Denoising Diffusion Probabilistic Models

Authors

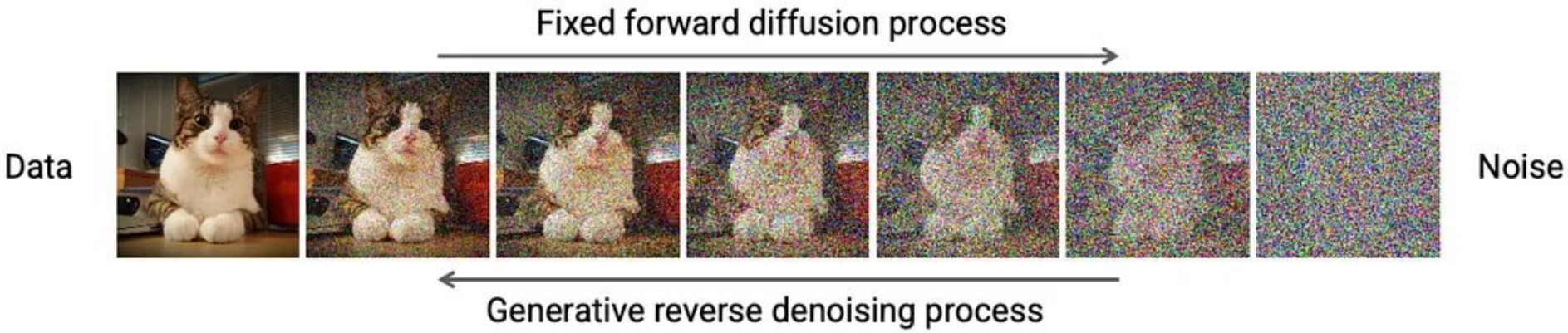
Jonathan Ho UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Denoising Diffusion Probabilistic Models (DDPM)

- A technique used in machine learning to generate new data that resembles a given dataset, a task known as data generation.
- Unlike some other models that work by classifying or differentiating data, DDPMs are generative models.
- Are designed to create new data closely matching a given set of real data.
- Start with random noise and then iteratively refine it to form a coherent data sample, a process guided by a neural network
- The data is being refined at each step to resemble real data.
- Refining the data is controlled by a schedule of noise reduction levels, which are applied at each step of the generation process

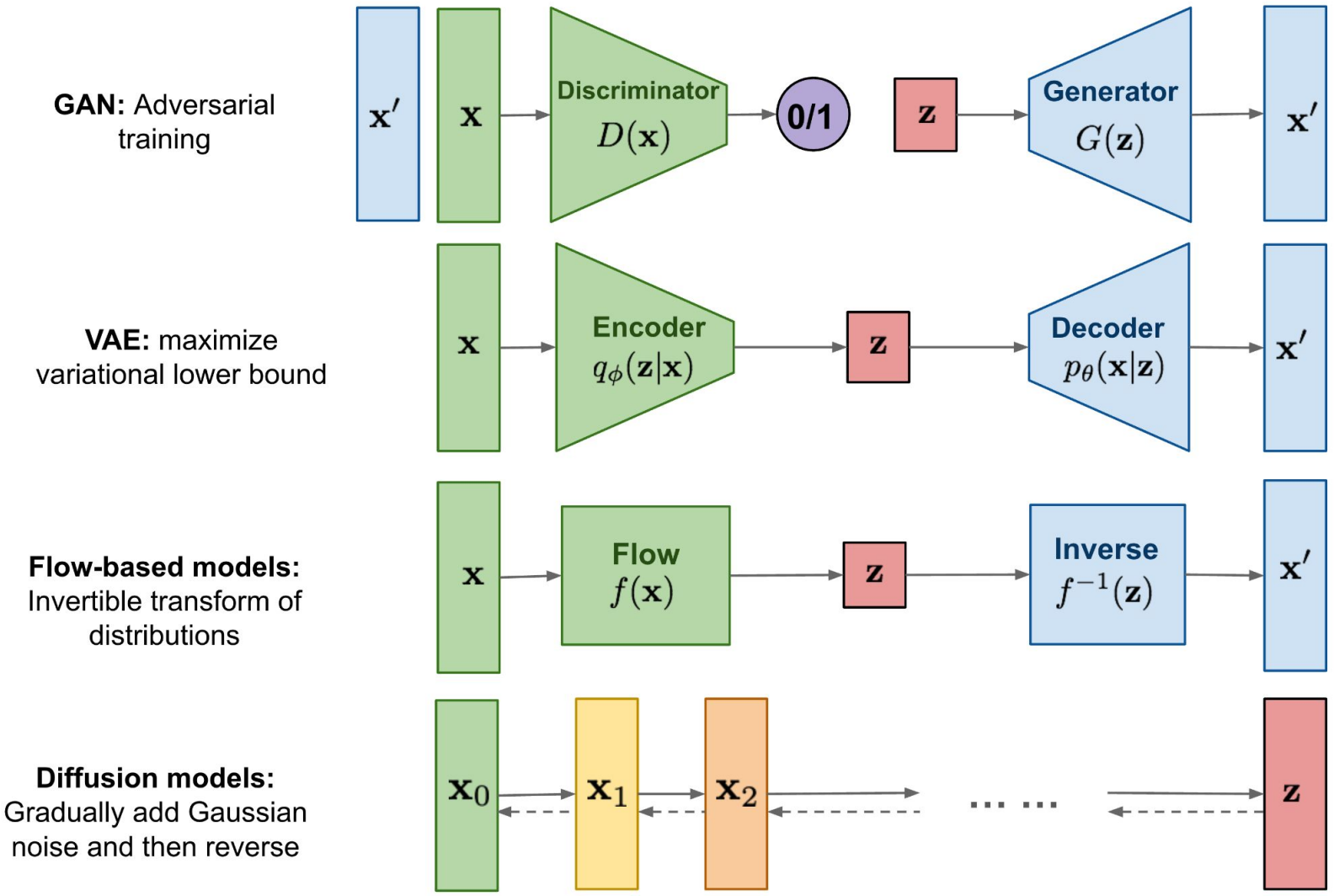


Background

Diffusion model or diffusion probabilistic model Or Score-based generative model

- Inspired by non-equilibrium thermodynamics
- A class of Latent variable generative models
- 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
- 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.
- Unlike VAE or flow models, diffusion models are learned with a fixed procedure and the latent variable has high dimensionality (same as the original data)
- The goal of diffusion models is to learn a diffusion process that generates the probability distribution of a given dataset
- They learn the latent structure of a dataset by modeling the way in which data points diffuse through their latent space.

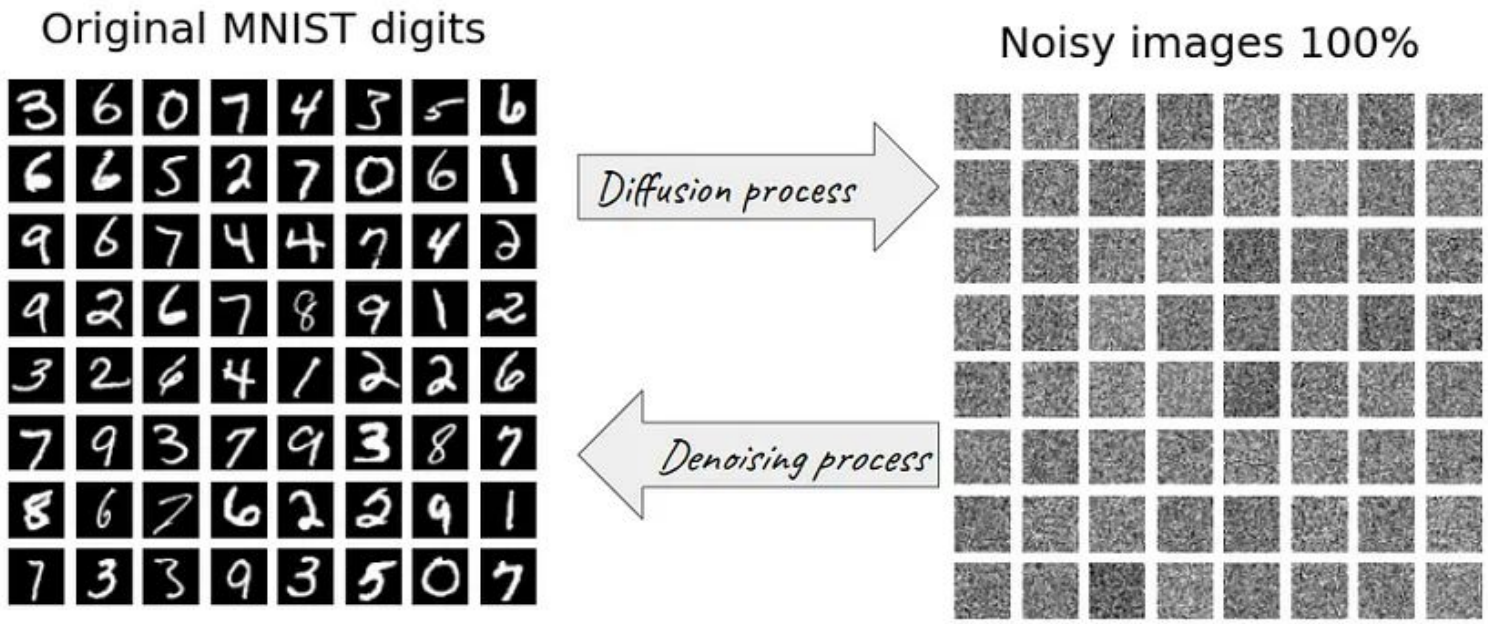
Diffusion models



Diffusion Models

Three Major Components

Forward Process	Reverse Process	The Sampling Procedure



Forward diffusion process that gradually adds noise to input
Reverse denoising process that learns to generate data by denoising

Forward Process

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

Use variational lower bound

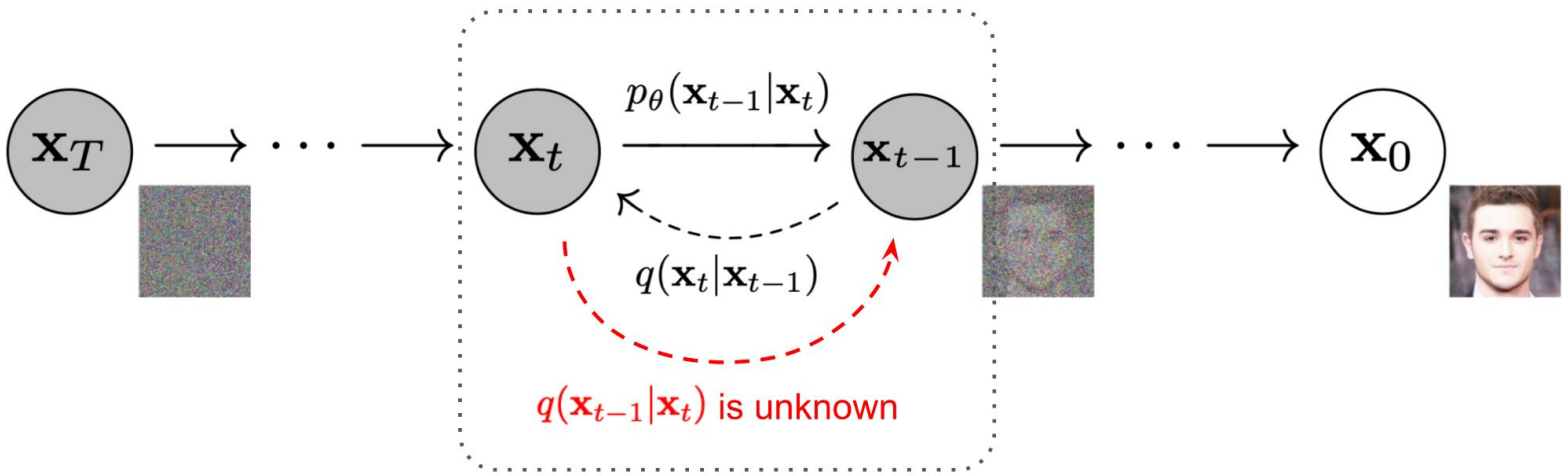


fig. 2. The Markov chain of forward (reverse) diffusion process of generating a sample by slowly adding (removing) noise. (Image source: Ho et al. 2020 with a few additional annotations)

$q(x_t|x_{t-1})$

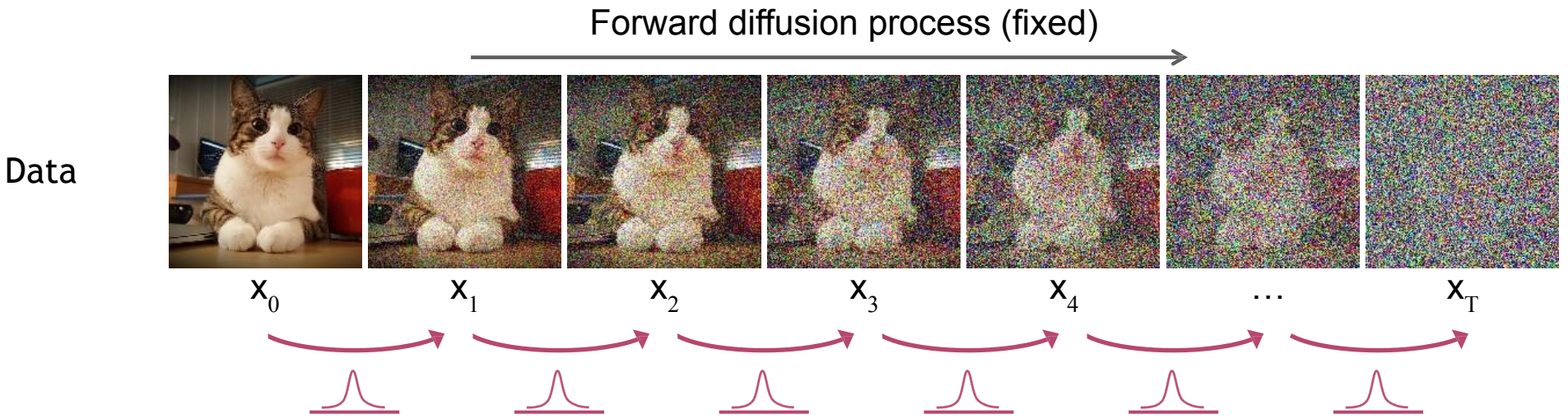
- This term is also known as the *forward diffusion kernel (FDK)*.
- It defines the PDF of an image at timestep t in the forward diffusion process x_t given image x_{t-1}.
- It denotes the "transition function" applied at each step in the *forward diffusion process*

$p_\theta(x_{t-1}|x_t)$

- This term is also known as the *forward diffusion kernel (FDK)*.
- It defines the PDF of an image at timestep t in the forward diffusion process x_t given image x_{t-1}.
- It denotes the "transition function" applied at each step in the *forward diffusion process*

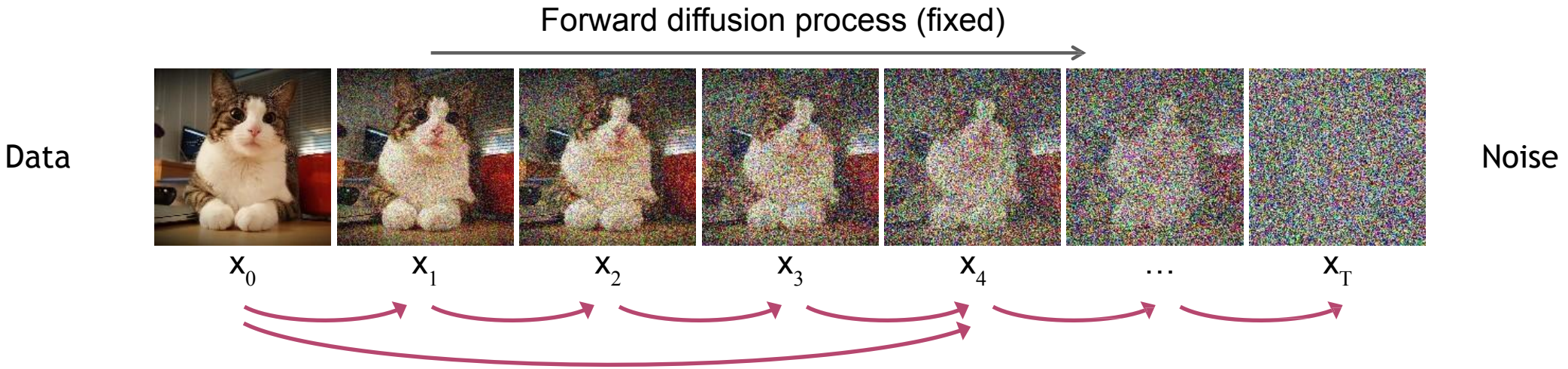
Forward Diffusion Process

The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Diffusion Kernel



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: where $\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$ $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Forward Process

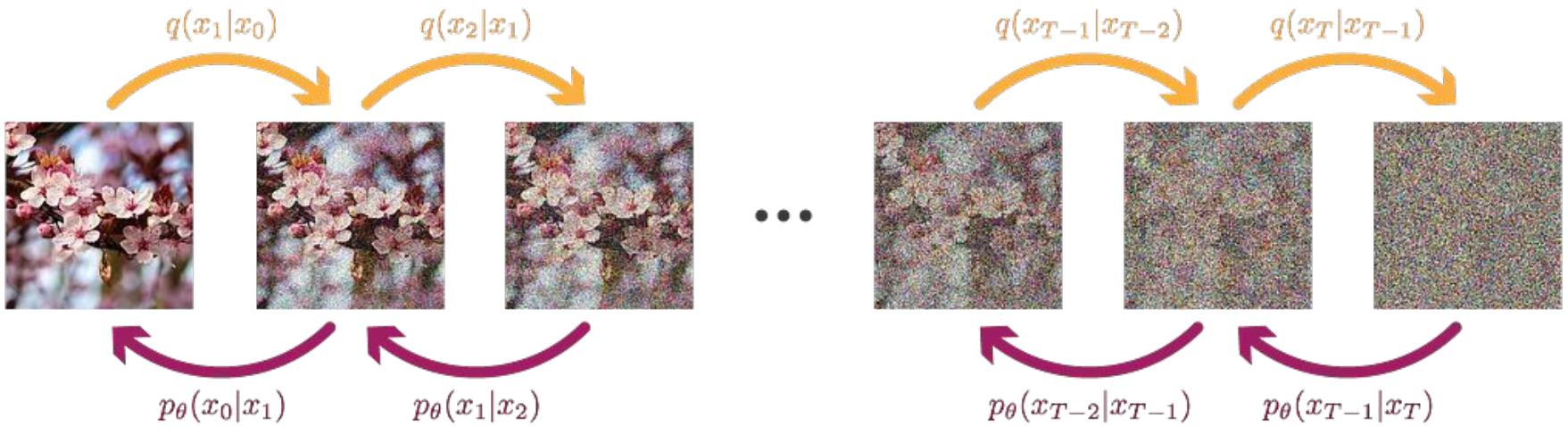
FIXED FORWARD PROCESS

Initial distribution

$$q(x_0)$$

Gaussian transition kernel

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



Approximation of

$$q(x_{t-1}|x_t)$$

Gaussian transition kernel with parameters to be learned

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Initial distribution

$$p(x_T) = \mathcal{N}(x_t; 0, I)$$

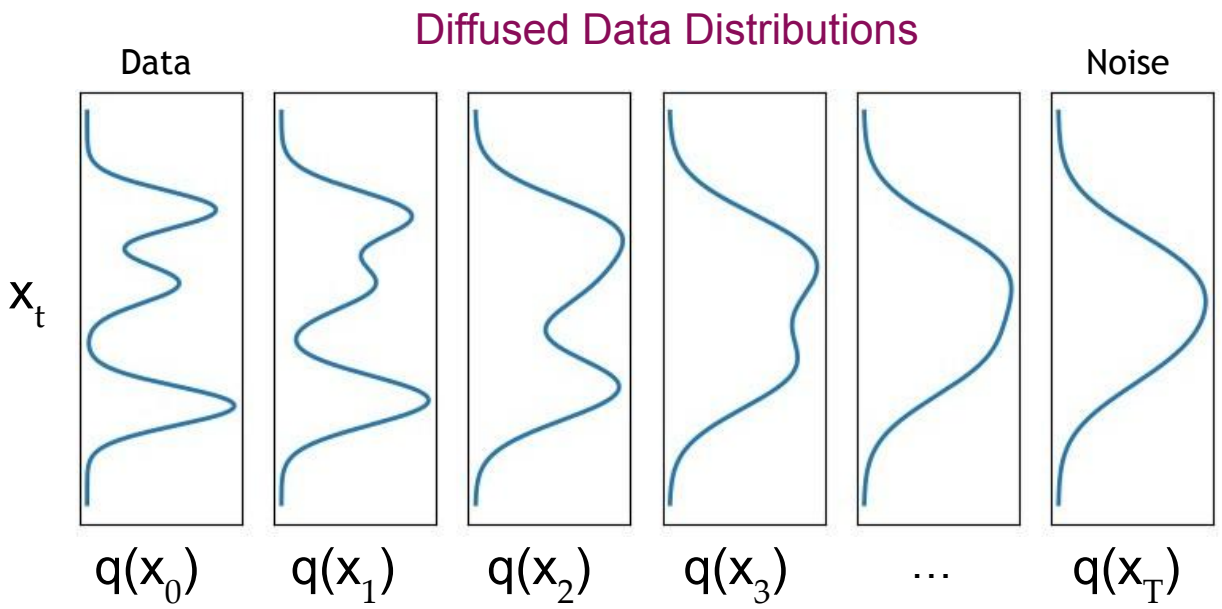
LEARNED BACKWARD PROCESS

What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

Generative Learning by Denoising

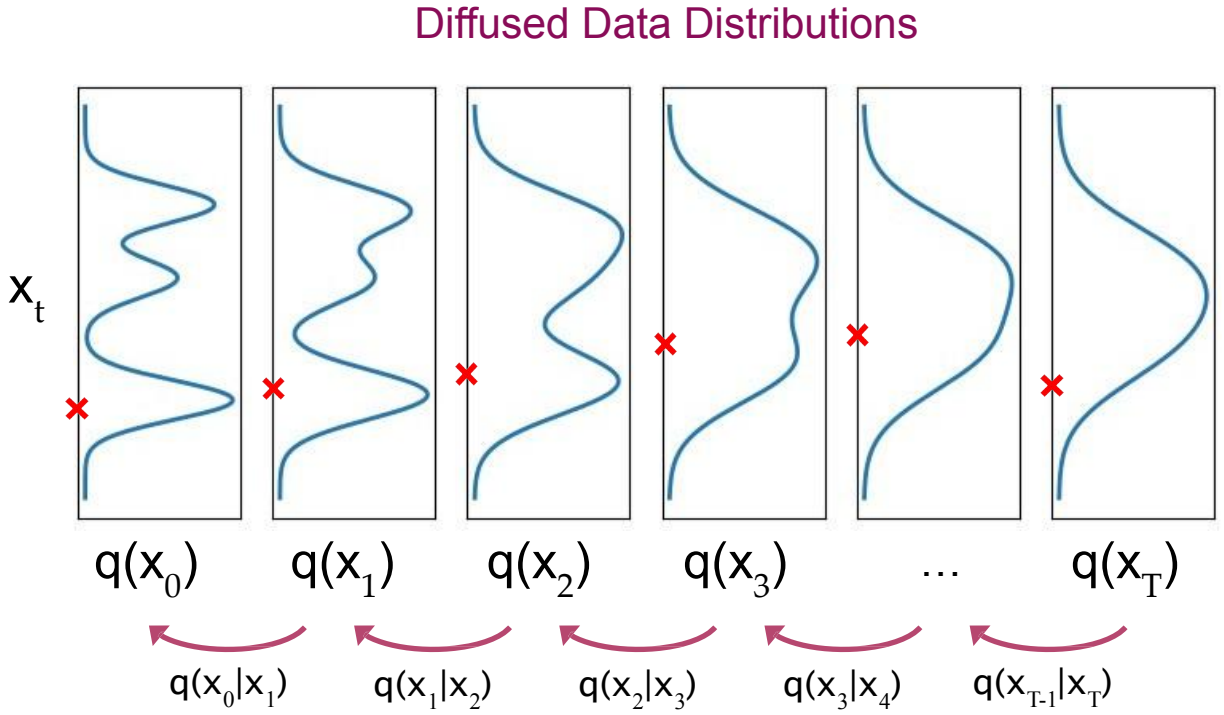
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}$

True Denoising Dist.



In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

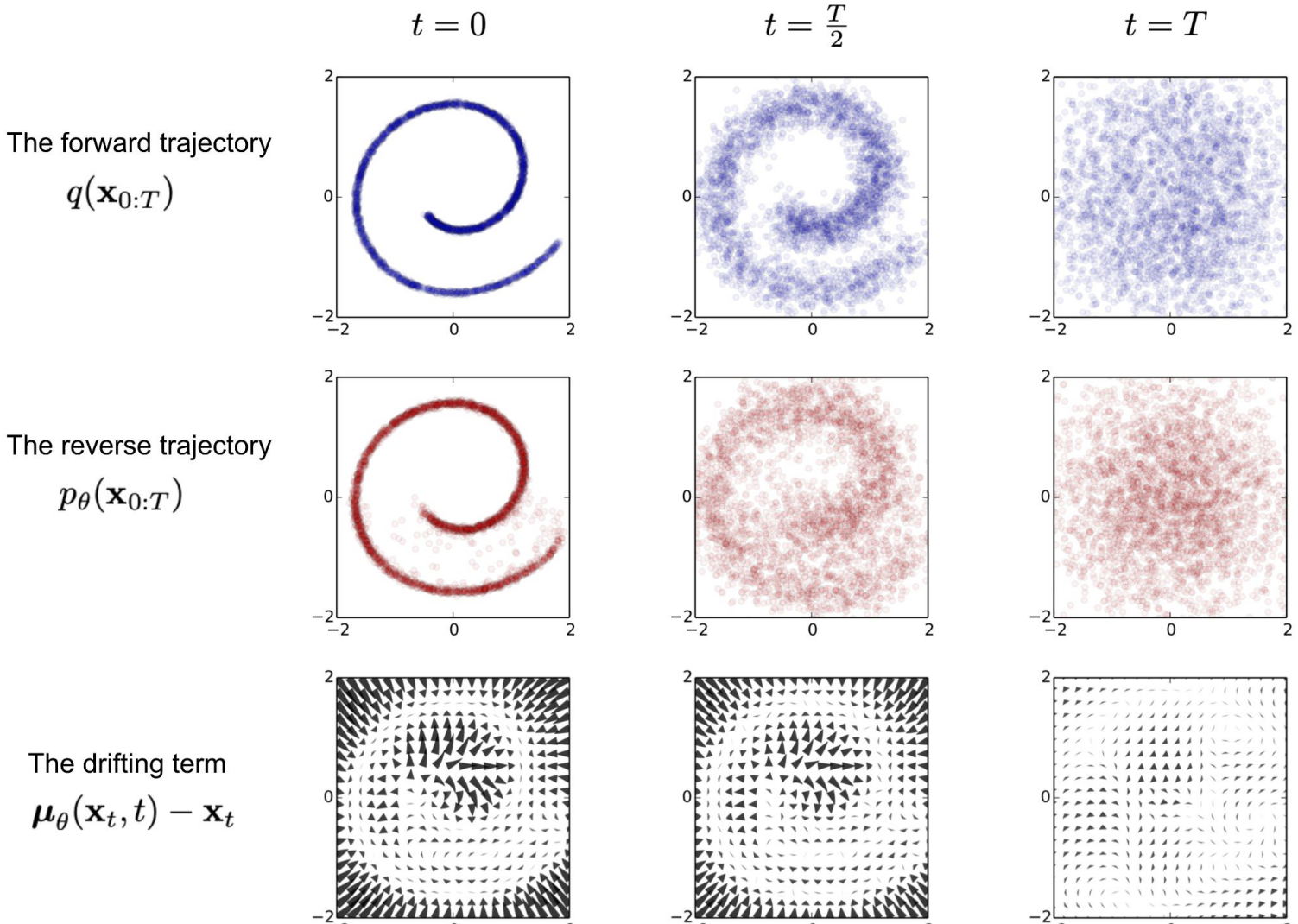
Forward Process

ignore the fact that the forward process variances β_t are learnable by reparameterization and instead fix them to constants .

Thus, in their implementation, the approximate posterior q has no learnable parameters, so LT is a constant during training and can be ignored

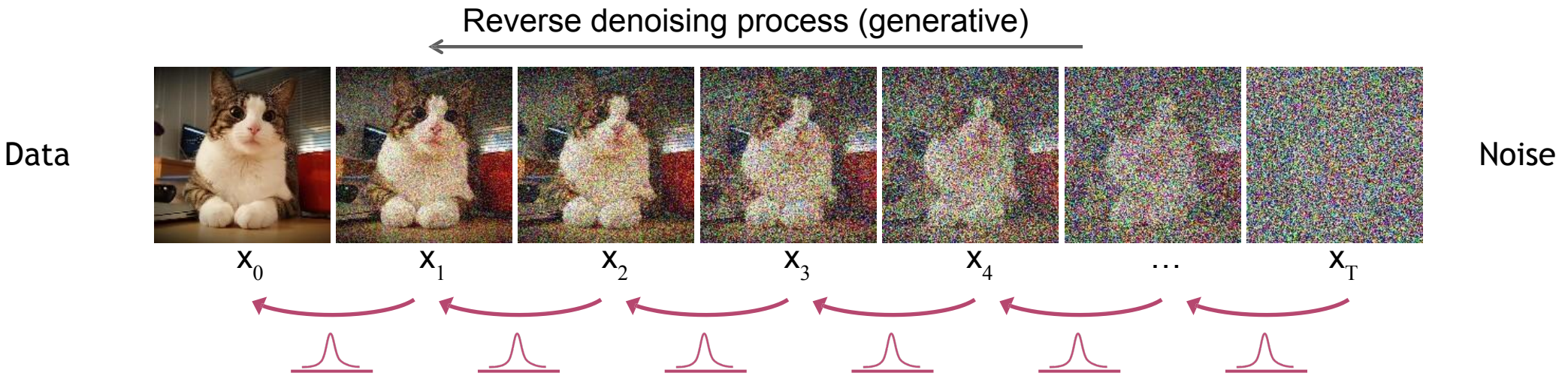
Reverse Process

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



Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

➡

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Learning Denoising Model : Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

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

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

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

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is the tractable posterior distribution:

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

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

Parameterizing the Denoising Model

Since both $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

They propose to represent the mean of the denoising model using a *noise-prediction* network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

With this parameterization

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right] + C$$

Training Objective Weighting : Trading likelihood for perceptual quality

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \Big]$$

λ_t

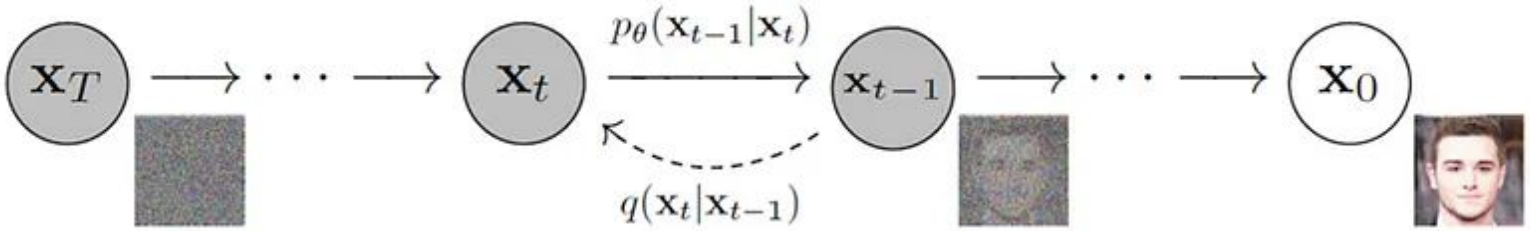
The time dependent ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small t's.

observe that simply setting $\lambda_t = 1$ improves sample quality. So, they propose to use:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t)||^2 \right]$$

Summary



- **Forward** $q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$

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

  $q(x_t|x_0) = N\left(x_t;\ \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$

  ⋯ where $\alpha_t := 1-\beta_t$ and $\overline{\alpha_t} := \prod_{s=1}^{T} \alpha_s$

- **Posterior** $q(x_{t-1}|x_t, x_0) = N(x_{t-1};\ \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$

  ⋯ where $\beta_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

- **Backward** $p_\theta(x_{t-1}|x_t) := N(x_{t-1};\ \mu_\theta(x_t, t), \sum_\theta(x_t, t))$

- **Loss Function** $D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

Summary

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: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
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** $\mathbf{x}_0$

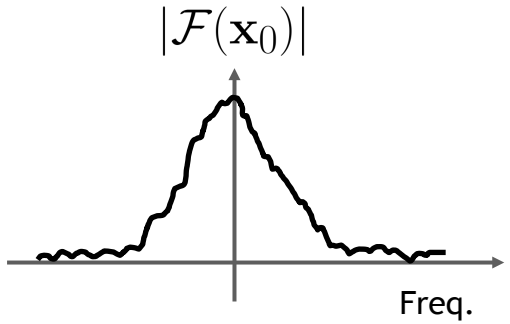
What happens to an image in the forward diffusion process?

Recall that sampling from $q(\mathbf{x}_t|\mathbf{x}_0)$ is done using $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$$

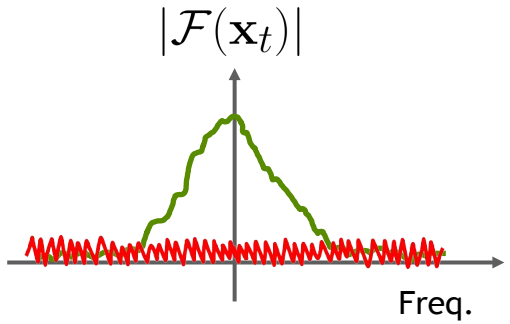
Fourier Transform

$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\,\mathcal{F}(\mathbf{x}_0) + \sqrt{(1-\bar{\alpha}_t)}\,\mathcal{F}(\epsilon)$$



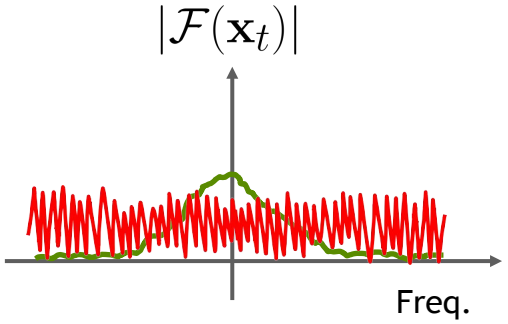
Small *t*

$\bar{\alpha}_t \sim 1$



Large *t*

$\bar{\alpha}_t \sim 0$



In the forward diffusion, the high frequency content is perturbed faster.

Training

$$\mathbb{E}_{x_0,t,\epsilon}\left(||\epsilon - \epsilon_\theta(x_t, t)||^2\right) = \mathbb{E}_{x_0,t,\epsilon}\left(||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right)$$

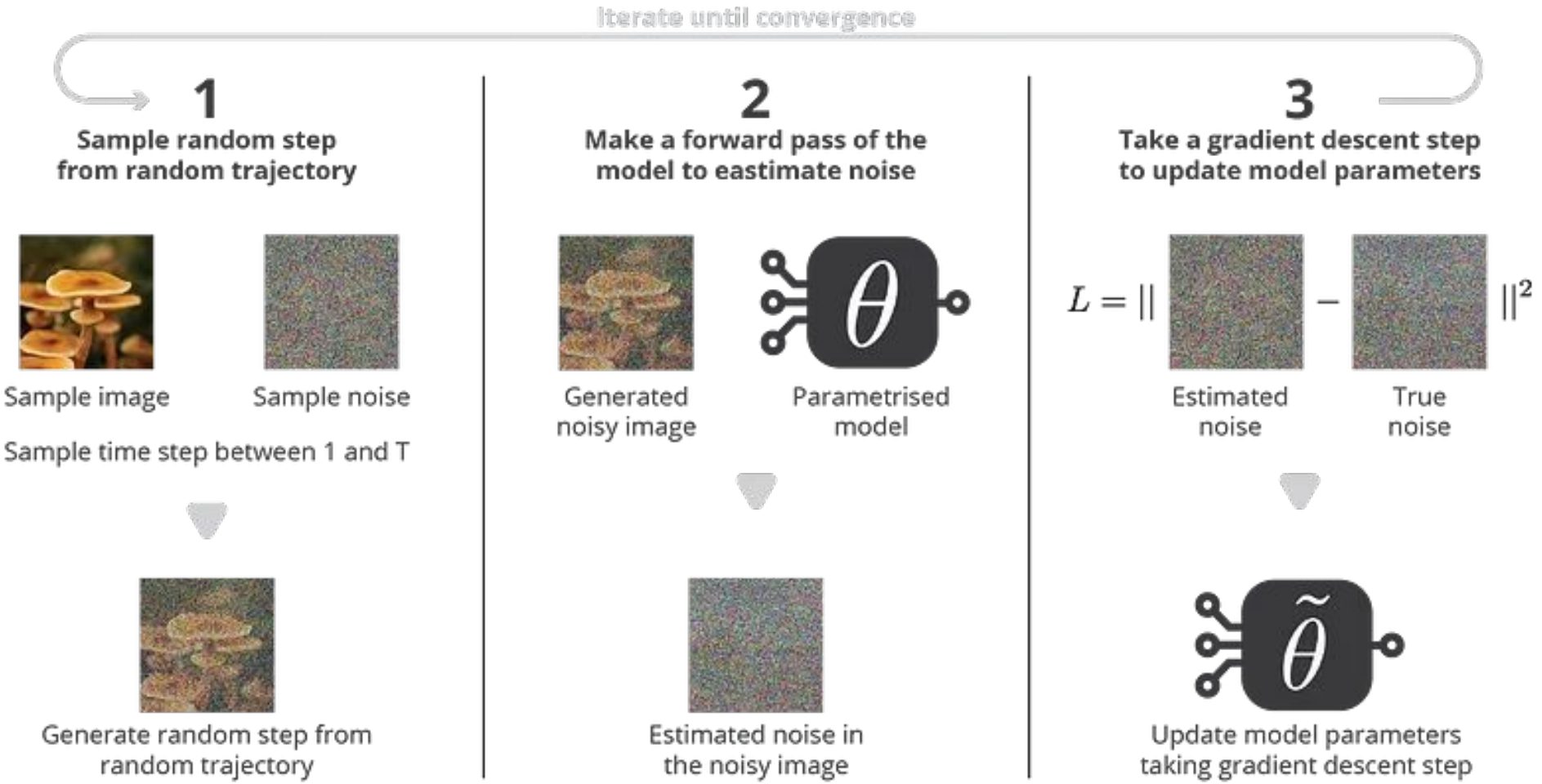


Illustration of the training process of a denoising diffusion probabilistic model training.

Samplaing

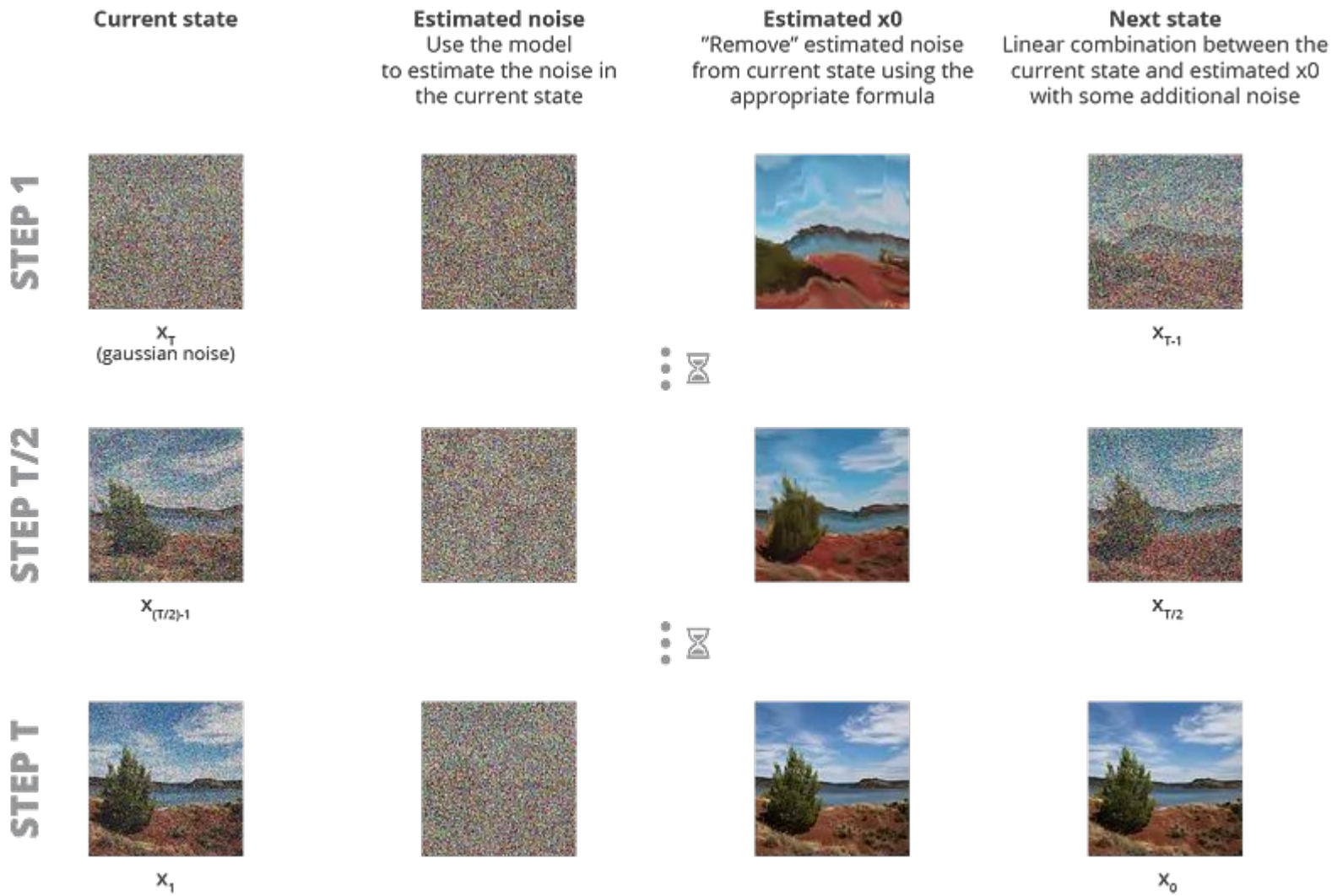


Illustration of the sampling process of a denoising diffusion probabilistic model.

Experiments

- T = 1000
- forward process $\beta 1 = 10^{-4}$ to $\beta T = 0.02$
- *These constants were chosen to be small relative to data scaled to [−1, 1]*
- Reverse and forward processes have approximately the same functional form while keeping the signal-to-noise ratio at xT as small as possible as $LT = DKL(q(xT|x0) \; k \; N(0, I)) \approx 10^{-5}$
- To represent the reverse process, used a U-Net backbone similar to an unmasked PixelCNN++ with group normalization throughout . Parameters are shared across time, which is specified to the network using the Transformer sinusoidal position embedding. We use self-attention at the 16 × 16 feature map resolution.

Inception Scores

The Inception Score (IS) is an algorithm used to assess the quality of images created by a generative image model such as a generative adversarial network (GAN)

The **Inception Score** of p_{gen} relative to p_{dis} is

$$IS(p_{gen}, p_{dis}) := \exp\left(\mathbb{E}_{x\sim p_{gen}}\left[D_{KL}\left(p_{dis}(\cdot|x)\| \int p_{dis}(\cdot|x)p_{gen}(x)dx\right)\right]\right)$$

Equivalent rewrites include

$$\ln IS(p_{gen}, p_{dis}) := \mathbb{E}_{x\sim p_{gen}}\left[D_{KL}\left(p_{dis}(\cdot|x)\|\mathbb{E}_{x\sim p_{gen}}[p_{dis}(\cdot|x)]\right)\right]$$

$$\ln IS(p_{gen}, p_{dis}) := H[\mathbb{E}_{x\sim p_{gen}}[p_{dis}(\cdot|x)]] - \mathbb{E}_{x\sim p_{gen}}[H[p_{dis}(\cdot|x)]]$$

$\ln IS$ is nonnegative by Jensen's inequality.

Experiments - Sample Quality

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

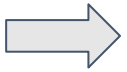
Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	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 shows Inception scores, FID scores, and negative log likelihoods (lossless codelengths) on CIFAR10. With FID score of 3.17, used unconditional model achieves better sample quality than most models in the literature, including class conditional models. This 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.

Experiments - Reverse process parameterization and training objective ablation

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



the sample quality effects of reverse process parameterizations and training objectives.That the baseline option of predicting μ˜ works well only when trained on the true variational bound instead of unweighted mean squared error, a simplified objective akin to. Also see that learning reverse process variances (by incorporating a parameterized diagonal Σθ(xt) into the variational bound) leads to unstable training and poorer sample quality compared to fixed variances. Predicting , as we proposed, performs approximately as well as predicting μ˜ when trained on the variational bound with fixed variances, but much better when trained with our simplified objective

Experiments

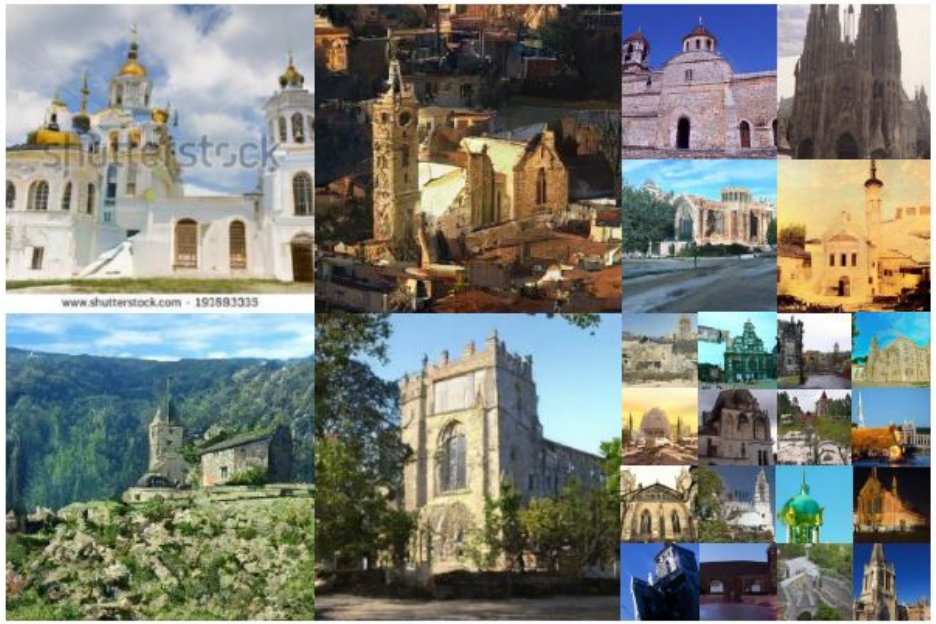


Figure 3: LSUN Church samples. FID=7.89

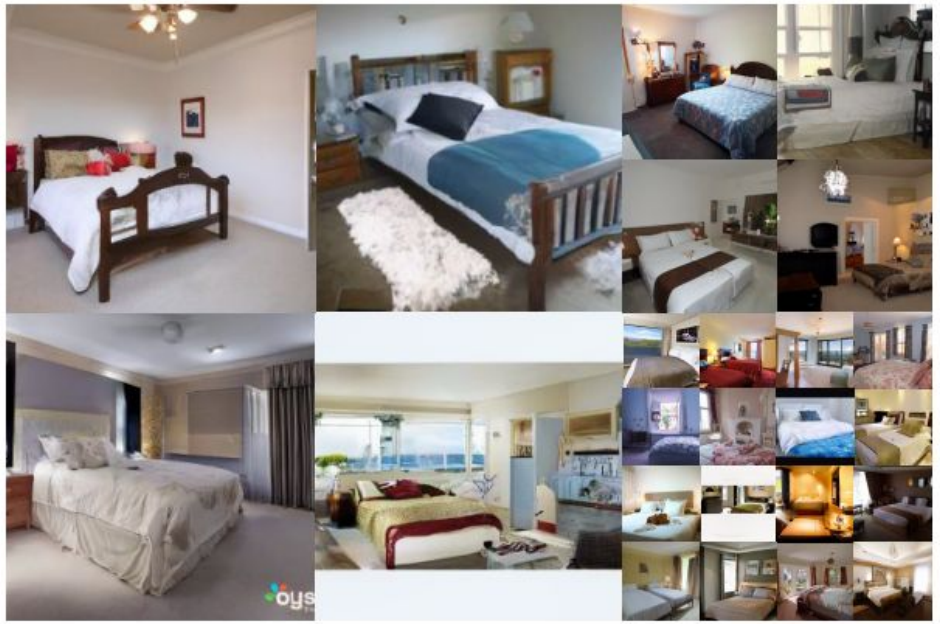


Figure 4: LSUN Bedroom samples. FID=4.90

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

Experiments

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	-	-

Experiments - Progressive coding

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. Still, while lossless code lengths are better than the large estimates reported for energy based models and score matching using annealed importance sampling, they are not competitive with other types of likelihood-based generative models. Since used samples are nonetheless of high quality, IT conclude that diffusion models have an inductive bias that makes them excellent lossy compressors. Treating the variational bound terms L1 +· · ·+LT as rate and L0 as distortion, The 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.

Experiments - Progressive lossy compression

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

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.

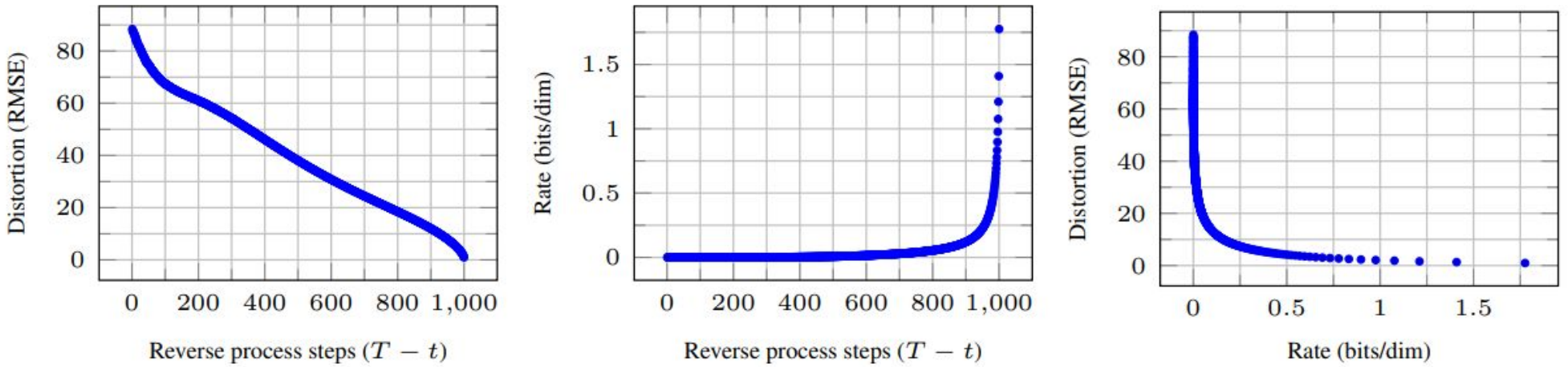


Figure 5: Unconditional CIFAR10 test set rate-distortion vs. time. Distortion is measured in root mean squared error on a $[0, 255]$ scale.

Experiments - Connection to autoregressive decodin

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

Now consider setting the diffusion process length T to the dimensionality of the data, defining the forward process so that q(xt|x0) places all probability mass on x0 with the first t coordinates masked out (i.e. q(xt|xt−1) masks out the t th coordinate), setting p(xT) to place all mass on a blank image, and, for the sake of argument, taking pθ(xt−1|xt) to be a fully expressive conditional distribution. With these choices, DKL(q(xT) k p(xT)) = 0, and minimizing DKL(q(xt−1|xt) k pθ (xt−1|xt)) trains pθ to copy coordinates t + 1, . . . , T unchanged and to predict the t th coordinate given t + 1, . . . , T. Thus, training pθ with this particular diffusion is training an autoregressive model.

Experiments - Interpolation

interpolate source images x0, x 0 0 ~ q(x0) in latent space using q as a stochastic encoder, xt, x 0 t ~ q(xt|x0), then decoding the linearly interpolated latent $\bar{x}t = (1 - \lambda)x0 + \lambda x 0 0$ into image space by the reverse process, $\bar{x}0 \sim p(x0|\bar{x}t)$. In effect,used the reverse process to remove artifacts from linearly interpolating corrupted versions of the source images, as depicted in Fig. 8 (left). fixed the noise for different values of λ so xt and x 0 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

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

Experiments - Interpolation

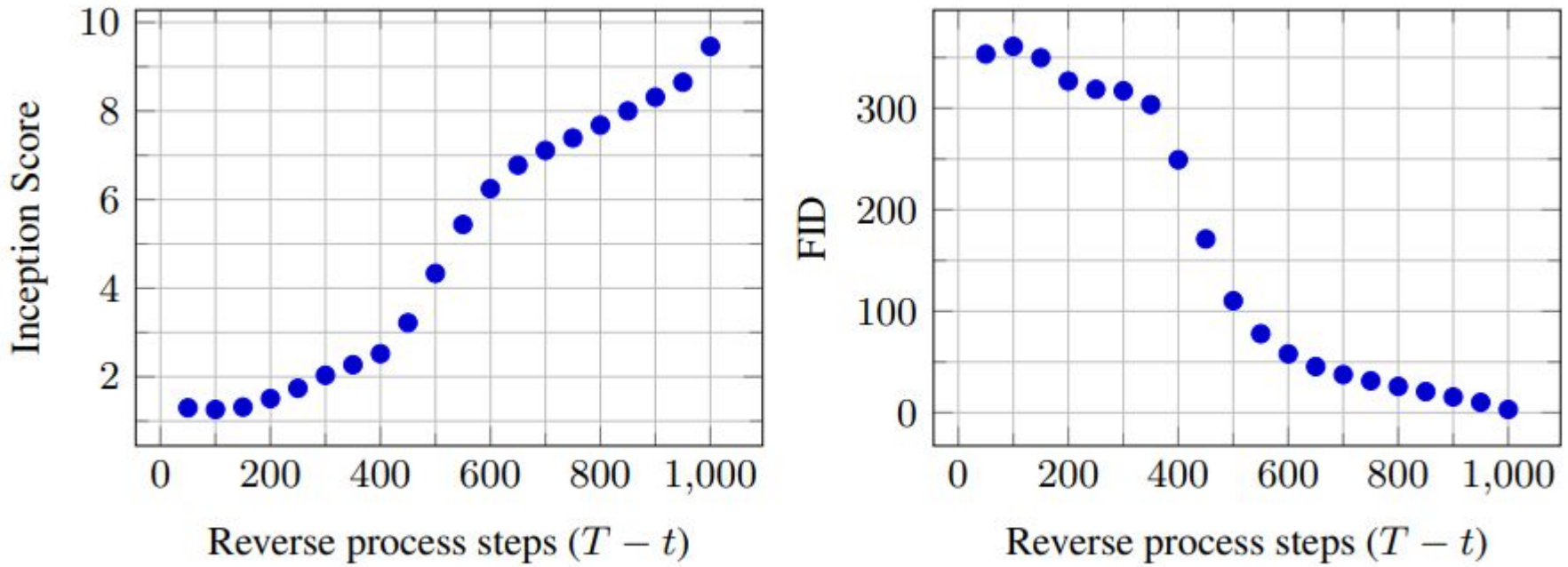


Figure 10: Unconditional CIFAR10 progressive sampling quality over time

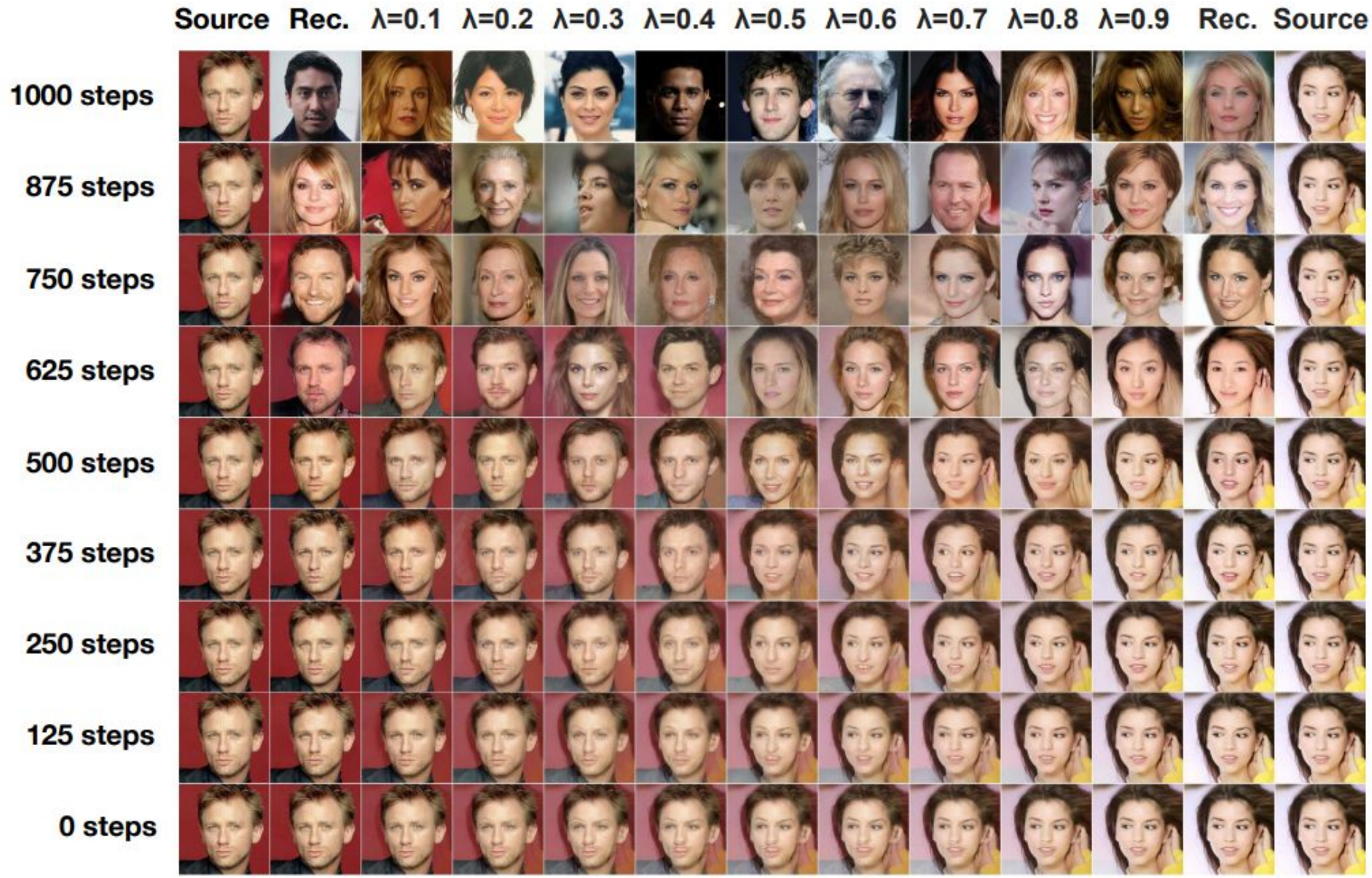


Figure 9: Coarse-to-fine interpolations that vary the number of diffusion steps prior to latent mixing.

Experiments - Progressive generation

progressive unconditional generation process given by progressive decompression from random bits. In other words, we predict the result of the reverse process, xˆ0, while sampling from the reverse process using Algorithm 2. *Figures 6* and *10* show the resulting sample quality of xˆ0 over the course of the reverse process. Large scale image features appear first and details appear last. *Figure 7* shows stochastic predictions x0 ~ pθ(x0|xt) with xt 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

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).

Experiments - Progressive generation

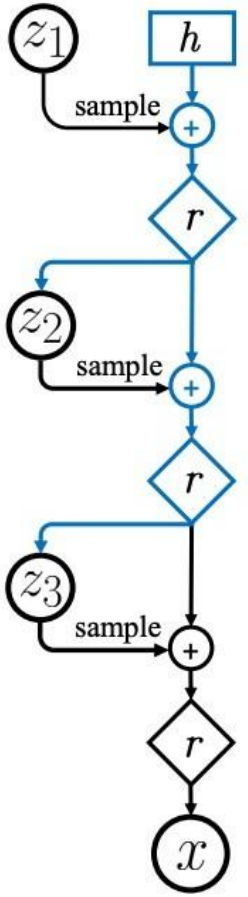
Figure 7: When conditioned on the same latent, CelebA-HQ 256×256 samples share high-level attributes. Bottom-right quadrants are $\mathbf{x}_t$, and other quadrants are samples from $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Connection to VAEs

Diffusion models can be considered as a special form of hierarchical VAEs.

However, in diffusion models:

- The encoder is fixed
- The latent variables have the same dimension as the data
- The denoising model is shared across different timestep
- The model is trained with some reweighting of the variational bound.



Additional References

- Archive Paper → https://arxiv.org/pdf/2006.11239.pdf
- https://generativeai.pub/denoising-diffusion-probabilistic-models-from-scratch-728df8228565
- https://towardsdatascience.com/understanding-the-denoising-diffusion-probabilistic-model-the-socratic-way-445c1bdc5756
- https://towardsdatascience.com/understanding-diffusion-probabilistic-models-dpms-1940329d6048
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- https://medium.com/@gitau_am/a-friendly-introduction-to-denoising-diffusion-probabilistic-models-cc76b8abef25
- https://en.wikipedia.org/wiki/Diffusion_model
- https://en.wikipedia.org/wiki/Inception_score
- https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a

– Peer Reviews on Paper

- https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Review.html
- https://medium.com/@AriaLeeNotAriel/numbynum-denoising-diffusion-probabilistic-models-reviewed-2b1aff8bb9a5

Extended derivations

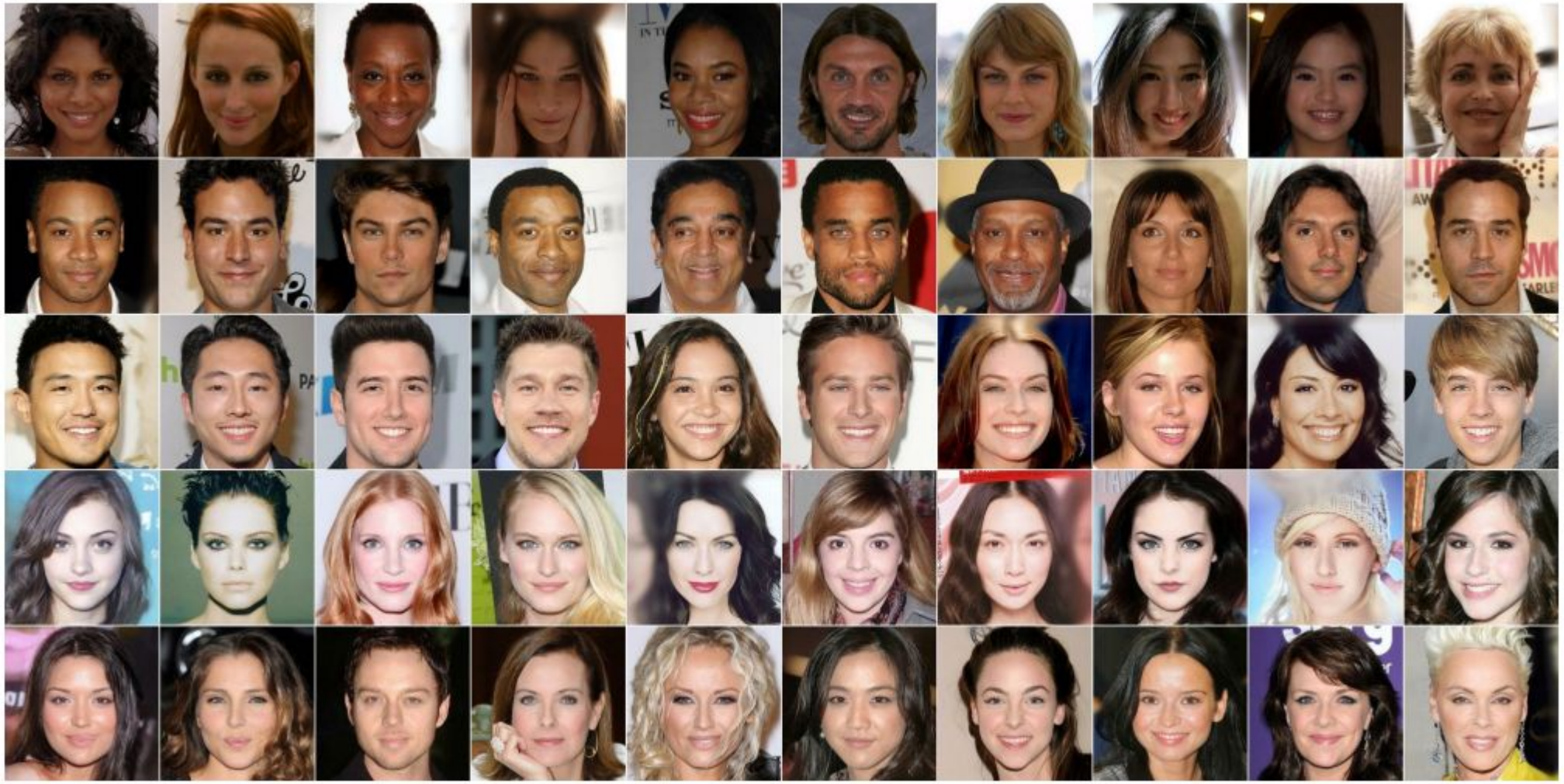
$$
\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q\left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]
\end{aligned}
$$

Alternate version of L

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

CelebA-HQ 256 × 256 generated samples

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

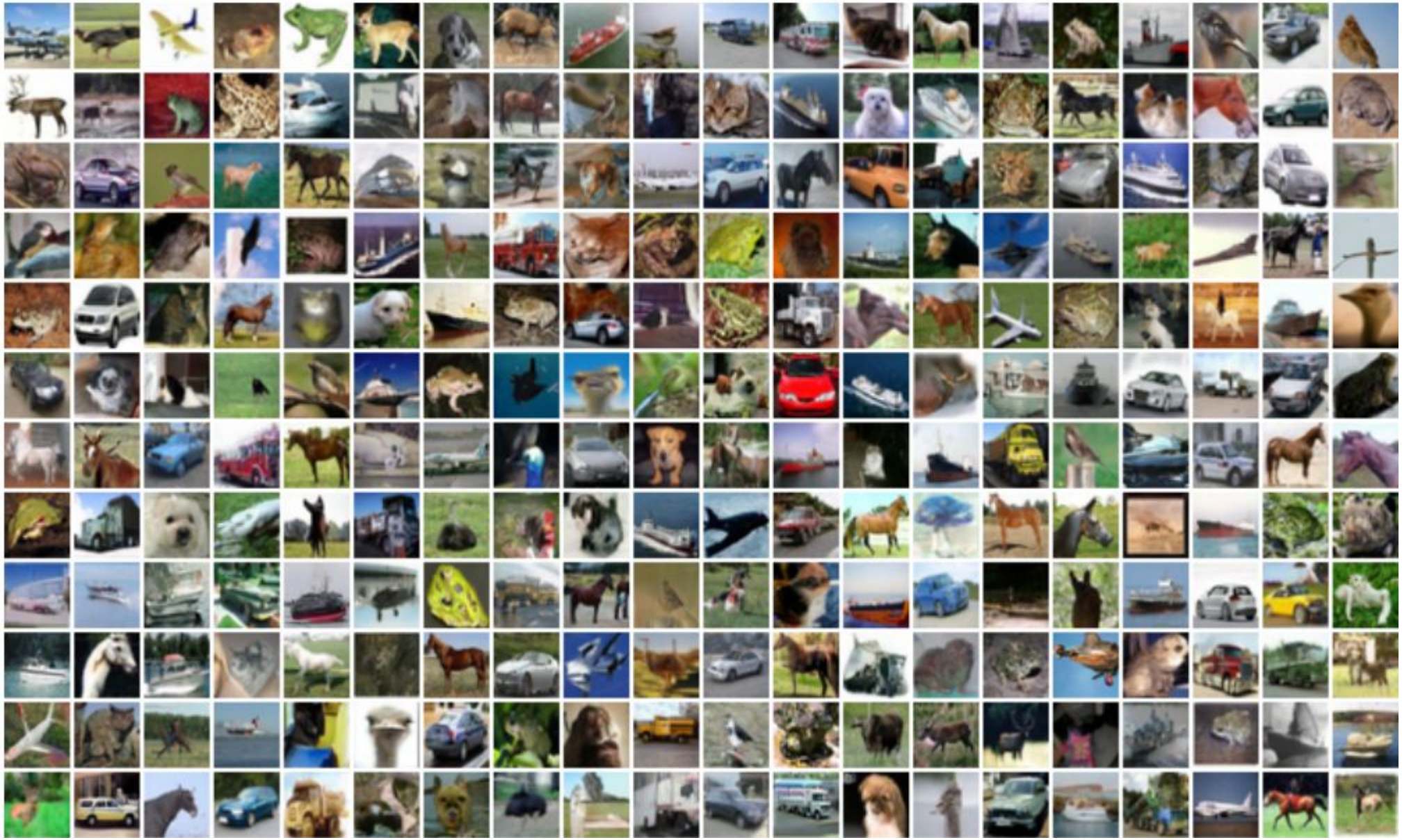


(a) Pixel space nearest neighbors

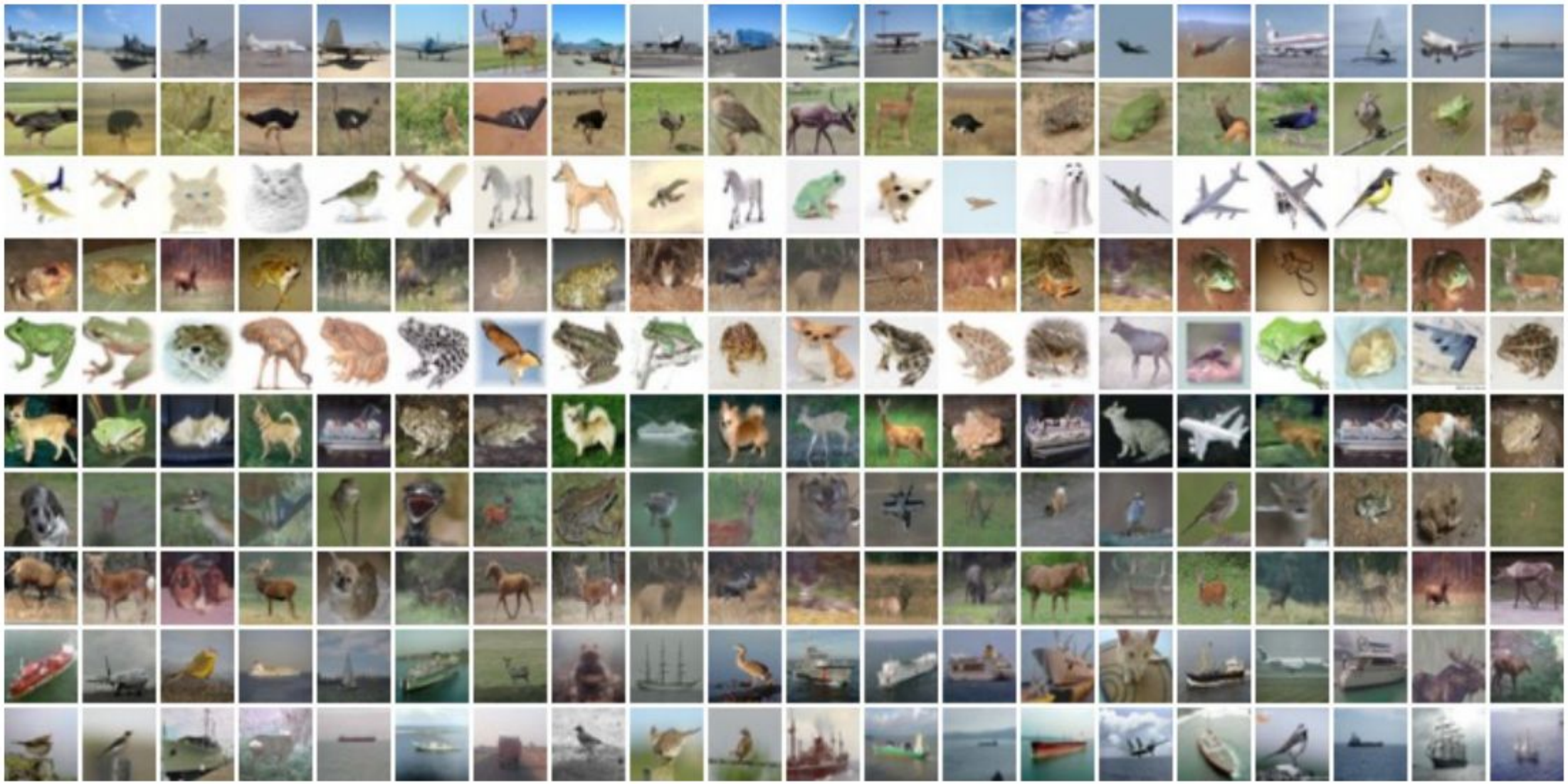
CelebA-HQ 256 × 256 nearest neighbors, computed on a 100 × 100 crop surrounding the faces. Generated samples are in the leftmost column, and training set nearest neighbors are in the remaining columns

(b) Inception feature space nearest neighbors

Unconditional CIFAR10 generated samples

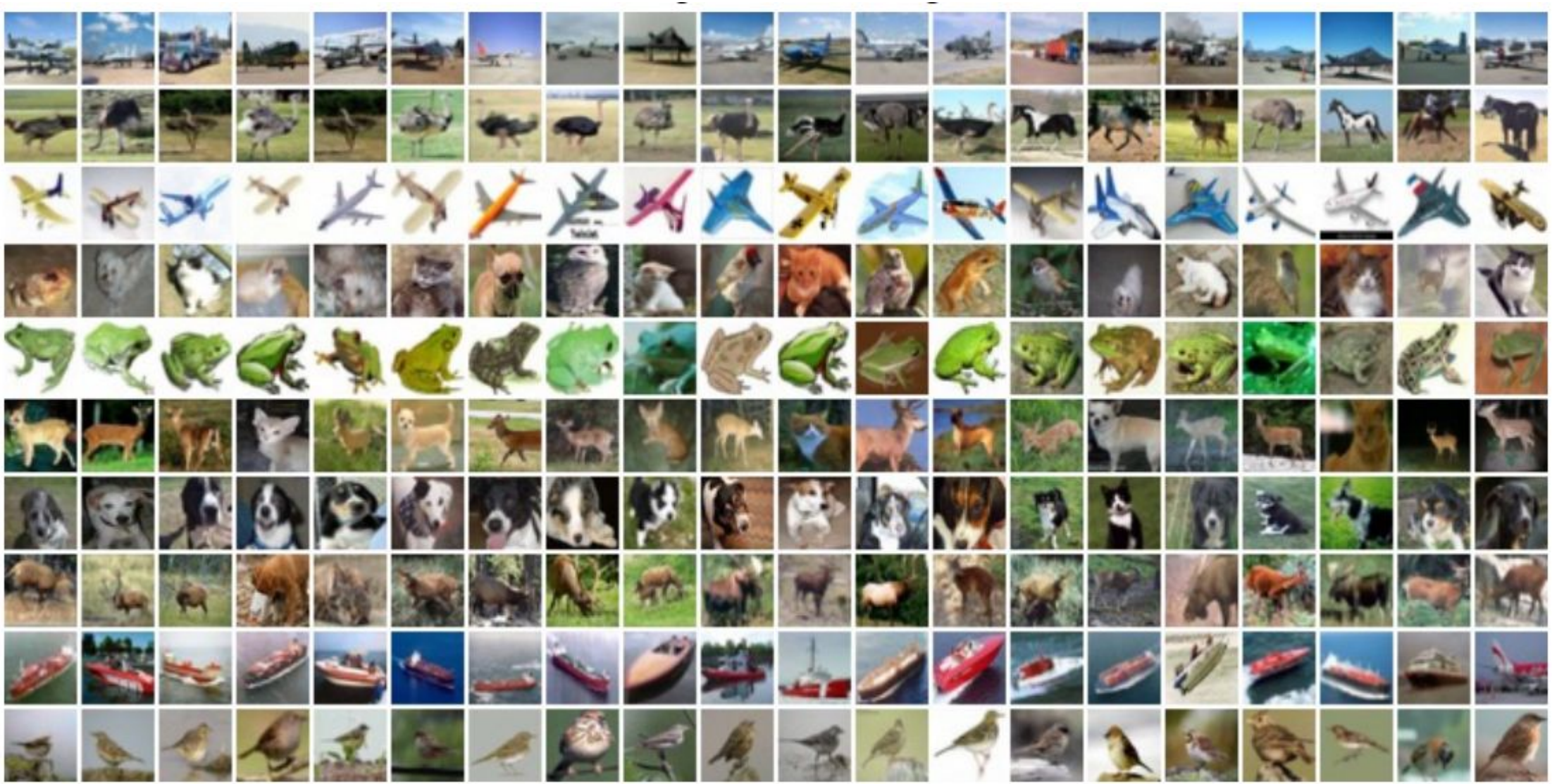


Unconditional CIFAR10 nearest neighbors. Generated samples are in the leftmost column, and training set nearest neighbors are in the remaining columns.



(a) Pixel space nearest neighbors

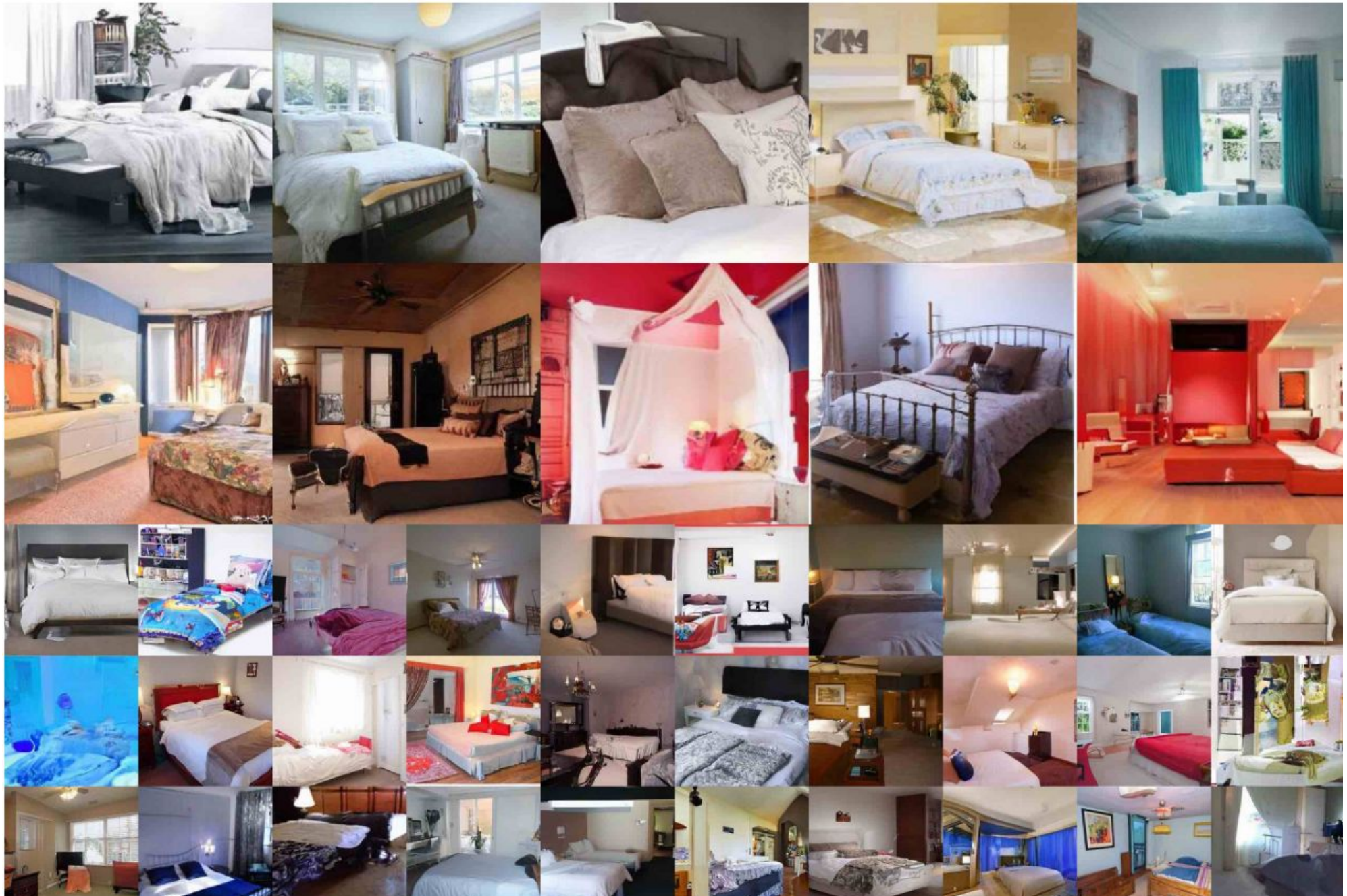
Unconditional CIFAR10 nearest neighbors. Generated samples are in the leftmost column, and training set nearest neighbors are in the remaining columns.



(b) Inception feature space nearest neighbors

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

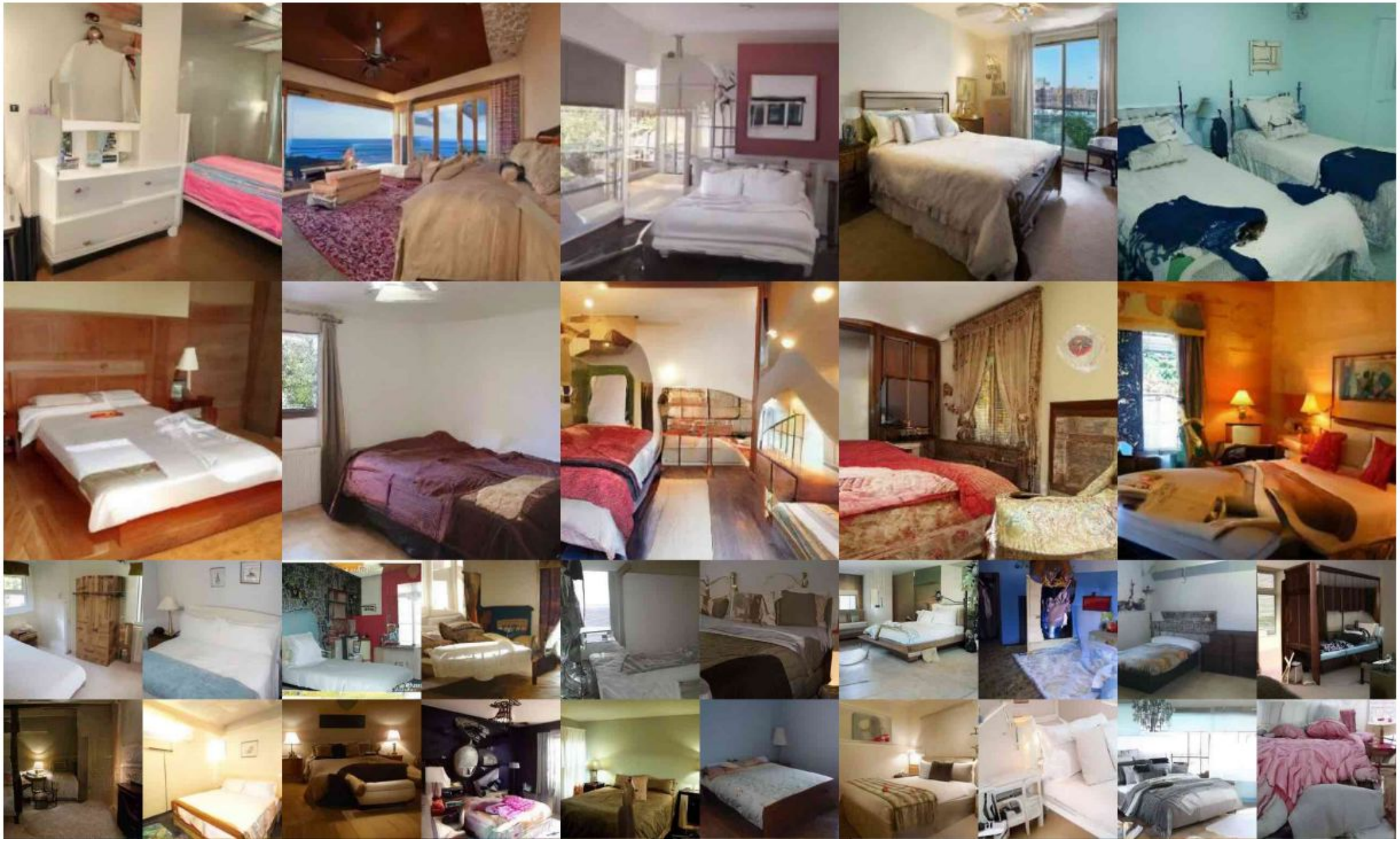
Figure 17: LSUN Bedroom generated samples, large model. FID=4.90



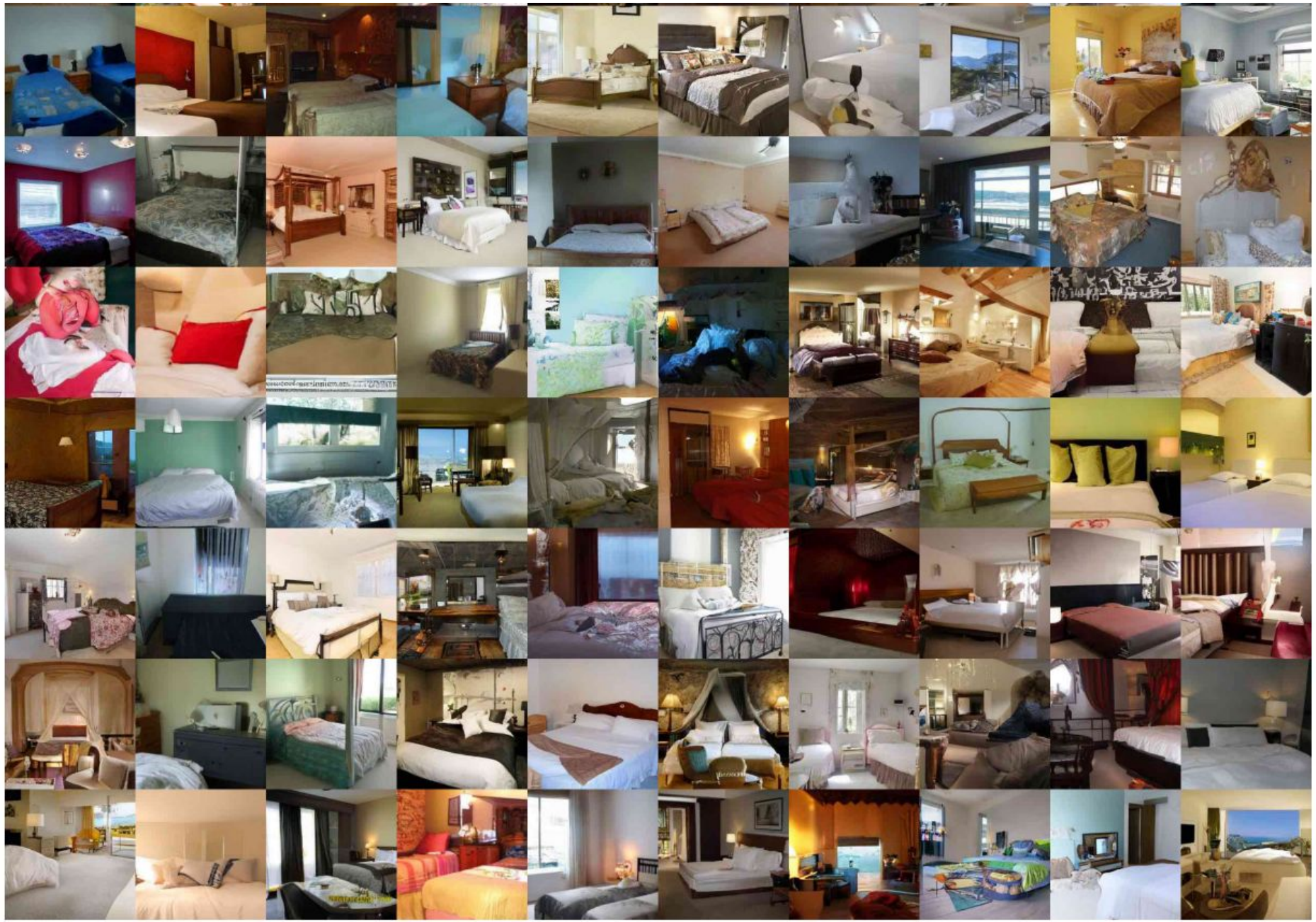
LSUN Church generated samples. FID=7.89

LSUN Church generated samples. FID=7.89

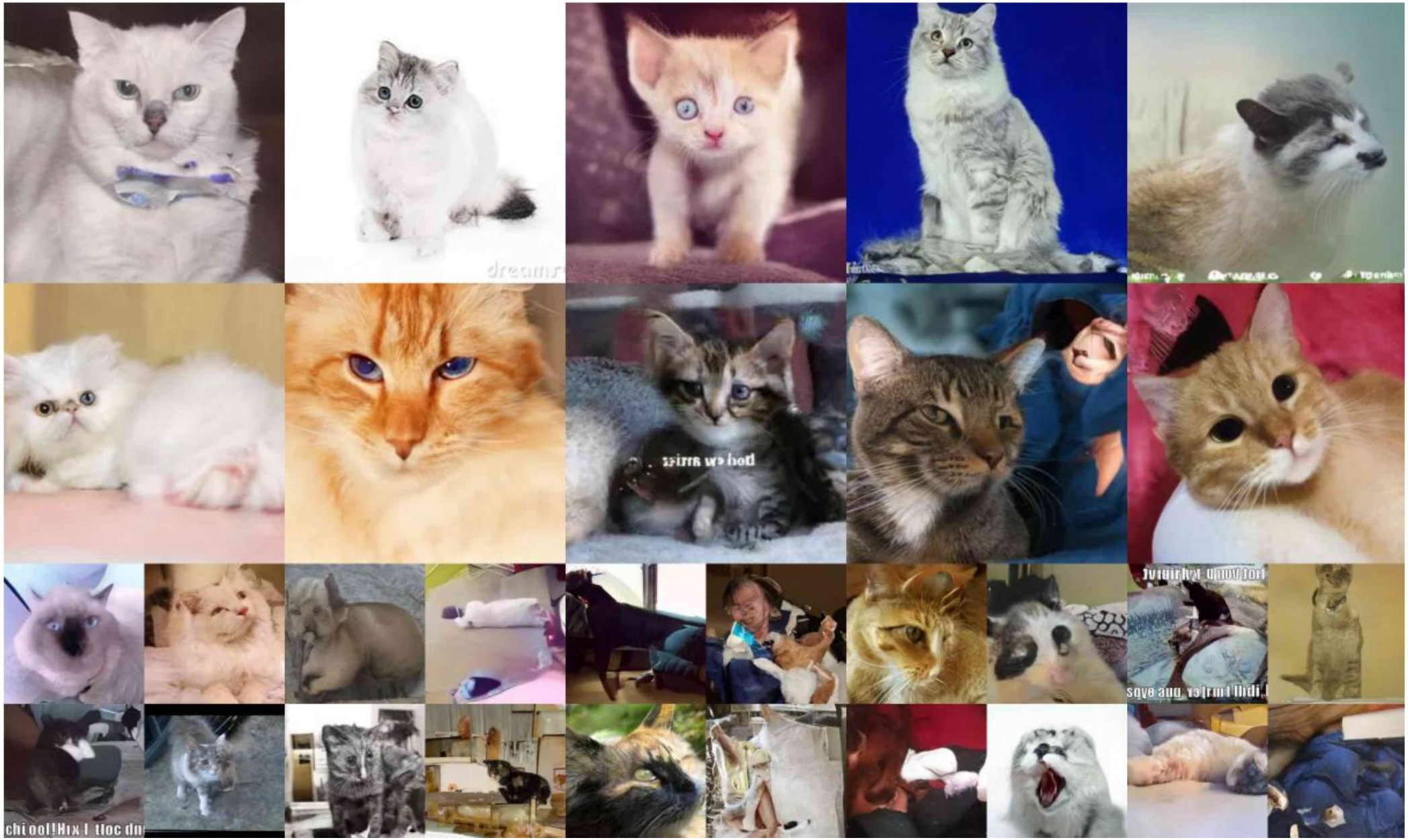
LSUN Bedroom generated samples, small model. FID=6.36



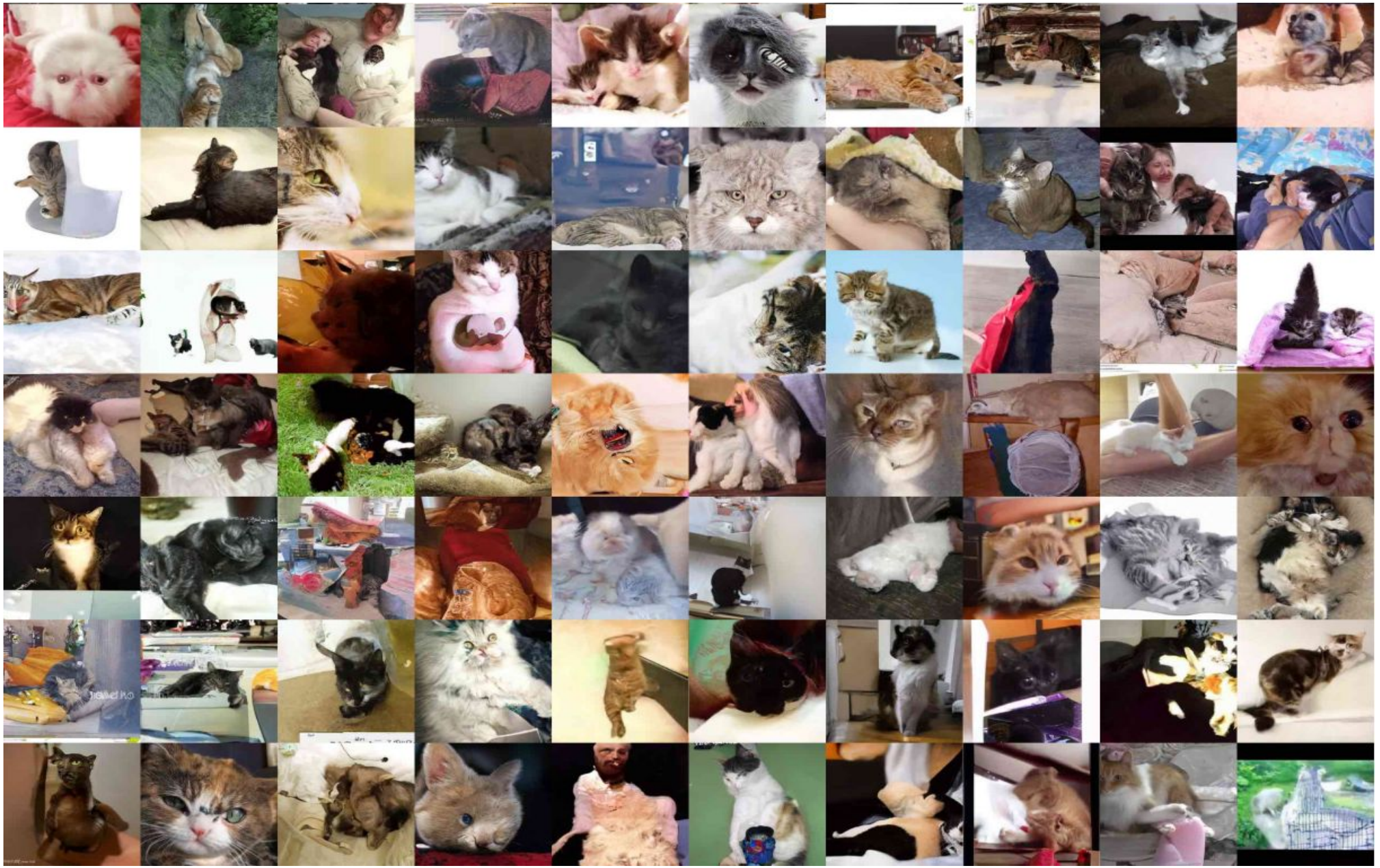
LSUN Bedroom generated samples, small model. FID=6.36



LSUN Cat generated samples. FID=19.75

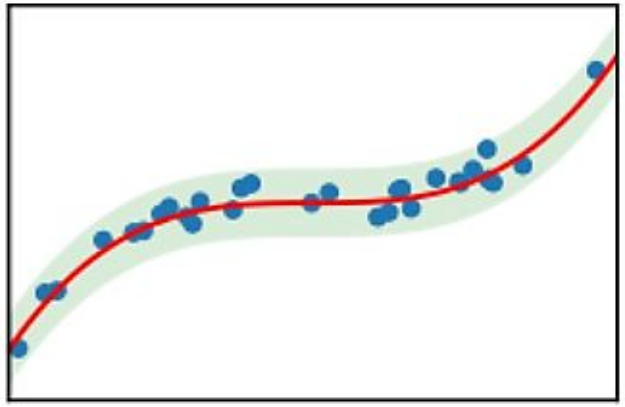


LSUN Cat generated samples. FID=19.75

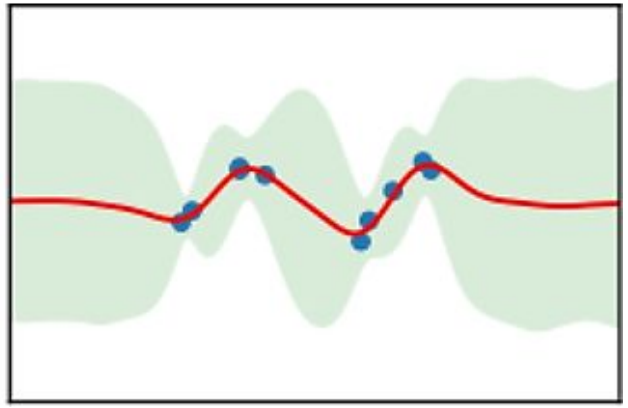


Types of Uncertainty

- There are two major kinds of uncertainty
- **Epistemic** Uncertainty describes what the model doesn't know. It is attributed to inadequate knowledge of the model. This is the uncertainty which can be reduced by having more data or increasing the model complexity.
- **Aleatoric** Uncertainty is the inherent uncertainty which is part of the data generating process. For example, a paper plane which is launched by a high precision equipment, which maintains the same degree of release, speed of release and a thousand other parameters will still not fall in the same place each trial. This inherent variability is Aleatoric Uncertainty.



Aleatoric Uncertainty



Epistemic Uncertainty

Illustration of aleatoric and epistemic uncertainty. Blue dots are the data points, red lines are the predictions, and the green shades is the ±3 Standard deviation around the prediction. Aleatoric uncertainty captures the noise in the dataset and is thus constat in the case if a data set with homoscedastic noise pictured above. Meanwhile, epistemic uncertainty captures the uncertainty of the model and thus decrease when more data points are observed