler method (top), S-PNDM (middle) and F-PNDM (bottom) with NFEs=15.



Baseline + Euler method (NFEs=20)
(FID: 8.40, recall: 0.434)

OGDM + Euler method (NFEs=20)
(FID: 7.92, recall: 0.444)

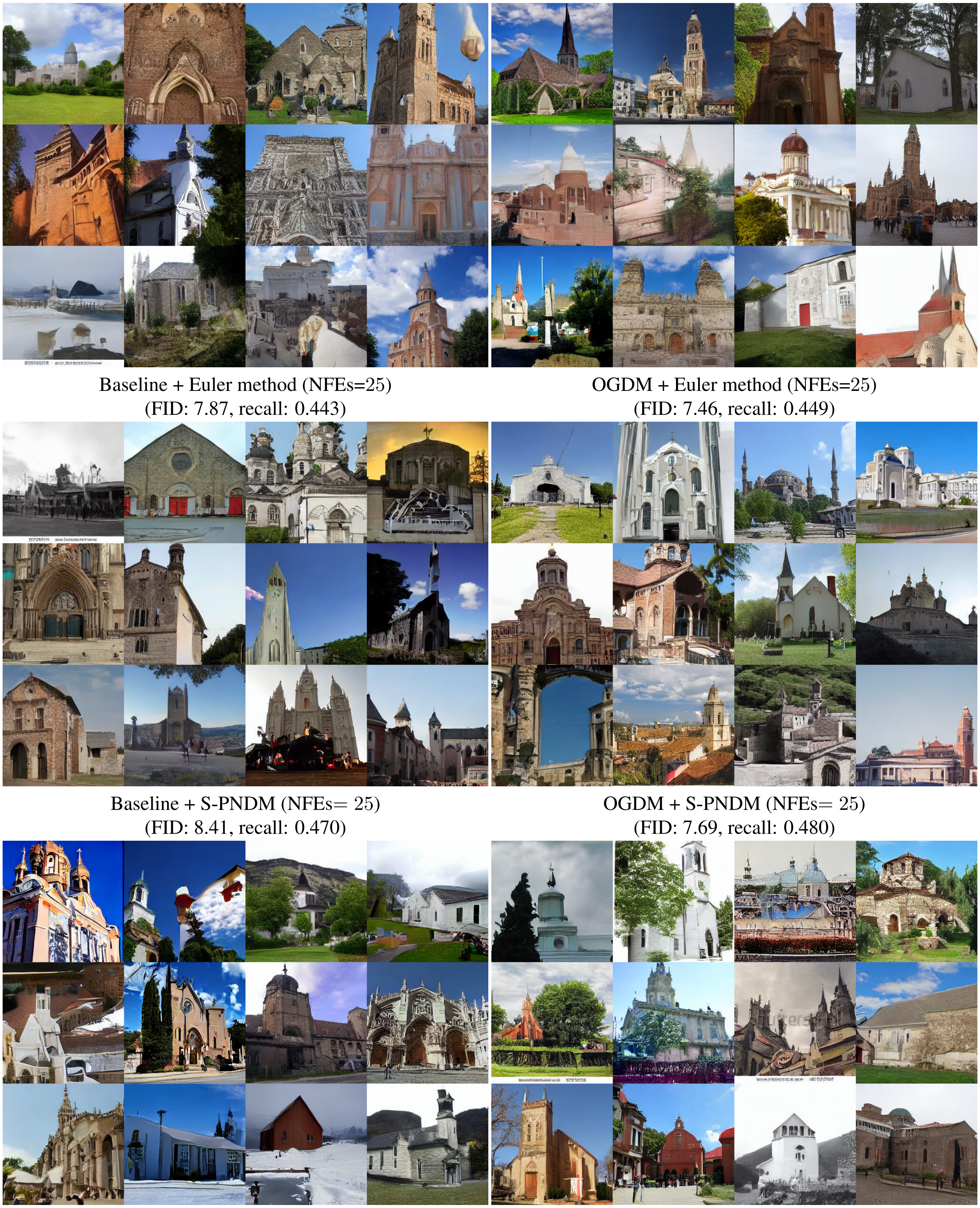
Baseline + S-PNDM (NFEs= 20)
(FID: 8.21, recall: 0.471)

OGDM + S-PNDM (NFEs= 20)
(FID: 7.48, recall: 0.489)

Baseline + F-PNDM (NFEs= 20)
(FID: 9.10, recall: 0.483)

OGDM + F-PNDM (NFEs= 20)
(FID: 8.39, recall: 0.495)

Figure 19. Qualitative results on LSUN Church dataset with the LDM backbone using Euler method (top), S-PNDM (middle) and F-PNDM (bottom) with NFEs=20.



Baseline + Euler method (NFEs=25)
(FID: 7.87, recall: 0.443)

OGDM + Euler method (NFEs=25)
(FID: 7.46, recall: 0.449)

Baseline + S-PNDM (NFEs= 25)
(FID: 8.41, recall: 0.470)

OGDM + S-PNDM (NFEs= 25)
(FID: 7.69, recall: 0.480)

Baseline + F-PNDM (NFEs= 25)
(FID: 9.04, recall: 0.474)

OGDM + F-PNDM (NFEs= 25)
(FID: 8.24, recall: 0.481)

Figure 20. Qualitative results on LSUN Church dataset with the LDM backbone using Euler method (top), S-PNDM (middle) and F-PNDM (bottom) with NFEs=25.

F.5. Nearest neighborhoods

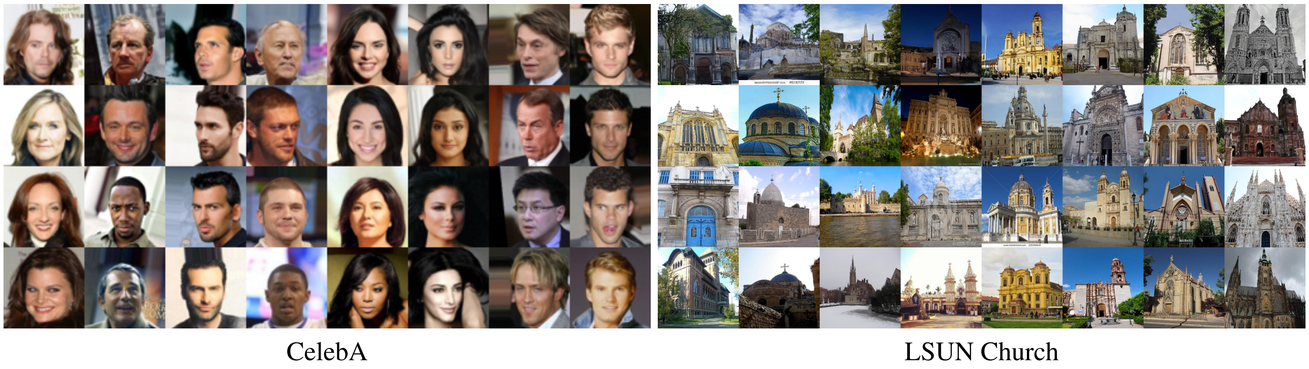


Figure 21. Nearest neighborhoods of generated samples from CelebA and LSUN Church datasets. The top row showcases our generated samples using Euler method with NFEs $= 50$, while the remaining three rows display the nearest neighborhoods from each training dataset. The distances are measured in the Inception-v3 [32] feature space.