Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at `https://github.com/hojonathanho/diffusion`.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

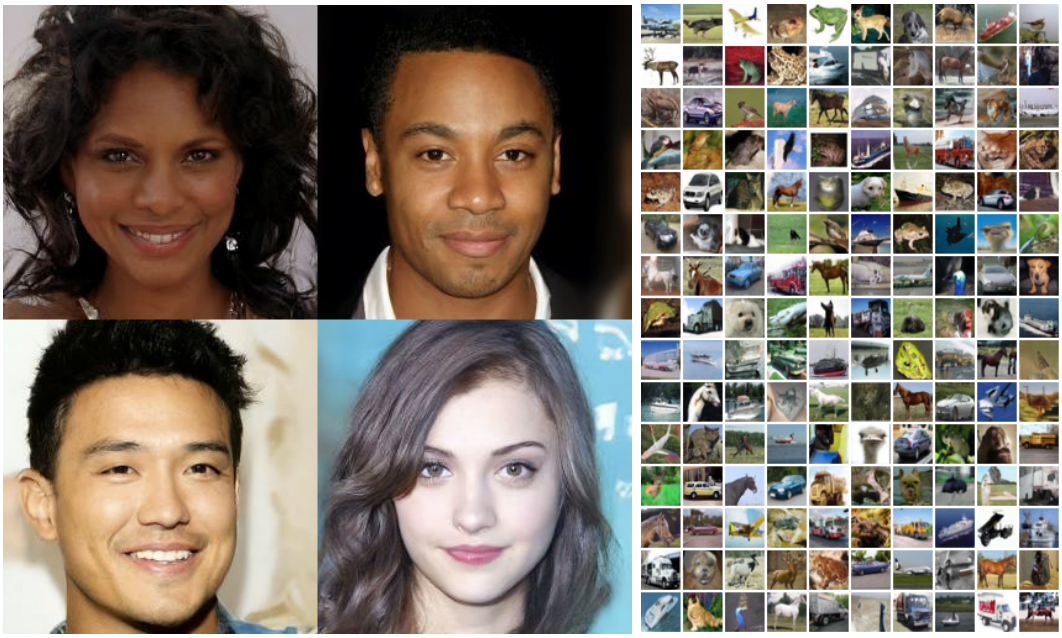


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

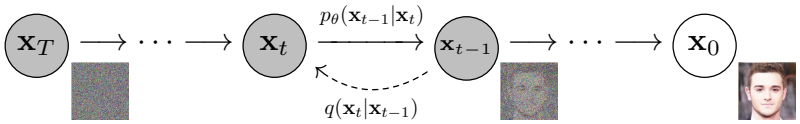


Figure 2: The directed graphical model considered in this work.

This paper presents progress in diffusion probabilistic models [53]. A diffusion probabilistic model (which we will call a "diffusion model" for brevity) is a parameterized Markov chain trained using variational inference to produce samples matching the data after finite time. Transitions of this chain are learned to reverse a diffusion process, which is a Markov chain that gradually adds noise to the data in the opposite direction of sampling until signal is destroyed. When the diffusion consists of small amounts of Gaussian noise, it is sufficient to set the sampling chain transitions to conditional Gaussians too, allowing for a particularly simple neural network parameterization.

Diffusion models are straightforward to define and efficient to train, but to the best of our knowledge, there has been no demonstration that they are capable of generating high quality samples. We show that diffusion models actually are capable of generating high quality samples, sometimes better than the published results on other types of generative models (Section 4). In addition, we show that a certain parameterization of diffusion models reveals an equivalence with denoising score matching over multiple noise levels during training and with annealed Langevin dynamics during sampling (Section 3.2) [55, 61]. We obtained our best sample quality results using this parameterization (Section 4.2), so we consider this equivalence to be one of our primary contributions.

Despite their sample quality, our models do not have competitive log likelihoods compared to other likelihood-based models (our models do, however, have log likelihoods better than the large estimates annealed importance sampling has been reported to produce for energy based models and score matching [11, 55]). We find that the majority of our models' lossless codelengths are consumed to describe imperceptible image details (Section 4.3). We present a more refined analysis of this phenomenon in the language of lossy compression, and we show that the sampling procedure of diffusion models is a type of progressive decoding that resembles autoregressive decoding along a bit ordering that vastly generalizes what is normally possible with autoregressive models.

2 Background

Diffusion models [53] are latent variable models of the form $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, \dots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is called the *reverse process*, and it is defined as a Markov chain with learned Gaussian transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{1}$$

What distinguishes diffusion models from other types of latent variable models is that the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \dots, \beta_T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L \tag{3}$$

The forward process variances β_t can be learned by reparameterization [33] or held constant as hyperparameters, and expressiveness of the reverse process is ensured in part by the choice of Gaussian conditionals in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, because both processes have the same functional form when β_t are small [53]. A notable property of the forward process is that it admits sampling $\mathbf{x}_t$ at an arbitrary timestep t in closed form: using the notation $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, we have

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{4}$$

Efficient training is therefore possible by optimizing random terms of L with stochastic gradient descent. Further improvements come from variance reduction by rewriting L (3) as:

$$\mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{L_T}+\sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Big] \tag{5}$$

(See Appendix A for details. The labels on the terms are used in Section 3.) Equation (5) uses KL divergence to directly compare $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ against forward process posteriors, which are tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_{t-1};\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0),\tilde{\beta}_t\mathbf{I}), \tag{6}$$

$$\text{where}\quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0):=\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0+\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad\text{and}\quad \tilde{\beta}_t:=\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{7}$$

Consequently, all KL divergences in Eq. (5) are comparisons between Gaussians, so they can be calculated in a Rao-Blackwellized fashion with closed form expressions instead of high variance Monte Carlo estimates.

3 Diffusion models and denoising autoencoders

Diffusion models might appear to be a restricted class of latent variable models, but they allow a large number of degrees of freedom in implementation. One must choose the variances β_t of the forward process and the model architecture and Gaussian distribution parameterization of the reverse process. To guide our choices, we establish a new explicit connection between diffusion models and denoising score matching (Section 3.2) that leads to a simplified, weighted variational bound objective for diffusion models (Section 3.4). Ultimately, our model design is justified by simplicity and empirical results (Section 4). Our discussion is categorized by the terms of Eq. (5).

3.1 Forward process and L_T

We ignore the fact that the forward process variances β_t are learnable by reparameterization and instead fix them to constants (see Section 4 for details). Thus, in our implementation, the approximate posterior q has no learnable parameters, so L_T is a constant during training and can be ignored.

3.2 Reverse process and $L_{1:T-1}$

Now we discuss our choices in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)=\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t))$ for $1<t\le T$. First, we set $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)=\sigma_t^2\mathbf{I}$ to untrained time dependent constants. Experimentally, both $\sigma_t^2=\beta_t$ and $\sigma_t^2=\tilde{\beta}_t=\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ had similar results. The first choice is optimal for $\mathbf{x}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I})$, and the second is optimal for $\mathbf{x}_0$ deterministically set to one point. These are the two extreme choices corresponding to upper and lower bounds on reverse process entropy for data with coordinatewise unit variance [53].

Second, to represent the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)$, we propose a specific parameterization motivated by the following analysis of L_t. With $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)=\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_\theta(\mathbf{x}_t,t),\sigma_t^2\mathbf{I})$, we can write:

$$L_{t-1}=\mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right]+C \tag{8}$$

where C is a constant that does not depend on θ. So, we see that the most straightforward parameterization of $\boldsymbol{\mu}_\theta$ is a model that predicts $\tilde{\boldsymbol{\mu}}_t$, the forward process posterior mean. However, we can expand Eq. (8) further by reparameterizing Eq. (4) as $\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon})=\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ and applying the forward process posterior formula (7):

$$L_{t-1}-C=\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon})-\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right)-\boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right] \tag{9}$$

$$=\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon})-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right)-\boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right] \tag{10}$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Equation (10) reveals that $\boldsymbol{\mu}_\theta$ must predict $\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right)$ given $\mathbf{x}_t$. Since $\mathbf{x}_t$ is available as input to the model, we may choose the parameterization

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t)) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \tag{11}$$

where $\boldsymbol{\epsilon}_\theta$ is a function approximator intended to predict $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$. To sample $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is to compute $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The complete sampling procedure, Algorithm 2, resembles Langevin dynamics with $\boldsymbol{\epsilon}_\theta$ as a learned gradient of the data density. Furthermore, with the parameterization (11), Eq. (10) simplifies to:

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right] \tag{12}$$

which resembles denoising score matching over multiple noise scales indexed by t [55]. As Eq. (12) is equal to (one term of) the variational bound for the Langevin-like reverse process (11), we see that optimizing an objective resembling denoising score matching is equivalent to using variational inference to fit the finite-time marginal of a sampling chain resembling Langevin dynamics.

To summarize, we can train the reverse process mean function approximator $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t$, or by modifying its parameterization, we can train it to predict $\boldsymbol{\epsilon}$. (There is also the possibility of predicting $\mathbf{x}_0$, but we found this to lead to worse sample quality early in our experiments.) We have shown that the $\boldsymbol{\epsilon}$-prediction parameterization both resembles Langevin dynamics and simplifies the diffusion model's variational bound to an objective that resembles denoising score matching. Nonetheless, it is just another parameterization of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, so we verify its effectiveness in Section 4 in an ablation where we compare predicting $\boldsymbol{\epsilon}$ against predicting $\tilde{\boldsymbol{\mu}}_t$.

3.3 Data scaling, reverse process decoder, and L_0

We assume that image data consists of integers in $\{0, 1, \dots, 255\}$ scaled linearly to $[-1, 1]$. This ensures that the neural network reverse process operates on consistently scaled inputs starting from the standard normal prior $p(\mathbf{x}_T)$. To obtain discrete log likelihoods, we set the last term of the reverse process to an independent discrete decoder derived from the Gaussian $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$:

$$\begin{aligned} p_\theta(\mathbf{x}_0|\mathbf{x}_1) &= \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2) \, dx \\ \delta_+(x) &= \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \qquad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases} \end{aligned} \tag{13}$$

where D is the data dimensionality and the i superscript indicates extraction of one coordinate. (It would be straightforward to instead incorporate a more powerful decoder like a conditional autoregressive model, but we leave that to future work.) Similar to the discretized continuous distributions used in VAE decoders and autoregressive models [34, 52], our choice here ensures that the variational bound is a lossless codelength of discrete data, without need of adding noise to the data or incorporating the Jacobian of the scaling operation into the log likelihood. At the end of sampling, we display $\boldsymbol{\mu}_\theta(\mathbf{x}_1, 1)$ noiselessly.

3.4 Simplified training objective

With the reverse process and decoder defined above, the variational bound, consisting of terms derived from Eqs. (12) and (13), is clearly differentiable with respect to θ and is ready to be employed for

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

Objective	IS	FID
$\tilde{\boldsymbol{\mu}}$ prediction (baseline)		
L, learned diagonal $\mathbf{\Sigma}$	7.28 ± 0.10	23.69
L, fixed isotropic $\mathbf{\Sigma}$	8.06 ± 0.09	13.22
$\|\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_\theta\|^2$	–	–
$\boldsymbol{\epsilon}$ prediction (ours)		
L, learned diagonal $\mathbf{\Sigma}$	–	–
L, fixed isotropic $\mathbf{\Sigma}$	7.67 ± 0.13	13.51
$\|\tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

training. However, we found it beneficial to sample quality (and simpler to implement) to train on the following variant of the variational bound:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right] \tag{14}$$

where t is uniform between 1 and T. The $t = 1$ case corresponds to L_0 with the integral in the discrete decoder definition (13) approximated by the Gaussian probability density function times the bin width, ignoring σ_1^2 and edge effects. The $t > 1$ cases correspond to an unweighted version of Eq. (12), analogous to the loss weighting used by the NCSN denoising score matching model [55]. (L_T does not appear because the forward process variances β_t are fixed.) Algorithm 1 displays the complete training procedure with this simplified objective.

Since our simplified objective (14) discards the weighting in Eq. (12), it is a weighted variational bound that emphasizes different aspects of reconstruction compared to the standard variational bound [18, 22]. In particular, our diffusion process setup in Section 4 causes the simplified objective to down-weight loss terms corresponding to small t. These terms train the network to denoise data with very small amounts of noise, so it is beneficial to down-weight them so that the network can focus on more difficult denoising tasks at larger t terms. We will see in our experiments that this reweighting leads to better sample quality.

4 Experiments

We set $T = 1000$ for all experiments so that the number of neural network evaluations needed during sampling matches previous work [53, 55]. We set the forward process variances to constants increasing linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. These constants were chosen to be small relative to data scaled to $[-1, 1]$, ensuring that reverse and forward processes have approximately the same functional form while keeping the signal-to-noise ratio at $\mathbf{x}_T$ as small as possible ($L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})) \approx 10^{-5}$ bits per dimension in our experiments).

To represent the reverse process, we use a U-Net backbone similar to an unmasked PixelCNN++ [52, 48] with group normalization throughout [66]. Parameters are shared across time, which is specified to the network using the Transformer sinusoidal position embedding [60]. We use self-attention at the 16×16 feature map resolution [63, 60]. Details are in Appendix B.

4.1 Sample quality

Table 1 shows Inception scores, FID scores, and negative log likelihoods (lossless codelengths) on CIFAR10. With our FID score of 3.17, our unconditional model achieves better sample quality than most models in the literature, including class conditional models. Our FID score is computed with respect to the training set, as is standard practice; when we compute it with respect to the test set, the score is 5.24, which is still better than many of the training set FID scores in the literature.

Figure 3: LSUN Church samples. FID=7.89

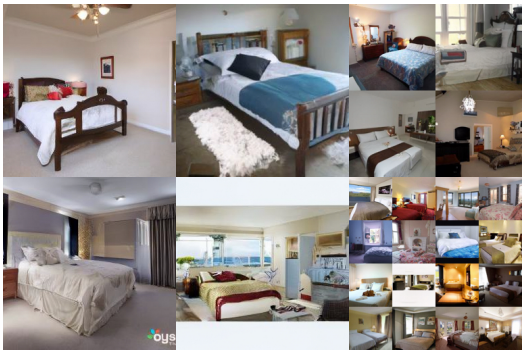


Figure 4: LSUN Bedroom samples. FID=4.90

Algorithm 3 Sending $\mathbf{x}_0$

1: Send $\mathbf{x}_T \sim q(\mathbf{x}_T|\mathbf{x}_0)$ using $p(\mathbf{x}_T)$
2: **for** $t = T-1, \dots, 2, 1$ **do**
3: Send $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0)$ using $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$
4: **end for**
5: Send $\mathbf{x}_0$ using $p_\theta(\mathbf{x}_0|\mathbf{x}_1)$

Algorithm 4 Receiving

1: Receive $\mathbf{x}_T$ using $p(\mathbf{x}_T)$
2: **for** $t = T-1, \dots, 1, 0$ **do**
3: Receive $\mathbf{x}_t$ using $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$
4: **end for**
5: **return** $\mathbf{x}_0$

We find that training our models on the true variational bound yields better codelengths than training on the simplified objective, as expected, but the latter yields the best sample quality. See Fig. 1 for CIFAR10 and CelebA-HQ 256×256 samples, Fig. 3 and Fig. 4 for LSUN 256×256 samples [71], and Appendix D for more.

4.2 Reverse process parameterization and training objective ablation

In Table 2, we show the sample quality effects of reverse process parameterizations and training objectives (Section 3.2). We find that the baseline option of predicting $\tilde{\boldsymbol{\mu}}$ works well only when trained on the true variational bound instead of unweighted mean squared error, a simplified objective akin to Eq. (14). We also see that learning reverse process variances (by incorporating a parameterized diagonal $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t)$ into the variational bound) leads to unstable training and poorer sample quality compared to fixed variances. Predicting $\boldsymbol{\epsilon}$, as we proposed, performs approximately as well as predicting $\tilde{\boldsymbol{\mu}}$ when trained on the variational bound with fixed variances, but much better when trained with our simplified objective.

4.3 Progressive coding

Table 1 also shows the codelengths of our CIFAR10 models. The gap between train and test is at most 0.03 bits per dimension, which is comparable to the gaps reported with other likelihood-based models and indicates that our diffusion model is not overfitting (see Appendix D for nearest neighbor visualizations). Still, while our lossless codelengths are better than the large estimates reported for energy based models and score matching using annealed importance sampling [11], they are not competitive with other types of likelihood-based generative models [7].

Since our samples are nonetheless of high quality, we conclude that diffusion models have an inductive bias that makes them excellent lossy compressors. Treating the variational bound terms $L_1 + \dots + L_T$ as rate and L_0 as distortion, our CIFAR10 model with the highest quality samples has a rate of **1.78** bits/dim and a distortion of **1.97** bits/dim, which amounts to a root mean squared error of 0.95 on a scale from 0 to 255. More than half of the lossless codelength describes imperceptible distortions.

Progressive lossy compression We can probe further into the rate-distortion behavior of our model by introducing a progressive lossy code that mirrors the form of Eq. (5): see Algorithms 3 and 4, which assume access to a procedure, such as minimal random coding [19, 20], that can transmit a sample $\mathbf{x} \sim q(\mathbf{x})$ using approximately $D_{\mathrm{KL}}(q(\mathbf{x}) \parallel p(\mathbf{x}))$ bits on average for any distributions p and q, for which only p is available to the receiver beforehand. When applied to $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, Algorithms 3 and 4 transmit $\mathbf{x}_T, \dots, \mathbf{x}_0$ in sequence using a total expected codelength equal to Eq. (5). The receiver,

at any time t, has the partial information $\mathbf{x}_t$ fully available and can progressively estimate:

$$\mathbf{x}_0 \approx \hat{\mathbf{x}}_0 = \left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)\right)/\sqrt{\bar{\alpha}_t} \tag{15}$$

due to Eq. (4). (A stochastic reconstruction $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ is also valid, but we do not consider it here because it makes distortion more difficult to evaluate.) Figure 5 shows the resulting rate-distortion plot on the CIFAR10 test set. At each time t, the distortion is calculated as the root mean squared error $\sqrt{\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2/D}$, and the rate is calculated as the cumulative number of bits received so far at time t. The distortion decreases steeply in the low-rate region of the rate-distortion plot, indicating that the majority of the bits are indeed allocated to imperceptible distortions.

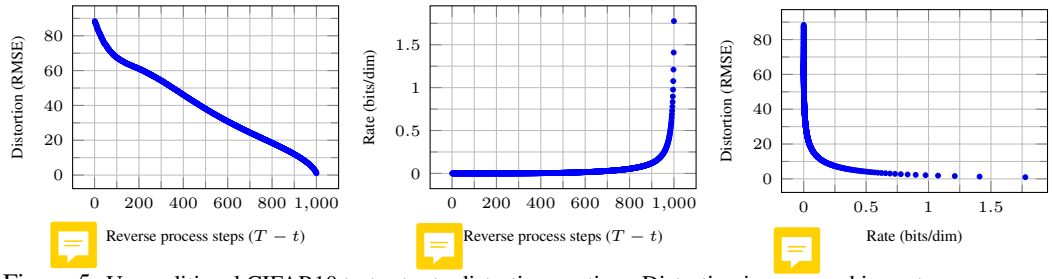


Figure 5: Unconditional CIFAR10 test set rate-distortion vs. time. Distortion is measured in root mean squared error on a $[0, 255]$ scale. See Table 4 for details.

Progressive generation We also run a progressive unconditional generation process given by progressive decompression from random bits. In other words, we predict the result of the reverse process, $\hat{\mathbf{x}}_0$, while sampling from the reverse process using Algorithm 2. Figures 6 and 10 show the resulting sample quality of $\hat{\mathbf{x}}_0$ over the course of the reverse process. Large scale image features appear first and details appear last. Figure 7 shows stochastic predictions $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ with $\mathbf{x}_t$ frozen for various t. When t is small, all but fine details are preserved, and when t is large, only large scale features are preserved. Perhaps these are hints of conceptual compression [18].

Figure 6: Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right). Extended samples and sample quality metrics over time in the appendix (Figs. 10 and 14).

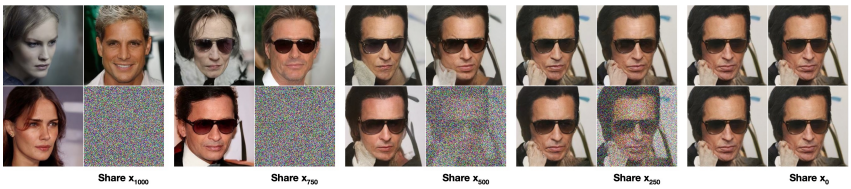


Figure 7: When conditioned on the same latent, CelebA-HQ 256×256 samples share high-level attributes. Bottom-right quadrants are $\mathbf{x}_t$, and other quadrants are samples from $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Connection to autoregressive decoding Note that the variational bound (5) can be rewritten as:

$$L = D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) + \mathbb{E}_q\left[\sum_{t\geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right] + H(\mathbf{x}_0) \tag{16}$$

(See Appendix A for a derivation.) Now consider setting the diffusion process length T to the dimensionality of the data, defining the forward process so that $q(\mathbf{x}_t|\mathbf{x}_0)$ places all probability mass on $\mathbf{x}_0$ with the first t coordinates masked out (i.e. $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ masks out the t^{th} coordinate), setting $p(\mathbf{x}_T)$ to place all mass on a blank image, and, for the sake of argument, taking $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to

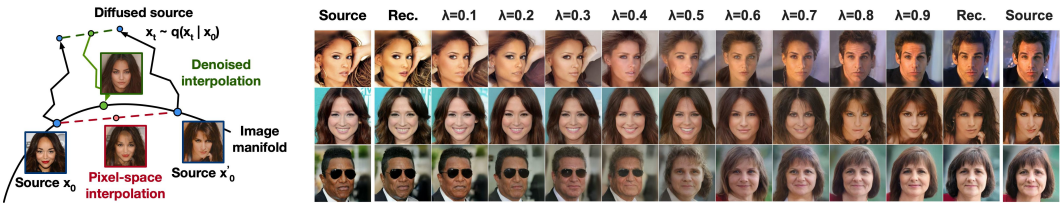


Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.

be a fully expressive conditional distribution. With these choices, $D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) = 0$, and minimizing $D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$ trains p_θ to copy coordinates $t+1, \dots, T$ unchanged and to predict the t^{th} coordinate given $t+1, \dots, T$. Thus, training p_θ with this particular diffusion is training an autoregressive model.

We can therefore interpret the Gaussian diffusion model (2) as a kind of autoregressive model with a generalized bit ordering that cannot be expressed by reordering data coordinates. Prior work has shown that such reorderings introduce inductive biases that have an impact on sample quality [38], so we speculate that the Gaussian diffusion serves a similar purpose, perhaps to greater effect since Gaussian noise might be more natural to add to images compared to masking noise. Moreover, the Gaussian diffusion length is not restricted to equal the data dimension; for instance, we use $T = 1000$, which is less than the dimension of the $32 \times 32 \times 3$ or $256 \times 256 \times 3$ images in our experiments. Gaussian diffusions can be made shorter for fast sampling or longer for model expressiveness.

4.4 Interpolation

We can interpolate source images $\mathbf{x}_0, \mathbf{x}'_0 \sim q(\mathbf{x}_0)$ in latent space using q as a stochastic encoder, $\mathbf{x}_t, \mathbf{x}'_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$, then decoding the linearly interpolated latent $\bar{\mathbf{x}}_t = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}'_0$ into image space by the reverse process, $\bar{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\bar{\mathbf{x}}_t)$. In effect, we use the reverse process to remove artifacts from linearly interpolating corrupted versions of the source images, as depicted in Fig. 8 (left). We fixed the noise for different values of λ so $\mathbf{x}_t$ and $\mathbf{x}'_t$ remain the same. Fig. 8 (right) shows interpolations and reconstructions of original CelebA-HQ 256×256 images ($t = 500$). The reverse process produces high-quality reconstructions, and plausible interpolations that smoothly vary attributes such as pose, skin tone, hairstyle, expression and background, but not eyewear. Larger t results in coarser and more varied interpolations, with novel samples at $t = 1000$ (Appendix Fig. 9).

5 Related Work

While diffusion models might resemble flows [9, 46, 10, 32, 5, 16, 23] and VAEs [33, 47, 37], diffusion models are designed so that q has no parameters and the top-level latent $\mathbf{x}_T$ has nearly zero mutual information with the data $\mathbf{x}_0$. Our ϵ-prediction reverse process parameterization establishes a connection between diffusion models and denoising score matching over multiple noise levels with annealed Langevin dynamics for sampling [55, 56]. Diffusion models, however, admit straightforward log likelihood evaluation, and the training procedure explicitly trains the Langevin dynamics sampler using variational inference (see Appendix C for details). The connection also has the reverse implication that a certain weighted form of denoising score matching is the same as variational inference to train a Langevin-like sampler. Other methods for learning transition operators of Markov chains include infusion training [2], variational walkback [15], generative stochastic networks [1], and others [50, 54, 36, 42, 35, 65].

By the known connection between score matching and energy-based modeling, our work could have implications for other recent work on energy-based models [67–69, 12, 70, 13, 11, 41, 17, 8]. Our rate-distortion curves are computed over time in one evaluation of the variational bound, reminiscent of how rate-distortion curves can be computed over distortion penalties in one run of annealed importance sampling [24]. Our progressive decoding argument can be seen in convolutional DRAW and related models [18, 40] and may also lead to more general designs for subscale orderings or sampling strategies for autoregressive models [38, 64].

6 Conclusion

We have presented high quality image samples using diffusion models, and we have found connections among diffusion models and variational inference for training Markov chains, denoising score matching and annealed Langevin dynamics (and energy-based models by extension), autoregressive models, and progressive lossy compression. Since diffusion models seem to have excellent inductive biases for image data, we look forward to investigating their utility in other data modalities and as components in other types of generative models and machine learning systems.

Broader Impact

Our work on diffusion models takes on a similar scope as existing work on other types of deep generative models, such as efforts to improve the sample quality of GANs, flows, autoregressive models, and so forth. Our paper represents progress in making diffusion models a generally useful tool in this family of techniques, so it may serve to amplify any impacts that generative models have had (and will have) on the broader world.

Unfortunately, there are numerous well-known malicious uses of generative models. Sample generation techniques can be employed to produce fake images and videos of high profile figures for political purposes. While fake images were manually created long before software tools were available, generative models such as ours make the process easier. Fortunately, CNN-generated images currently have subtle flaws that allow detection [62], but improvements in generative models may make this more difficult. Generative models also reflect the biases in the datasets on which they are trained. As many large datasets are collected from the internet by automated systems, it can be difficult to remove these biases, especially when the images are unlabeled. If samples from generative models trained on these datasets proliferate throughout the internet, then these biases will only be reinforced further.

On the other hand, diffusion models may be useful for data compression, which, as data becomes higher resolution and as global internet traffic increases, might be crucial to ensure accessibility of the internet to wide audiences. Our work might contribute to representation learning on unlabeled raw data for a large range of downstream tasks, from image classification to reinforcement learning, and diffusion models might also become viable for creative uses in art, photography, and music.

Acknowledgments and Disclosure of Funding

This work was supported by ONR PECASE and the NSF Graduate Research Fellowship under grant number DGE-1752814. Google's TensorFlow Research Cloud (TFRC) provided Cloud TPUs.

References

[1] Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, Eric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. GSNs: generative stochastic networks. *Information and Inference: A Journal of the IMA*, 5(2):210–249, 2016.

[2] Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. In *International Conference on Learning Representations*, 2017.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[4] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv preprint arXiv:2003.06060*, 2020.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[6] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. PixelSNAIL: An improved autoregressive generative model. In *International Conference on Machine Learning*, pages 863–871, 2018.

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*, 2020.

[9] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[11] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, pages 3603–3613, 2019.

[12] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative ConvNets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.

[13] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7518–7528, 2020.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[15] Anirudh Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational walkback: Learning a transition operator as a stochastic recurrent net. In *Advances in Neural Information Processing Systems*, pages 4392–4402, 2017.

[16] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

[17] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

[18] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pages 3549–3557, 2016.

[19] Prahladh Harsha, Rahul Jain, David McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 10–23. IEEE, 2007.

[20] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal random code learning: Getting bits back from compressed model parameters. In *International Conference on Learning Representations*, 2019.

[21] 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 *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[23] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, 2019.

[24] Sicong Huang, Alireza Makhzani, Yanshuai Cao, and Roger Grosse. Evaluating lossy compression rates of deep generative models. In *International Conference on Machine Learning*, 2020.

[25] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779, 2017.

[26] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419, 2018.

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages

4401–4410, 2019.

[29] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676v1*, 2020.

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[32] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[33] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[34] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[35] John Lawson, George Tucker, Bo Dai, and Rajesh Ranganath. Energy-inspired models: Learning with sampler-induced distributions. In *Advances in Neural Information Processing Systems*, pages 8501–8513, 2019.

[36] Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations*, 2018.

[37] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems*, pages 6548–6558, 2019.

[38] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*, 2019.

[39] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[40] Alex Nichol. VQ-DRAW: A sequential discrete VAE. *arXiv preprint arXiv:2003.01599*, 2020.

[41] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. *arXiv preprint arXiv:1903.12370*, 2019.

[42] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. In *Advances in Neural Information Processing Systems*, pages 5233–5243, 2019.

[43] Georg Ostrovski, Will Dabney, and Remi Munos. Autoregressive quantile networks for generative modeling. In *International Conference on Machine Learning*, pages 3936–3945, 2018.

[44] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[45] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, pages 14837–14847, 2019.

[46] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[47] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[49] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[50] Tim Salimans, Diederik Kingma, and Max Welling. Markov Chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

[51] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[52] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

[53] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015.

[54] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-NICE-MC: Adversarial training for MCMC. In *Advances in Neural Information Processing Systems*, pages 5140–5150, 2017.

[55] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019.

[56] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

[57] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[58] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, 2016.

[59] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[61] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[62] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[63] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[64] Auke J Wiggers and Emiel Hoogeboom. Predictive sampling with forecasting autoregressive models. *arXiv preprint arXiv:2002.09928*, 2020.

[65] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *arXiv preprint arXiv:2002.06707*, 2020.

[66] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[67] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[68] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7093–7101, 2017.

[69] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8629–8638, 2018.

[70] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based spatial-temporal generative convnets for dynamic patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[71] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, 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.

[72] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Extra information

LSUN FID scores for LSUN datasets are included in Table 3. Scores marked with * are reported by StyleGAN2 as baselines, and other scores are reported by their respective authors.

Table 3: FID scores for LSUN 256×256 datasets

Model	LSUN Bedroom	LSUN Church	LSUN Cat
ProgressiveGAN [27]	8.34	6.42	37.52
StyleGAN [28]	**2.65**	4.21*	8.53*
StyleGAN2 [30]	-	**3.86**	**6.93**
Ours (L_{simple})	6.36	7.89	19.75
Ours (L_{simple}, large)	4.90	-	-

Progressive compression Our lossy compression argument in Section 4.3 is only a proof of concept, because Algorithms 3 and 4 depend on a procedure such as minimal random coding [20], which is not tractable for high dimensional data. These algorithms serve as a compression interpretation of the variational bound (5) of Sohl-Dickstein et al. [53], not yet as a practical compression system.

Table 4: Unconditional CIFAR10 test set rate-distortion values (accompanies Fig. 5)

Reverse process time ($T - t + 1$)	Rate (bits/dim)	Distortion (RMSE $[0, 255]$)
1000	1.77581	0.95136
900	0.11994	12.02277
800	0.05415	18.47482
700	0.02866	24.43656
600	0.01507	30.80948
500	0.00716	38.03236
400	0.00282	46.12765
300	0.00081	54.18826
200	0.00013	60.97170
100	0.00000	67.60125

A Extended derivations

Below is a derivation of Eq. (5), the reduced variance variational bound for diffusion models. This material is from Sohl-Dickstein et al. [53]; we include it here only for completeness.

$$L = \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \tag{17}$$

$$= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \tag{18}$$

$$= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right] \tag{19}$$

$$= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right] \tag{20}$$

$$= \mathbb{E}_q \left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \tag{21}$$

$$= \mathbb{E}_q \left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \tag{22}$$

The following is an alternate version of L. It is not tractable to estimate, but it is useful for our discussion in Section 4.3.

$$L = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \tag{23}$$

$$= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} \right] \tag{24}$$

$$= \mathbb{E}_q \left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T)} - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log q(\mathbf{x}_0) \right] \tag{25}$$

$$= D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) + \mathbb{E}_q \left[\sum_{t \geq 1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right] + H(\mathbf{x}_0) \tag{26}$$

B Experimental details

Our neural network architecture follows the backbone of PixelCNN++ [52], which is a U-Net [48] based on a Wide ResNet [72]. We replaced weight normalization [49] with group normalization [66] to make the implementation simpler. Our 32×32 models use four feature map resolutions (32×32 to 4×4), and our 256×256 models use six. All models have two convolutional residual blocks per resolution level and self-attention blocks at the 16×16 resolution between the convolutional blocks [6]. Diffusion time t is specified by adding the Transformer sinusoidal position embedding [60] into each residual block. Our CIFAR10 model has 35.7 million parameters, and our LSUN and CelebA-HQ models have 114 million parameters. We also trained a larger variant of the LSUN Bedroom model with approximately 256 million parameters by increasing filter count.

We used TPU v3-8 (similar to 8 V100 GPUs) for all experiments. Our CIFAR model trains at 21 steps per second at batch size 128 (10.6 hours to train to completion at 800k steps), and sampling a batch of 256 images takes 17 seconds. Our CelebA-HQ/LSUN (256^2) models train at 2.2 steps per second at batch size 64, and sampling a batch of 128 images takes 300 seconds. We trained on CelebA-HQ for 0.5M steps, LSUN Bedroom for 2.4M steps, LSUN Cat for 1.8M steps, and LSUN Church for 1.2M steps. The larger LSUN Bedroom model was trained for 1.15M steps.

Apart from an initial choice of hyperparameters early on to make network size fit within memory constraints, we performed the majority of our hyperparameter search to optimize for CIFAR10 sample quality, then transferred the resulting settings over to the other datasets:

- We chose the β_t schedule from a set of constant, linear, and quadratic schedules, all constrained so that $L_T \approx 0$. We set $T = 1000$ without a sweep, and we chose a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.
- We set the dropout rate on CIFAR10 to 0.1 by sweeping over the values $\{0.1, 0.2, 0.3, 0.4\}$. Without dropout on CIFAR10, we obtained poorer samples reminiscent of the overfitting artifacts in an unregularized PixelCNN++ [52]. We set dropout rate on the other datasets to zero without sweeping.
- We used random horizontal flips during training for CIFAR10; we tried training both with and without flips, and found flips to improve sample quality slightly. We also used random horizontal flips for all other datasets except LSUN Bedroom.
- We tried Adam [31] and RMSProp early on in our experimentation process and chose the former. We left the hyperparameters to their standard values. We set the learning rate to 2×10^{-4} without any sweeping, and we lowered it to 2×10^{-5} for the 256×256 images, which seemed unstable to train with the larger learning rate.

- We set the batch size to 128 for CIFAR10 and 64 for larger images. We did not sweep over these values.
- We used EMA on model parameters with a decay factor of 0.9999. We did not sweep over this value.

Final experiments were trained once and evaluated throughout training for sample quality. Sample quality scores and log likelihood are reported on the minimum FID value over the course of training. On CIFAR10, we calculated Inception and FID scores on 50000 samples using the original code from the OpenAI [51] and TTUR [21] repositories, respectively. On LSUN, we calculated FID scores on 50000 samples using code from the StyleGAN2 [30] repository. CIFAR10 and CelebA-HQ were loaded as provided by TensorFlow Datasets (`https://www.tensorflow.org/datasets`), and LSUN was prepared using code from StyleGAN. Dataset splits (or lack thereof) are standard from the papers that introduced their usage in a generative modeling context. All details can be found in the source code release.

C Discussion on related work

Our model architecture, forward process definition, and prior differ from NCSN [55, 56] in subtle but important ways that improve sample quality, and, notably, we directly train our sampler as a latent variable model rather than adding it after training post-hoc. In greater detail:

1. We use a U-Net with self-attention; NCSN uses a RefineNet with dilated convolutions. We condition all layers on t by adding in the Transformer sinusoidal position embedding, rather than only in normalization layers (NCSNv1) or only at the output (v2).
2. Diffusion models scale down the data with each forward process step (by a $\sqrt{1-\beta_t}$ factor) so that variance does not grow when adding noise, thus providing consistently scaled inputs to the neural net reverse process. NCSN omits this scaling factor.
3. Unlike NCSN, our forward process destroys signal ($D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})) \approx 0$), ensuring a close match between the prior and aggregate posterior of $\mathbf{x}_T$. Also unlike NCSN, our β_t are very small, which ensures that the forward process is reversible by a Markov chain with conditional Gaussians. Both of these factors prevent distribution shift when sampling.
4. Our Langevin-like sampler has coefficients (learning rate, noise scale, etc.) derived rigorously from β_t in the forward process. Thus, our training procedure directly trains our sampler to match the data distribution after T steps: it trains the sampler as a latent variable model using variational inference. In contrast, NCSN's sampler coefficients are set by hand post-hoc, and their training procedure is not guaranteed to directly optimize a quality metric of their sampler.

D Samples

Additional samples Figure 11, 13, 16, 17, 18, and 19 show uncurated samples from the diffusion models trained on CelebA-HQ, CIFAR10 and LSUN datasets.

Latent structure and reverse process stochasticity During sampling, both the prior $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and Langevin dynamics are stochastic. To understand the significance of the second source of noise, we sampled multiple images conditioned on the same intermediate latent for the CelebA 256×256 dataset. Figure 7 shows multiple draws from the reverse process $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ that share the latent $\mathbf{x}_t$ for $t \in \{1000, 750, 500, 250\}$. To accomplish this, we run a single reverse chain from an initial draw from the prior. At the intermediate timesteps, the chain is split to sample multiple images. When the chain is split after the prior draw at $\mathbf{x}_{T=1000}$, the samples differ significantly. However, when the chain is split after more steps, samples share high-level attributes like gender, hair color, eyewear, saturation, pose and facial expression. This indicates that intermediate latents like $\mathbf{x}_{750}$ encode these attributes, despite their imperceptibility.

Coarse-to-fine interpolation Figure 9 shows interpolations between a pair of source CelebA 256×256 images as we vary the number of diffusion steps prior to latent space interpolation. Increasing the number of diffusion steps destroys more structure in the source images, which the

model completes during the reverse process. This allows us to interpolate at both fine granularities and coarse granularities. In the limiting case of 0 diffusion steps, the interpolation mixes source images in pixel space. On the other hand, after 1000 diffusion steps, source information is lost and interpolations are novel samples.

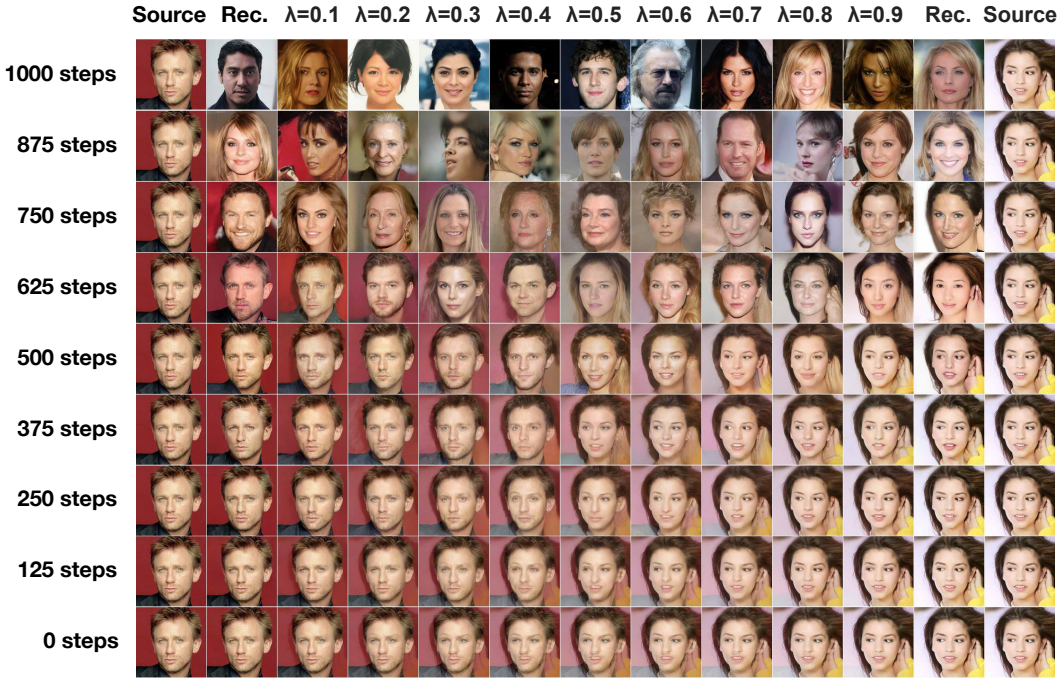


Figure 9: Coarse-to-fine interpolations that vary the number of diffusion steps prior to latent mixing.

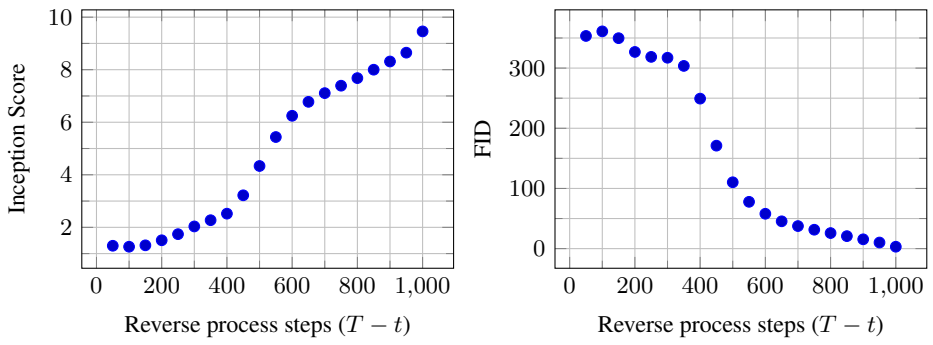
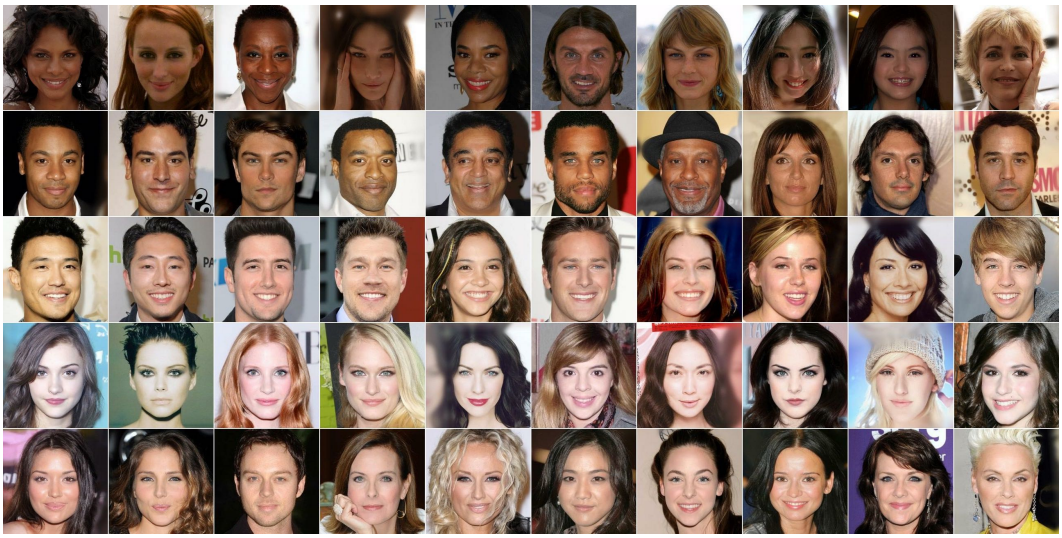


Figure 10: Unconditional CIFAR10 progressive sampling quality over time

Figure 11: CelebA-HQ 256×256 generated samples



(a) Pixel space nearest neighbors

(b) Inception feature space nearest neighbors

Figure 12: CelebA-HQ 256×256 nearest neighbors, computed on a 100×100 crop surrounding the faces. Generated samples are in the leftmost column, and training set nearest neighbors are in the remaining columns.

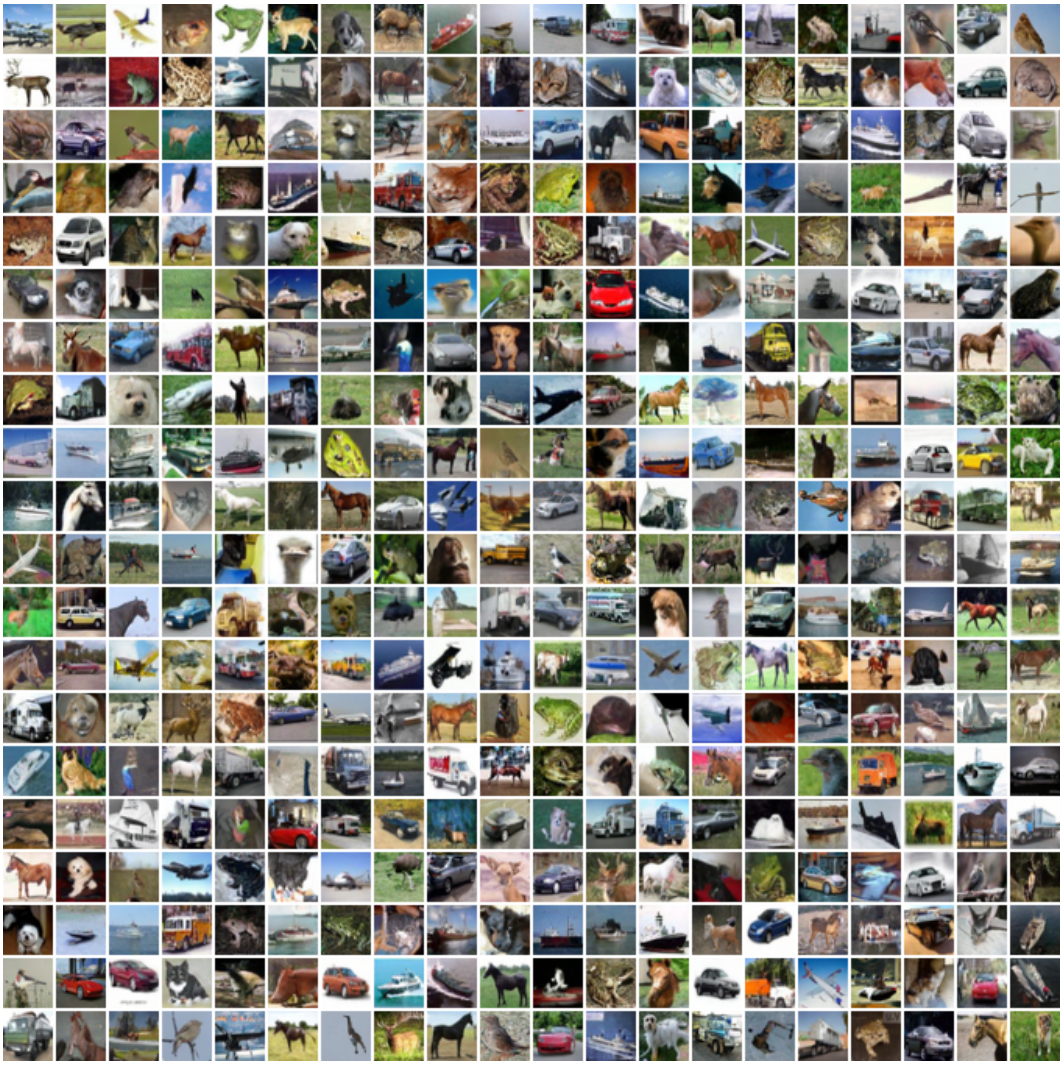


Figure 13: Unconditional CIFAR10 generated samples

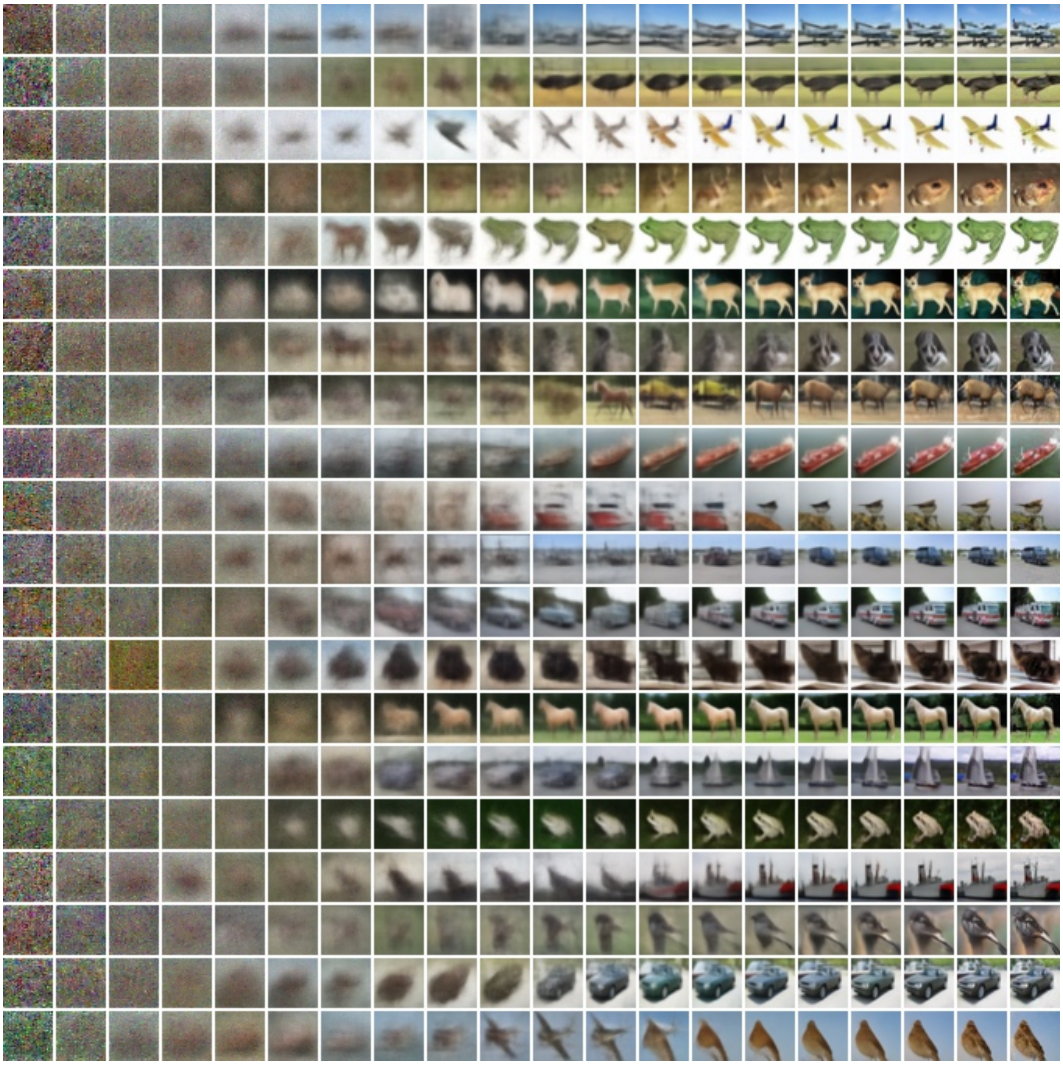
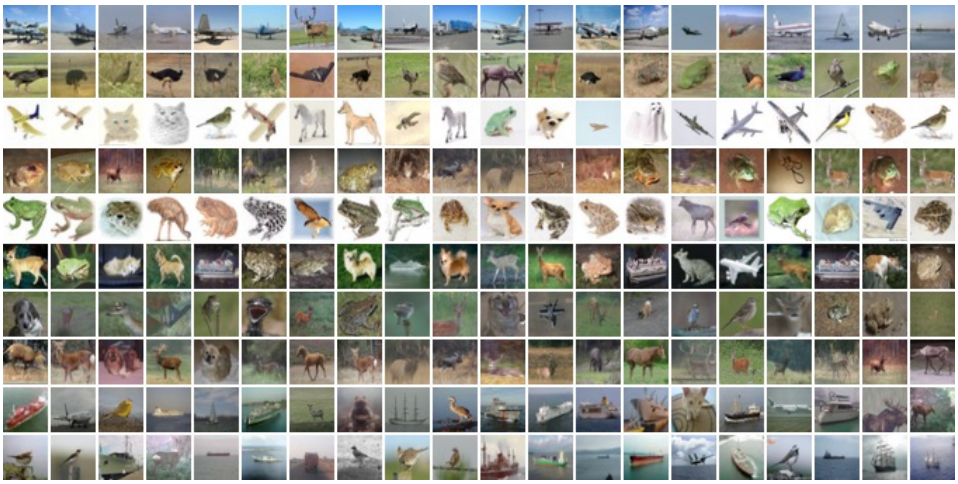
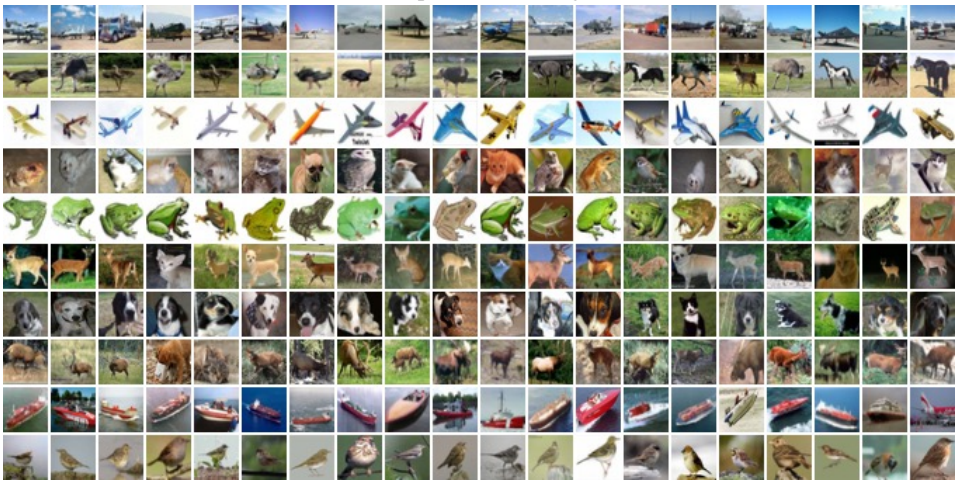


Figure 14: Unconditional CIFAR10 progressive generation



(a) Pixel space nearest neighbors



(b) Inception feature space nearest neighbors

Figure 15: Unconditional CIFAR10 nearest neighbors. Generated samples are in the leftmost column, and training set nearest neighbors are in the remaining columns.

Figure 16: LSUN Church generated samples. FID=7.89

Figure 17: LSUN Bedroom generated samples, large model. FID=4.90

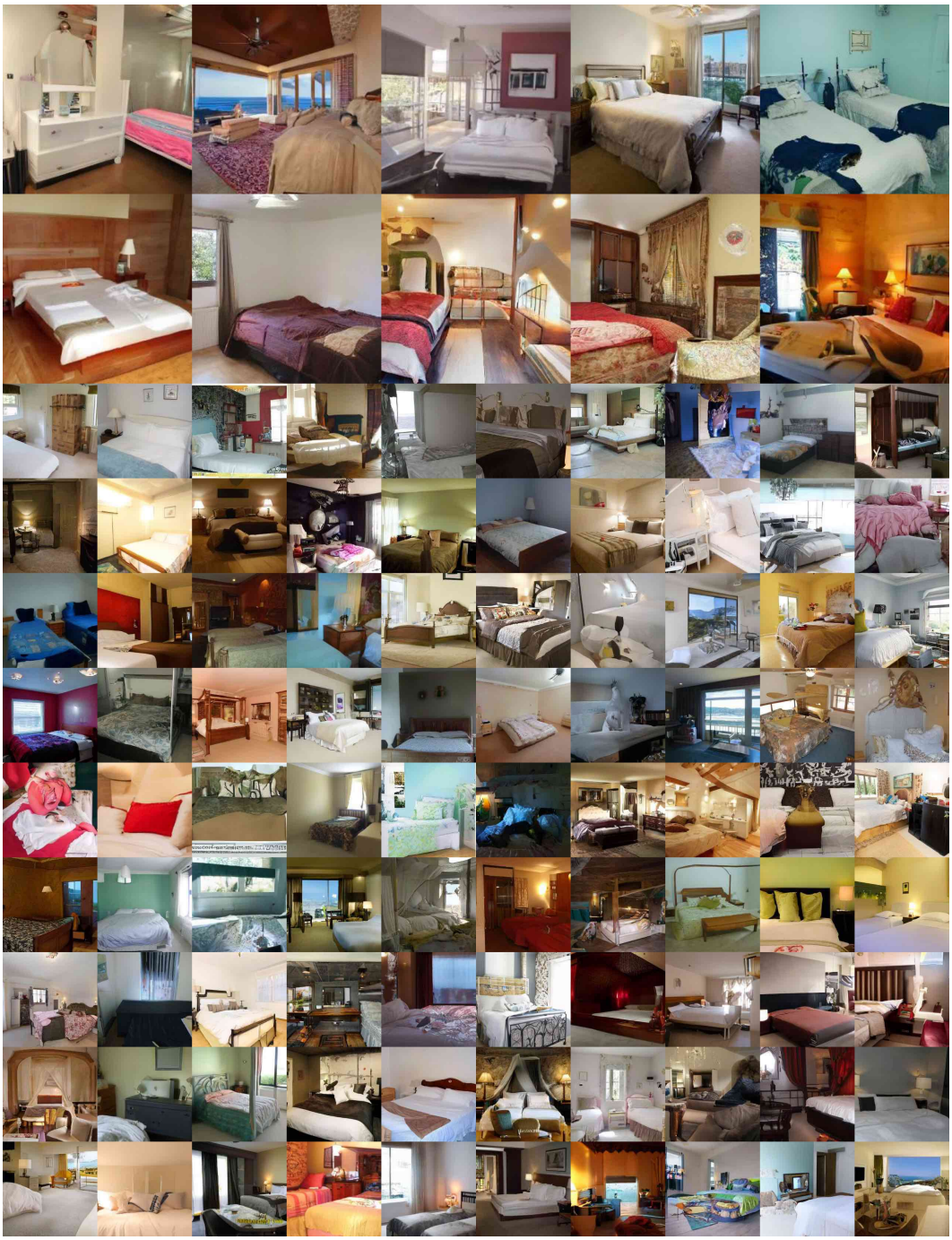


Figure 18: LSUN Bedroom generated samples, small model. FID=6.36

Figure 19: LSUN Cat generated samples. FID=19.75