Diffusion Models Chapter 4: Conditional Generation I

Generative AI and Foundation Models

Spring 2024

Department of Mathematical Sciences

Ernest K. Ryu

Seoul National University

Langevin MCMC (Markov chain Monte Carlo)

Consider

$$p(x) = e^{V(x)}/Z$$

where $V(x)$ is nice and $Z = \int_{\mathbb{R}^d} e^{V(x)}\, dx$ is the normalization constant. Assume $Z < \infty$.

The marginals of

$$dX_t = \frac{1}{2}\nabla \log p(X_t)dt + dW_t = \frac{1}{2}\nabla V(X_t)dt + dW_t$$

converge to the stationary distribution $X_t \xrightarrow{D} p$ as $t \to \infty$.[#]

(If $X_0 \sim p$ then $X_t \sim p$. Follows from directly verifying the Fokker–Planck equation.)

T.-S. Chiang, C.-R. Hwang, and S. J. Sheu, Diffusion for global optimization in $\mathbb{R}^n$, *SIAM Journal on Control and Optimization*, 1987.

Langevin MCMC

In principle, if we train $s_\theta(X) \approx \nabla_X \log p_{\text{data}}(X)$ perfectly, then Langevin MCMC with $s_\theta(X)$ eventually generates samples from p_{data}. Will the following work? (It doesn't.)

Step 1. Train s_θ with loss

$$\mathcal{L}(\theta) = \mathbb{E}_{X \sim p_{\text{data}}} \| s_\theta(X) - \nabla_X \log p_{\text{data}}(X) \|^2$$

using SSM (or DSM).

Step 2. Run Langevin MCMC with small Δt:

$$X_{k+1} = X_k + \frac{\Delta t}{2} s_\theta(X_k) + \sqrt{\Delta t} Z_k$$

Langevin v.1 (doesn't work)

Train score network via sliced score matching (SSM).

while (not converged)

$X \sim p_{\text{data}}$

$\nu \sim p_\nu$ #$\mathbb{E}_{\nu \sim p_\nu}[\nu\nu^\top] = I$

Backprop on h with $\frac{d}{dh}\nu^\intercal s_\theta(X + h\nu)\Big|_{h=0}$

Call optimizer with $\nabla_\theta \left(\|s_\theta(X)\|^2 + 2\frac{d}{dh}\nu^\intercal s_\theta(X + h\nu)\Big|_{h=0} \right)$

end

Output: $s_\theta \approx \nabla \log p_{\text{data}}$?

Problem 1 of Langevin

Manifold hypothesis) If p_{data} does not have full support (data resides on a low-dimensional manifold), then $\log p_{\text{data}}(X) = -\infty$ out of support. The learning $s_\theta(X) \approx \nabla_X \log p_{\text{data}}(X)$ is an ill-posed problem.

Resolution is to perturb data with Gaussian noise with small variance σ^2 and learn the perturbed distribution $\tilde{p}_{\text{data}}$.

(We give up on sampling from p_{data}. Try to sample from $\tilde{p}_{\text{data}}$ instead.)

Langevin SSM v.2 (doesn't work)

while (not converged)

$X \sim p_{\text{data}}$

$Z \sim \mathcal{N}(0, I)$

$\tilde{X} = X + \sigma Z$

$\nu \sim p_\nu$ #$\mathbb{E}_{\nu \sim p_\nu}[\nu\nu^\top] = I$

Backprop on h with $\frac{d}{dh}\nu^\intercal s_\theta(\tilde{X} + h\nu)\Big|_{h=0}$

Call optimizer with $\nabla_\theta \left(\|s_\theta(\tilde{X})\|^2 + 2\frac{d}{dh}\nu^\intercal s_\theta(\tilde{X} + h\nu)\Big|_{h=0} \right)$

end

Output: $s_\theta \approx \nabla \log \tilde{p}_{\text{data}}$?

Langevin DSM v.2 (doesn't work)

We can also use denoising score matching (DSM).

while (not converged)

$X \sim p_{\text{data}}$

$\varepsilon \sim \mathcal{N}(0, I)$

$\tilde{X} = X + \sigma\varepsilon$

Call optimizer with $\nabla_\theta \left\| \varepsilon_\theta(\tilde{X}) - \varepsilon \right\|^2$

end

Output: $s_\theta = -\frac{1}{\sigma}\varepsilon_\theta \approx \nabla \log \tilde{p}_{\text{data}}$?

Here, we use

$$\nabla_\theta \| s_\theta(\tilde{X}) - \nabla_{\tilde{X}} \log p(\tilde{X}|X) \|^2 = \nabla_\theta \left\| s_\theta(\tilde{X}) + \frac{1}{\sigma^2}(\tilde{X} - X) \right\|^2 = \nabla_\theta \left\| s_\theta(\tilde{X}) + \frac{1}{\sigma}\varepsilon \right\|^2 = \frac{1}{\sigma^2}\nabla_\theta \left\| \varepsilon_\theta(\tilde{X}) - \varepsilon \right\|^2$$

Problem 2 of Langevin

Slow mixing of Langevin) The Langevin MCMC sampling

for $k = 0, 1, \ldots, K-1$

$$Z_k \sim \mathcal{N}(0, I)$$

$$X_{k+1} = X_k + \frac{\Delta t}{2} s_\theta(X_k) + \sqrt{\Delta t} Z_k$$

end

takes too long to converge (even though convergence is guaranteed with infinite iterations).

Resolution is to start with large "temperature" $\tau \gg 1$ and gradually reduce it to $\tau = 1$.

Sampling with annealed Langevin

The Langevin MCMC with temperature $\tau \geq 1$

$$X_{k+1} = X_k + \frac{\Delta t}{2\tau} s_\theta(X_k) + \sqrt{\Delta t} Z_k$$

corresponds to the SDE

$$dX_t = \frac{1}{2\tau} \nabla \log p(X_t) dt + dW_t = \frac{1}{2} \nabla \frac{V(X_t)}{\tau} dt + dW_t$$

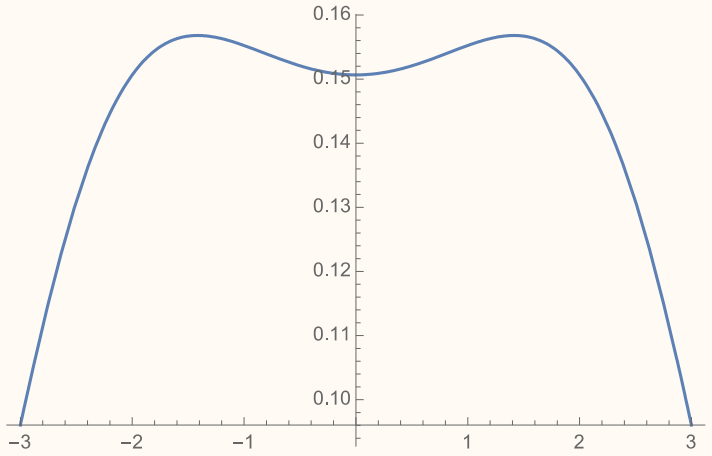
The stationary distribution is

$$X_t \xrightarrow{D} p_\tau = e^{V(x)/\tau} / Z_\tau$$

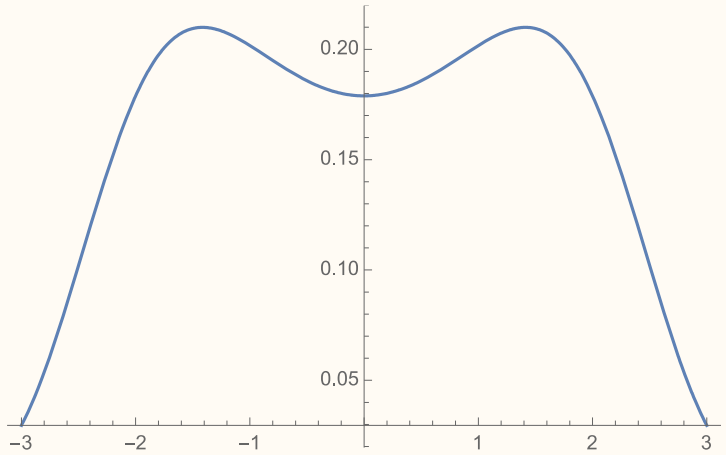
At high temperature, stationary distribution becomes nicer (closer to unimodal), so sampling is easier. At $\tau = 1$, the sampling is harder, but the gradual reduction in τ means p_τ changes gradually, so annealing allows us to eventually sample from p_1.

With $V(x) = -x^2(x-2)(x+2)$, we have $e^{V(x)/\tau}$

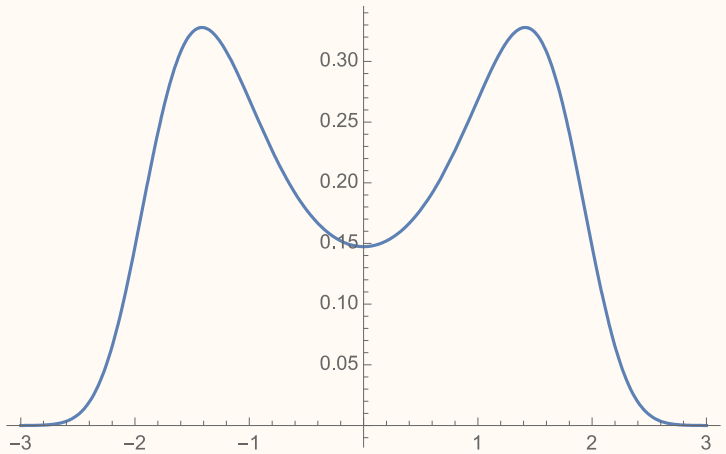
$\tau = 100$



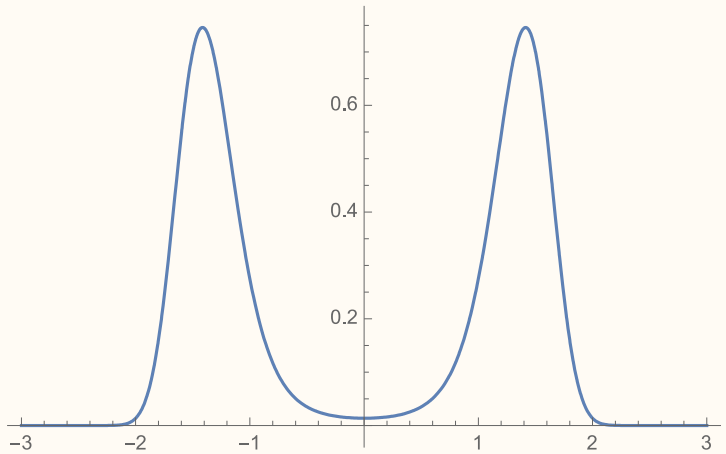
$\tau = 25$



$\tau = 5$



$\tau = 1$



Sampling with annealed Langevin

Initialize X_0

Set decreasing temperature schedule $\tau_1, \tau_2, \ldots \to 1$

for $\ell = 0, 1, \ldots$

  for $k = 0, 1, \ldots, K-1$

    $Z_k \sim \mathcal{N}(0, I)$

    $$X_{k+1} = X_k + \frac{\Delta t}{2\tau_\ell} s_\theta(X_k) + \sqrt{\Delta t} Z_k$$

  end

  # $X_K \sim p_{\tau_\ell}$ approximately

  $X_0 = X_K$

  # decrease temperature $\tau_\ell \to \tau_{\ell+1}$

end

return X_K

Z. Li, Y. Chen, and F. T. Sommer, Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching, *arXiv*, 2019.

Sampling with annealed Langevin

One advantage of this approach is that the learned score network $s_\theta(X)$ is time independent, so it is, in principle, simpler. However, empirical performance is not good.[#]

Annealed Langevin dynamics can work well in Bayesian statistics, where $V(x)$ or the score function $\nabla V(x)$ is precisely known.

In diffusion probabilistic models, however, the score function must be learned.
This is difficult due to problem 3.

[#]Z. Li, Y. Chen, and F. T. Sommer, Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching, *arXiv*, 2019.

Problem 3 of Langevin

Inaccurate score estimation low-density regions.) Training of s_θ with SSM or DSM happens only at datapoints $X \sim p$ or at its perturbations $\tilde{X} = X + \sigma\varepsilon$.

If X' is highly unlikely, then $s_\theta(X')$ will not be accurately trained. However, X' may be an image reachable with high temperature or it may be in a low-probability region that must be traversed for an image in one peak to reach another peak.

Resolution is to add noise to $\sigma \gg 0$ and gradually reduce it $\sigma = 0$, and learn separate score function for all noise levels.

Annealed Langevin dynamics with NCSN

Noise conditioned score network (NCSN) resolves all three problems by considering varying levels of noise perturbations and a score network conditioned on the varying noise levels.

Let $X \sim p_{\text{data}}$ and $\varepsilon \sim \mathcal{N}(0, I)$. Define $\tilde{p}^{\sigma}_{\text{data}}$ via

$$X + \sigma\varepsilon \sim \tilde{p}^{\sigma}_{\text{data}}$$

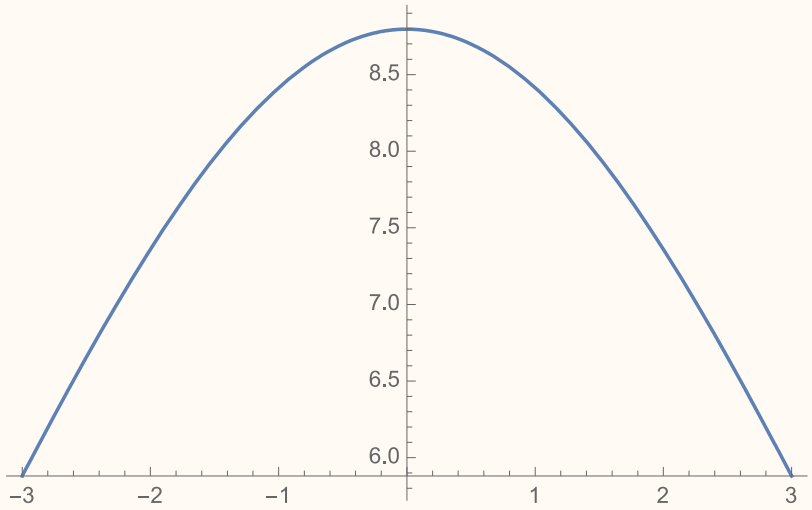
Equivalently,

$$\tilde{p}^{\sigma}_{\text{data}}(x) = \int_{\mathbb{R}^d} \frac{1}{(2\pi\sigma^2)^{d/2}} e^{-\frac{1}{2\sigma^2}\|x-y\|^2} p_{\text{data}}(y)\, dy$$

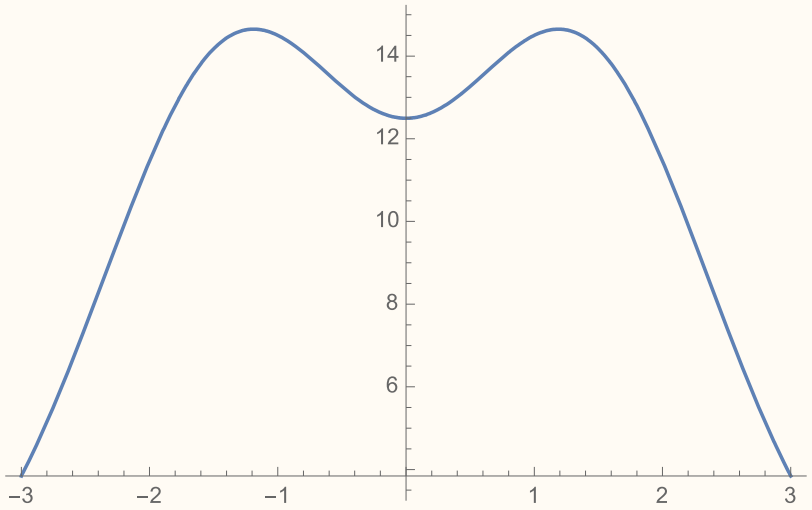
Y. Song and S. Ermon, Generative modeling by estimating gradients of the data distribution, *NeurIPS*, 2019.

With $p(x) \propto e^{-x^2(x-2)(x+2)}$, we have $\tilde{p}^{\sigma}_{\text{data}}$

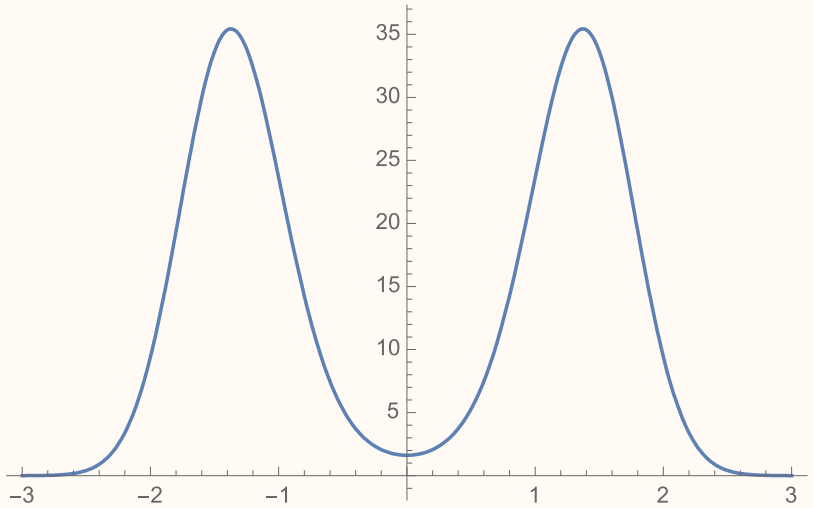
$\sigma = 3$



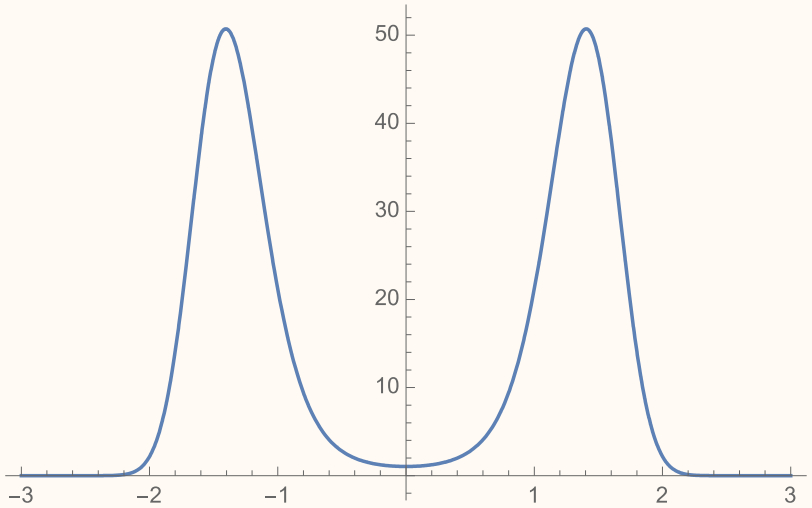
$\sigma = 1$



$\sigma = 0.3$



$\sigma = 0.1$



Training for NCSN

Set noise schedule $\sigma_1, \sigma_2, \ldots, \sigma_L$

while (not converged)

$X \sim p_{\text{data}}$

$i \sim \text{Uniform}(\{1, \ldots, L\})$

$\varepsilon \sim \mathcal{N}(0, I)$

$\tilde{X} = X + \sigma_i \varepsilon$

Call optimizer with $\nabla_\theta \left\| \varepsilon_\theta(\tilde{X}, \sigma_i) - \varepsilon \right\|^2$

end

Output: $s_\theta(x, \sigma_i) = -\frac{1}{\sigma_i} \varepsilon_\theta(x, \sigma_i) \approx \nabla \log \tilde{p}_{\text{data}}^{\sigma_i}$

Y. Song and S. Ermon, Generative modeling by estimating gradients of the data distribution, *NeurIPS*, 2019.

Sampling with NCSC

Initialize X_0

Set decreasing noise schedule $\sigma_1, \sigma_2, \ldots, \sigma_L$

for $\ell = 0, 1, \ldots$

  for $k = 0, 1, \ldots, K-1$

    $Z_k \sim \mathcal{N}(0, I)$

    $$X_{k+1} = X_k + \frac{\Delta t}{2} s_\theta(X_k, \sigma_\ell) + \sqrt{\Delta t} Z_k$$

  end

  # $X_K \sim \tilde{p}^{\sigma_\ell}_{\text{data}}(x)$ approximately

  $X_0 = X_K$

  # decrease noise level $\sigma_\ell \to \sigma_{\ell+1}$

end

return X_K

(The NCSN paper also considers a stepsize schedule $\Delta t_1, \Delta t_2, \ldots, \Delta t_L$.)

Discussion

The authors of NCSN refer to their sampling algorithm as an “an annealed version of Langevin dynamics” despite it not being exactly the same as the classical sense of annealing, which involves the notion of temperature.

If X' is an unlikely data and $\sigma_{\text{big}} \gg \sigma_{\text{small}} \approx 0$, then $s_\theta(X', \sigma_{\text{big}})$ may be trained well, but $s_\theta(X', \sigma_{\text{small}})$ will not be trained well. This is okay because:

- $s_\theta(X', \sigma_{\text{small}})$ is not used often in sampling.
- The conditional U-Net architecture encourages $s_\theta(X', \sigma_{\text{small}}) \approx s_\theta(X', \sigma_{\text{big}})$ when there is training data for $(X', \sigma_{\text{big}})$ but not for $(X', \sigma_{\text{small}})$. If $s_\theta(X', \sigma_{\text{small}})$ is used in a small-probability event, then $s_\theta(X', \sigma_{\text{small}}) \approx s_\theta(X', \sigma_{\text{big}})$ is a reasonable direction.

For time-dependent score networks $s_\theta(X, t)$ for diffusion SDEs and DDPM, using a single time-conditioned U-net allows the training at one time-step to influence other adjacent timesteps. This is good as it $s_\theta(X', t)$ to have reasonable directions for small t (i.e., low noise) even for unlikely data X'.

NCSN = discretization of VE SDE

Consider the general VE forward-time SDE

$$dX_t = \sqrt{\frac{d(\sigma_t^2)}{dt}} dW_t$$

With $\Delta t = 1$, the Euler–Maruyama discretization is

$$X_{t+1} = X_t + \sqrt{\sigma_{t+1}^2 - \sigma_t^2} Z_t \overset{\mathcal{D}}{=} X_0 + \sigma_{t+1} Z_t$$

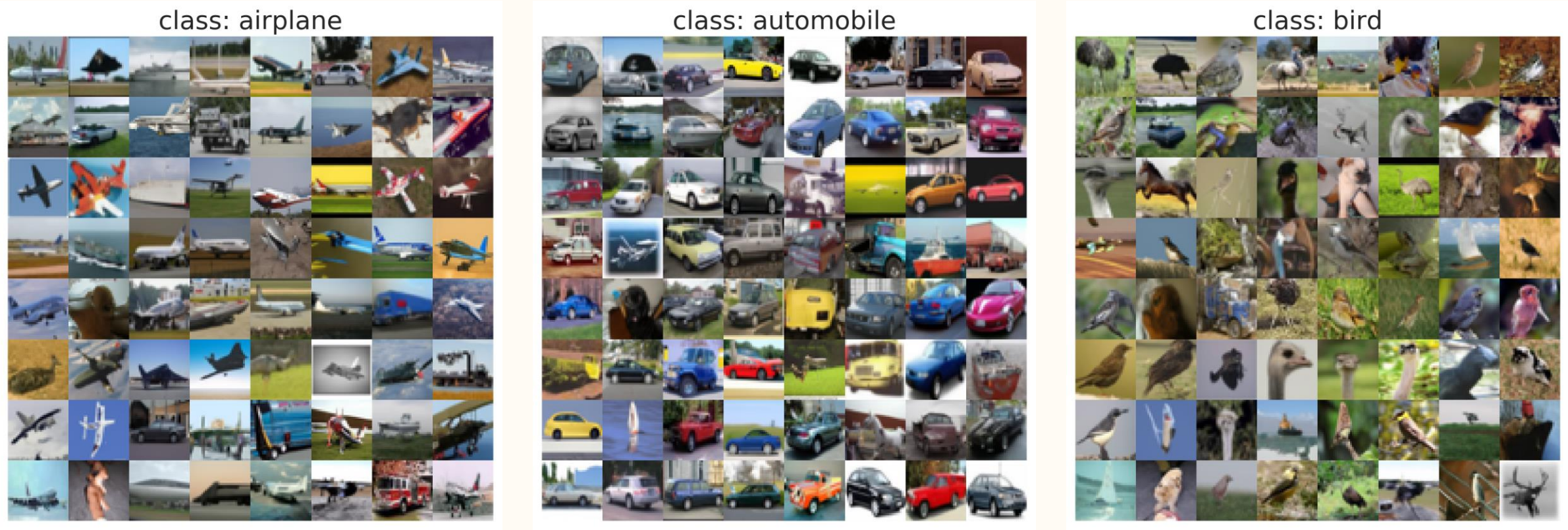
corresponds to the forward noising process of NCSN, which learns the score functions

$$\tilde{X} = X + \sigma_i Z \sim \tilde{p}_{\text{data}}^{\sigma_i}$$

for a noise schedule $\sigma_1, \sigma_2, \ldots, \sigma_L$. There is no direct correspondence with sampling.

Conditional generation

Assume we have access to $(X, Y) \sim p_{\text{data}}^{X,Y}$. The goal of conditional generation is to sample from $X \sim p_{\text{data}}^{X|Y}(\cdot\,|Y)$. (Here, the Y may be "labels" but it may also be other auxiliary information as we shall see later.)



Conditional forward-time SDE

Consider a setup where (X, Y) are randomly generated and we applying a forward-time SDE to X. So,

$$dX_t = f(X_t, t)dt + g(t)dW_t, \qquad X_0 = X, \qquad (X, Y) \sim p_{\text{data}}^{X,Y}$$

which is, of course, equivalent to

$$dX_t = f(X_t, t)dt + g(t)dW_t, \qquad X_0 \sim p_{\text{data}}^{X|Y}(\cdot \mid Y), \qquad Y \sim p_{\text{data}}^{Y}$$

In particular, the transition $X_0 \mapsto X_t$ is independent of Y conditioned on X_0:

$$p(X_t \mid X_0, Y) = p(X_t \mid X_0)$$

Conditional reverse-time SDE

Given Y, the forward-time SDE generates X_t conditioned on Y and has $X_t \sim p_t(\cdot\,|Y)$:

$$dX_t = f\,dt + g\,dW_t, \qquad X_0 \sim q_0 = p_{\text{data}}^{X|Y}(\cdot\,|\,Y)$$

The reverse-time SDE generates $\bar{X}_0 \sim p_0(\cdot\,|Y) = p_{\text{data}}^{X|Y}(\cdot\,|Y)$:

$$d\overline{X}_t = (f - g^2 \nabla_{\overline{X}_t} \log p_t(\overline{X}_t\,|\,Y))dt + g\,d\bar{W}_t, \qquad \overline{X}_T \sim p_T(\cdot\,|\,Y) \approx \mathcal{N}(0, \sigma_T^2 I)$$

Can generate from $p_0(\cdot\,|\,Y) = p_{\text{data}}^{X|Y}(\cdot\,|\,Y)$ by learning conditional score $\nabla_{X_t} \log p_t(X_t\,|\,Y)$.

By Bayes's rule,

$$p_t(X_t\,|\,Y) = \frac{p_t(X_t)p_t(Y\,|\,X_t)}{p(Y)} \qquad \text{and} \qquad \nabla_{X_t} \log p_t(X_t\,|\,Y) = \nabla_{X_t} \log p_t(X_t) + \nabla_{X_t} \log p_t(Y\,|\,X_t)$$

Base classifier guidance

1. Train regular score function $s_\theta(X_t, t) \approx \nabla \log p_t(X_t)$ by disregarding the label Y:

$$dX_t = f(X_t, t)dt + g(t)dW_t, \qquad X_0 \sim p_{\text{data}}^X$$

while (not converged)

$(X, Y) \sim p_{\text{data}}^{X,Y}$

$X_0 = X$ `# discard` Y

$t \sim \text{Uniform}([0, T])$

$\varepsilon \sim \mathcal{N}(0, I)$

$X_t = \gamma_t X_0 + \sigma_t \varepsilon$

Call optimizer with $\frac{\lambda(t)}{\sigma_t^2} \nabla_\theta \|\varepsilon_\theta(X_t, t) - \varepsilon\|^2$

end

Base classifier guidance

2. Train a *time-dependent classifier* $c_\phi(X_t, Y, t) \approx p_t(Y|X_t)$ that predicts the label Y given the corrupted data X_t. The classifier c_ϕ is like a usual classifier in supervised learning, but it takes in an additional time parameter t.

Let $Y \in \{1, \dots, K\}$. Let ℓ^{CE} be the cross-entropy loss.

Let $\tilde{c}_\phi(X_t, t) \in \mathbb{R}^K$ be the logits of the time-dependent classifier, i.e., $c_\phi(X_t, Y, t) = (\text{softmax}(\tilde{c}_\phi(X_t, t)))_Y$.

while (not converged)

$(X, Y) \sim p_{\text{data}}^{X,Y}$

$X_0 = X$

$t \sim \text{Uniform}([0, T])$

$\varepsilon \sim \mathcal{N}(0, I)$

$X_t = \gamma_t X_0 + \sigma_t \varepsilon$

Call optimizer with $\nabla_\phi \ell^{\text{CE}}(\tilde{c}_\phi(X_t, t), Y)$

end

Base classifier guidance

3. Use $s_\theta(X_t, t)$ and $c_\phi(X_t, Y, t)$ to approximate the reverse-time conditional SDE

$$d\overline{X}_t = (f - g^2(\nabla_{\overline{X}_t} \log p_t(\overline{X}_t) + \nabla_{\overline{X}_t} \log p_t(Y \mid \overline{X}_t)))dt + g d\bar{W}_t, \qquad \overline{X}_T \sim p_T$$

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, *ICML*, *2015*.
Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, Score-based generative modeling through stochastic differential equations, *ICLR*, 2021.

Scaled classifier guidance

Experimentally, it helps to scale the classifier gradients by a constant factor ω larger than 1. When using a scale of $\omega = 1$, it can happen that the classifier assigned reasonable probabilities (around 50%) to the desired classes for the final samples, but these samples did not match the intended classes upon visual inspection. Scaling up the classifier gradients remedies this problem, and the class probabilities from the classifier increases to nearly 100%.

In other words, we follow the SDE

$$d\overline{X}_t = (f - g^2(\nabla_{\overline{X}_t} \log p_t(\overline{X}_t) + \omega \nabla_{\overline{X}_t} \log p_t(Y \mid \overline{X}_t)))dt + g d\bar{W}_t, \qquad \overline{X}_T \sim p_T$$

with $\omega > 1$.

P. Dhariwal and A. Nichol, Diffusion models beat GANs on image synthesis, *NeurIPS*, 2021.

Scaled classifier guidance

The parameter ω trades off diversity for fidelity. Consider $Y =$“Pembroke Welsh corgi”. For $\omega = 1.0$ (left), images do not all fit the designated class. For $\omega = 10.0$ (right), images are much more class-consistent.

P. Dhariwal and A. Nichol, Diffusion models beat GANs on image synthesis, *NeurIPS*, 2021.

Classifier-free guidance

The problem with classifier guidance is that we must train a separate classifier. (We need two neural networks.) *Classifier-free guidance* relies on

$$p(y \mid x) \propto \frac{p(x \mid y)}{p(x)}$$

to get

$$d\overline{X}_t = (f - g^2((1-\omega)\nabla_{X_t} \log p_t(\overline{X}_t) + \omega \nabla_{X_t} \log p_t(\overline{X}_t \mid Y)))dt + g d\bar{W}_t, \qquad \overline{X}_T \sim p_T$$

with a classifier scale ω.

Instead of a separate classifier, we train *one* score network to represent both the unconditional score $\nabla_{X_t} \log p_t(X_t)$ and conditional score $\nabla_{X_t} \log p_t(X_t|Y)$.

J. Ho and T. Salimans, Classifier-free diffusion guidance, *NeurIPS 2021 Workshop on Deep Generative Models*, 2021.

Training for classifier-free guidance

Choose a forward SDE which defines γ_t and σ_t.

while (not converged)

$(X, Y) \sim p_{\text{data}}^{X,Y}$

$Y = \emptyset$ with probability $p_{\text{uncond}} \approx 20\%$

$X_0 = X$

$t \sim \text{Uniform}([0, T])$

$\varepsilon \sim \mathcal{N}(0, I)$

$X_t = \gamma_t X_0 + \sigma_t \varepsilon$

Call optimizer with $\frac{\lambda(t)}{\sigma_t^2} \nabla_\theta \|\varepsilon_\theta(X_t, Y, t) - \varepsilon\|^2$

end

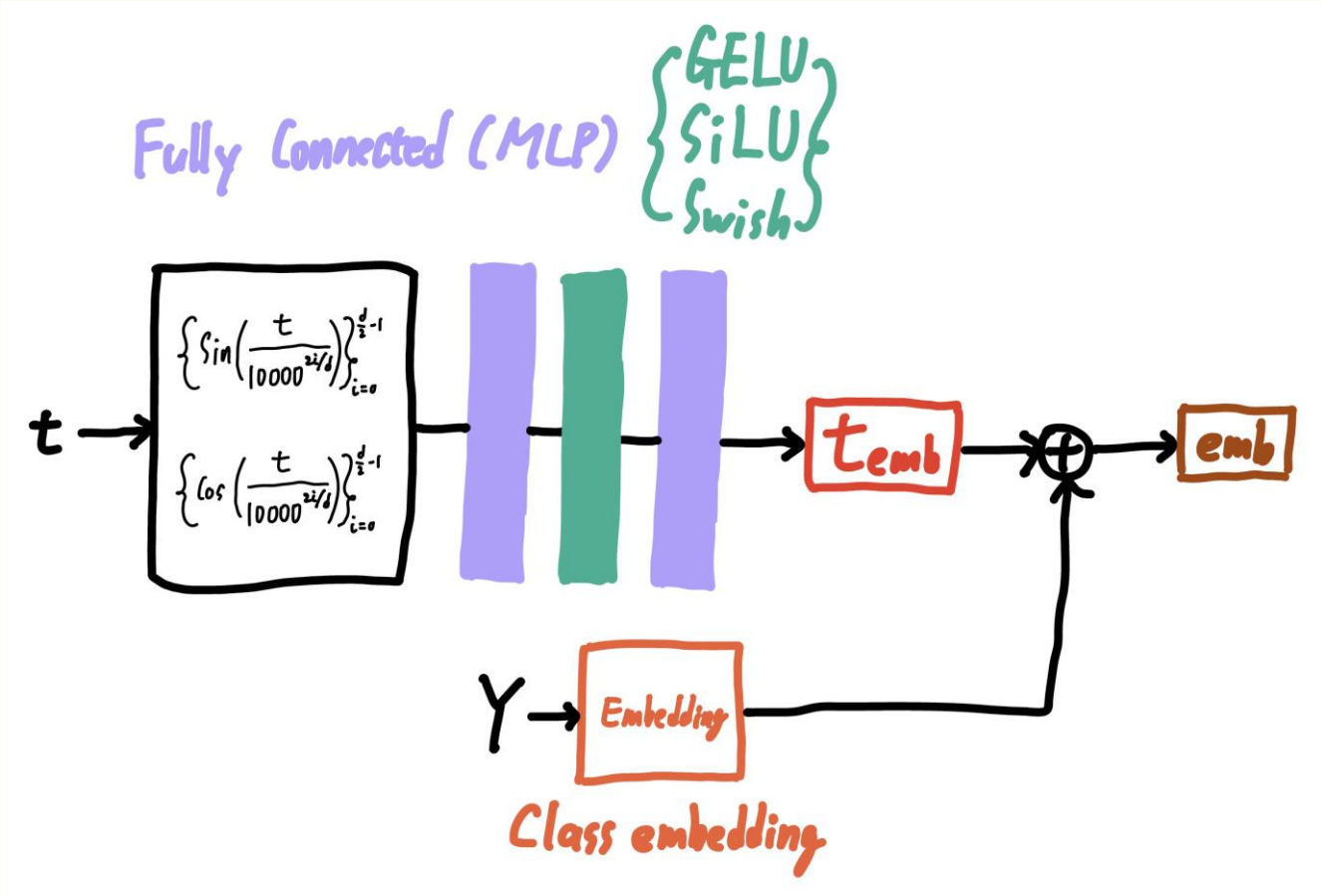
Sampling for classifier-free guidance

Since $\nabla_{X_t} \log p_t(X_t) = \nabla_{X_t} \log p_t(X_t|Y = \emptyset)$, generate samples with

$$d\overline{X}_t = (f - g^2((1-\omega)s_\theta(\bar{X}_t, \emptyset, t) + \omega s_\theta(\bar{X}_t, Y, t)))dt + g d\bar{W}_t, \qquad \overline{X}_T \sim p_T$$

Class embedding

Class embeddings are injected into U-Net architecture as an embedding layer added to the time embedding. In PyTorch, an `nn.Embedding` layer has a trainable output vector for each inedex/label.



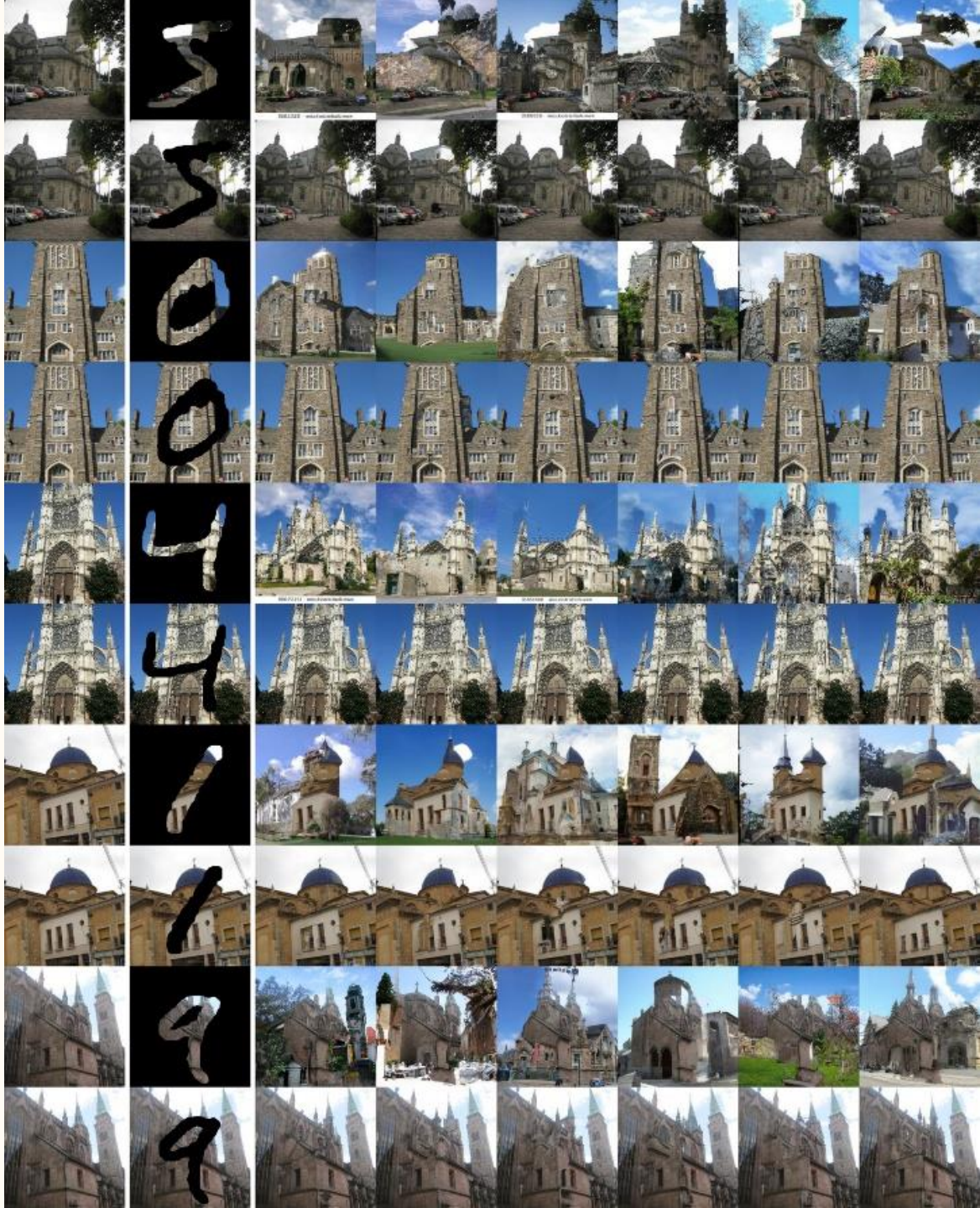
Inpainting

In image inpainting, we reconstruct parts of the image based on partial measurement.

Assume we have a score network $s_\theta(X,t) \approx \nabla_X \log p_t(X)$ trained for

$$dX_t = f(X_t,t)dt + g(t)dW_t, \qquad X_0 \sim p_0$$

We do not re-train a different score network for image inpainting.



Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, Score-based generative modeling through stochastic differential equations, *ICLR*, 2021.

Inpainting

Assume f and g are defined elementwise. More specifically, assume

$$dX_t = f(X_t, t)dt + g(t)dW_t, \qquad X_0 \sim p_0$$

can be decomposed into

$$(dX_t)_i = f_i((X_t)_i, t)dt + g_i(t)(dW_t)_i, \qquad \forall\, i$$

Under this assumption, forward-time corruption is independent across pixels and channels. (Reverse-time generation is not independent across pixels and channels, because the score function $\nabla_X \log p_t(X)$ does not split elementwise.)

The forward-time SDEs we have been considering fall under this category.

$$dX_t = -\beta(t)X_t dt + \sigma(t)dW_t, \qquad X_0 \sim p_0$$

Inpainting

Let $X_0 \in \mathbb{R}^{C\times W\times H}$ be the true, partially observed image. Let $\Omega : \mathbb{R}^{C\times W\times H} \to \mathbb{R}^K$, where $K \leq CWH$, be a pixel/channel subsampling operation. Our observed data is $Y = \Omega(X_0)$. Let $X_t^\Omega = \Omega(X_t)$.

Let Ω^C be a complementary pixel subsampling operation, i.e., Ω^C selects the other pixels not selected by Ω. Let $X_t^C = \Omega^C(X_t)$. Then X_t^C follows basically the same forward-time SDE

$$dX_t^C = f^C(X_t^C, t)dt + g^C(t)dW_t^C, \qquad X_0^C = \Omega^C(X_0), \qquad X_0 \sim p_0$$

where $f^C(Z_t, t) = \Omega^C\big(f(X_t, t)\big)$, $g^C(t) = \Omega^C\big(g(t)\big)$, and $W_t^C = \Omega^C(W_t)$.

Since the Brownian motion W_t is coordinate-wise independent, W_t^C is simply a lower-dimensional Brownian motion.

Goal is to sample X_0^C given $Y = \Omega(X_0)$.

We have $$p_t(X_t^C \mid X_0^\Omega = Y) = \int p_t(X_t^C \mid X_t^\Omega, X_0^\Omega = Y) p_t(X_t^\Omega \mid X_0^\Omega = Y)\, dX_t^\Omega$$
$$= \mathbb{E}_{X_t^\Omega \sim p_t(X_t^\Omega \mid X_0^\Omega = Y)}[p_t(X_t^C \mid X_t^\Omega, X_0^\Omega = Y)]$$
$$\approx \mathbb{E}_{X_t^\Omega \sim p_t(X_t^\Omega \mid X_0^\Omega = Y)}[p_t(X_t^C \mid X_t^\Omega)]$$
$$\approx p_t(X_t^C \mid \hat{X}_t^\Omega)$$

where $\hat{X}_t^\Omega$ is a forward noise sample given $Y = X_0^\Omega$.

First approximation is based on argument that $X_0^\Omega = Y$ does not additional provide much information about X_t^C given X_t^Ω. For small t, $X_t^\Omega \approx X_0^\Omega = Y$ so the approximation holds. For large t, $X_0^\Omega = Y$ becomes further away from X_t^C in the Markov process, and thus have smaller impact on X_t^C. Moreover, the approximation error for large t matter less for the final sample, since it is used early in the sampling process.

I do not find this argument fully convincing, and there are more sophisticated[#] diffusion-based inpainting techniques. However, this approach serves as a nice and simple baseline.

[#]When trained directly on the inpainting task, diffusion models can smoothly inpaint regions of an image without edge artifacts:
C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, Palette: Image-to-image diffusion models, *SIGGRAPH*, 2022.

Inpainting

$$
\begin{aligned}
p_t(X_t^C \mid X_0^\Omega = Y) &= \int p_t(X_t^C \mid X_t^\Omega, X_0^\Omega = Y) p_t(X_t^\Omega \mid X_0^\Omega = Y)\, dX_t^\Omega \\
&= \mathbb{E}_{X_t^\Omega \sim p_t(X_t^\Omega \mid X_0^\Omega = Y)} [p_t(X_t^C \mid X_t^\Omega, X_0^\Omega = Y)] \\
&\approx \mathbb{E}_{X_t^\Omega \sim p_t(X_t^\Omega \mid X_0^\Omega = Y)} [p_t(X_t^C \mid X_t^\Omega)] \\
&\approx p_t(X_t^C \mid \hat{X}_t^\Omega)
\end{aligned}
$$

The second approximation is a single-sample estimate of the expected value.
So $\hat{X}_t^\Omega$ is the forward-time corruption from $X_0^\Omega = Y$.

Finally, we perform the SDE sampling via

$$
\begin{aligned}
\nabla_{X_t^C} \log p_t(X_t^C \mid X_0^\Omega = Y) &\approx \nabla_{X_t^C} \log p_t(X_t^C \mid \hat{X}_t^\Omega) \\
&= \nabla_{X_t^C} \log p_t([X_t^C; \hat{X}_t^\Omega]) \\
&\approx \Omega^C \left(s_\theta([X_t^C; \hat{X}_t^\Omega], t) \right)
\end{aligned}
$$

SDE inpainting sampling

We generate $\bar{X}_k^C$ in reverse-time while forward-corrupting $Y = X_0^\Omega$ to get $\hat{X}_t^\Omega$. The score function for $\bar{X}_k^C$ is obtained by evaluating the score network s_θ with input $\left[\bar{X}_k^C; \hat{X}_t^\Omega\right]$.

$\bar{X}_K^C \sim \mathcal{N}(0, \sigma_T^2 I)$
for $k = K, K-1, \ldots, 2, 1$
  $\varepsilon_k \sim \mathcal{N}(0, I)$
  $\hat{X}_k^\Omega = \beta_k Y + \gamma_k \varepsilon_k$ # forward-corrupt Y to time k
  $Z_k \sim \mathcal{N}(0, I)$
  $\bar{X}_{k-1}^C = \bar{X}_k^C - \Delta t(-\beta \bar{X}_k^C - \sigma^2 \Omega^C(s_\theta([\bar{X}_k^C; \hat{X}_k^\Omega], k\Delta t))) + \sigma\sqrt{\Delta t} Z_k$
end

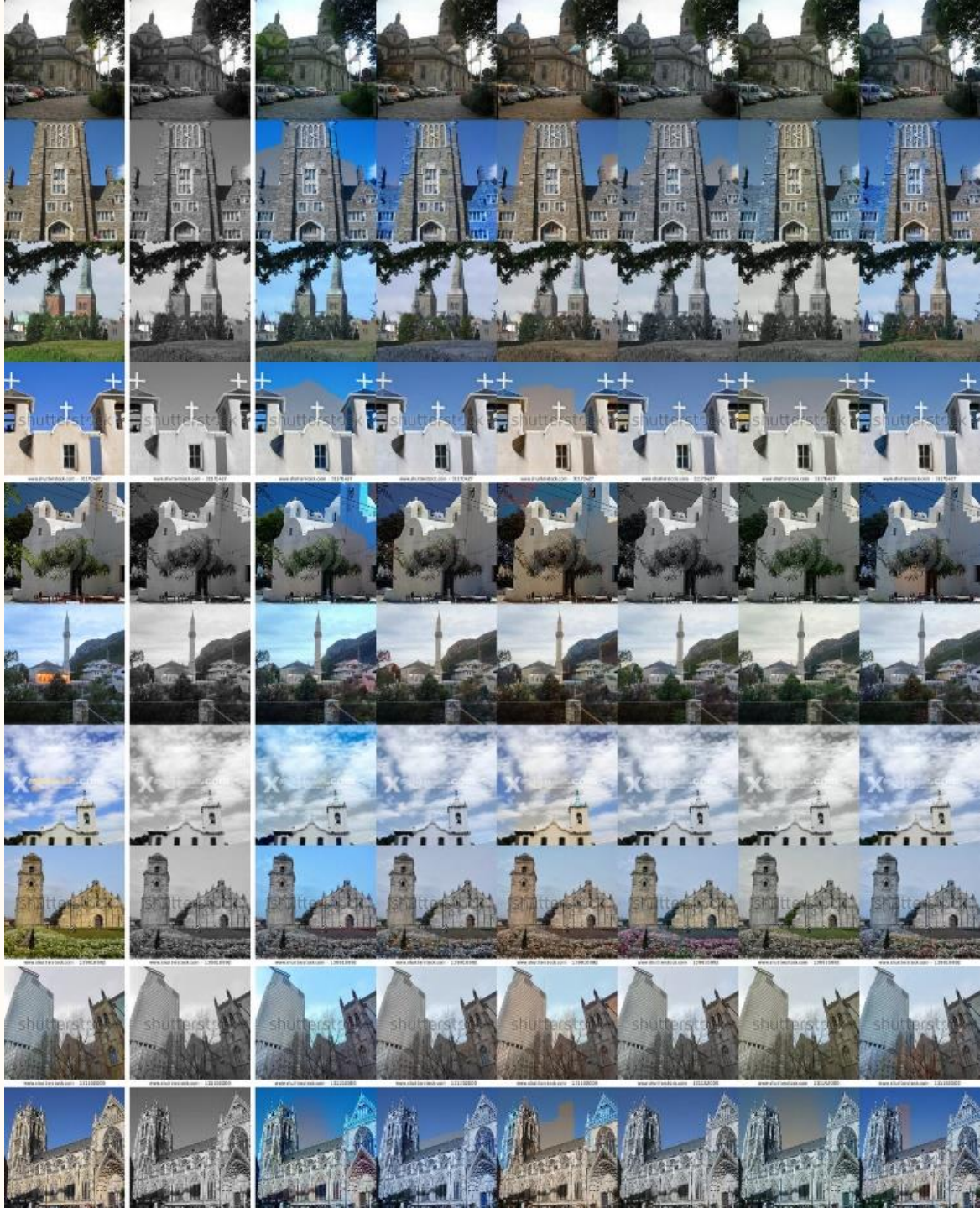
Image colorization

In image colorization, we reconstruct the colors based on a greyscale image.

The colorization is an instance of inpainting after an orthogonal change of coordinates. Use the orthogonal matrix

$$U = \begin{bmatrix} 0.577 & -0.816 & 0 \\ 0.577 & 0.408 & 0.707 \\ 0.577 & 0.408 & -0.707 \end{bmatrix}$$

to map $[I; A; B] \mapsto [R; G; B]$.



Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, Score-based generative modeling through stochastic differential equations, *ICLR*, 2021.

Image colorization

Consider the forward-time SDE

$$dX_t = -\beta(t)X_t dt + \sigma(t)dW_t, \qquad X_0 \sim p_0$$

Let $\{p_t\}_{t\in[0,T]}$ be the marginal densities of $\{X_t\}_{t\in[0,T]}$. Let $s_\theta(X_t, t) \approx \nabla_{X_t} \log p_t(X_t)$ be a trained score network. Let

$$Y_t = \mathbf{U}^T X_t, \qquad \forall\, t \in [0, T],$$

where $\mathbf{U}$ is an orthogonal matrix. Let $\{q_t\}_{t\in[0,T]}$ be the marginal densities of $\{Y_t\}_{t\in[0,T]}$. By the change of variables formula, (since orthogonal matrices have unit Jacobian)

$$q_t(Y_t) = p_t(\mathbf{U}Y_t), \qquad p_t(X_t) = q_t(\mathbf{U}^T X_t), \qquad \forall\, t \in [0, T].$$

So $\mathbf{U}^T s_\theta(\mathbf{U}Y_t, t)$ approximates the score function for Y_t.

Image colorization

The forward-time SDE of Y_t is

$$dMX_t = -\beta(t)MX_t dt + \sigma(t)MdW_t, \qquad X_0 \sim p_0$$

Using the fact that $UW_t \overset{\mathcal{D}}{=} W_t$, we can express it equivalently as

$$dY_t = -\beta(t)Y_t dt + \sigma(t)dW_t, \qquad Y_0 = U^T X_0, \qquad X_0 \sim p_0$$

The corresponding reverse-time SDE is

$$d\overline{Y}_t = (-\beta(t)\overline{Y}_t - \sigma(t)^2 \nabla_{\overline{Y}_t} \log p_t(U\overline{Y}_t))dt + \sigma(t)dW_t, \qquad \overline{Y}_T = U^T \overline{X}_T, \qquad \overline{X}_T \sim p_T$$

and it is approximated by

$$d\overline{Y}_t = (-\beta(t)\overline{Y}_t - \sigma(t)^2 U^T s_\theta(U\overline{Y}_t, t))dt + \sigma(t)dW_t, \qquad \overline{Y}_T \sim \mathcal{N}(0, \sigma_T^2 I).$$

IAB channels. I observed

$X^I = Y$ # greyscale image $Y \in \mathbb{R}^{w \times h}$

$\bar{X}_K^A \sim \mathcal{N}(0, \sigma_T^2 I), \quad \bar{X}_K^B \sim \mathcal{N}(0, \sigma_T^2 I)$

for $k = K, K-1, \ldots, 2, 1$

$\varepsilon_k \sim \mathcal{N}(0, I)$

$\hat{X}_k^I = \beta_k X^I + \gamma_k \varepsilon_k$ # forward-corrupt I channel to time k

$Z_k \sim \mathcal{N}(0, I)$

$\bar{X}_{k-1}^A = \bar{X}_k^A - \Delta t(-\beta \bar{X}_k^A - \sigma^2 \mathbf{U}^{T,A}(s_\theta(\mathbf{U}[\hat{X}_k^I, \bar{X}_k^A, \bar{X}_k^B], k\Delta t))) + \sigma\sqrt{\Delta t} Z_k$

$\bar{X}_{k-1}^B = \bar{X}_k^B - \Delta t(-\beta \bar{X}_k^B - \sigma^2 \mathbf{U}^{T,B}(s_\theta(\mathbf{U}[\hat{X}_k^I, \bar{X}_k^A, \bar{X}_k^B], k\Delta t))) + \sigma\sqrt{\Delta t} Z_k$

end

return $\mathbf{U}[X^I, \bar{X}_0^A, \bar{X}_0^B]$

$\mathbf{U} = U \otimes I \otimes I$ transforms [I,A,B] $\mapsto$ [R,G,B] for each pixel

$\mathbf{U}^{T,A} = U_{2,:}^T \otimes I \otimes I$ transforms [R,G,B] $\mapsto$ [I,A,B]

and extracts A channel for each pixel

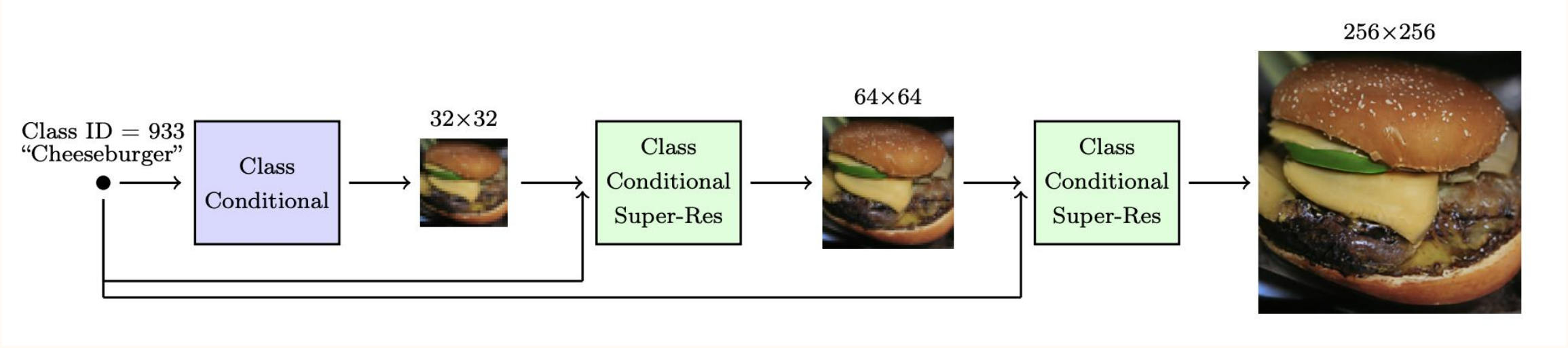
$\mathbf{U}^{T,B} = U_{3,:}^T \otimes I \otimes I$

Cascaded diffusion model (CDM)

Generating a high-resolution image directly in a single step via diffusion is expensive.

Cascaded diffusion models reduce the computational cost by:

- Generate a low-resolution image by a diffusion model.
- Generate higher resolution image by diffusion conditioned on the lower resolution image.

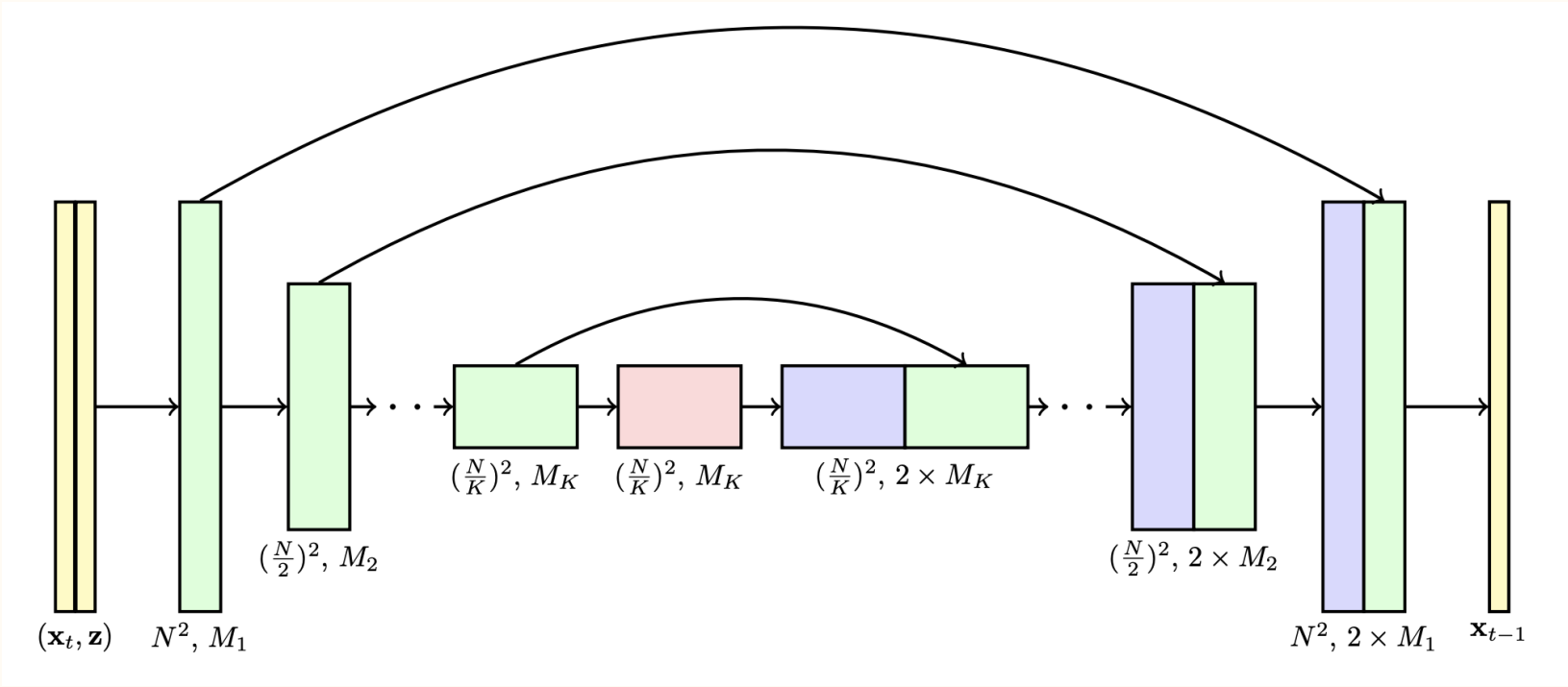


J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, Cascaded diffusion models for high fidelity image generation, *JMLR*, 2022.

CDM architecture

The first network in CDM is identical to that of a standard diffusion model.

The second and subsequent networks use U-Nets for conditional score functions with intput being the concatenation of the previous image $\bar{X}_t$ and an “upsampled version” of the low-resolution image Z.



Class label C and timestep t are injected into each block as an embedding, not depicted here.

Compounding error problem

In super-resolution-based generative models, the compounding error problems is

1. Generative model generates a low-resolution image with some “error”, a certain unrealistic and undesirable feature in the image.
2. The super-resolution routine magnifies (compounds) this error.

CDMs mitigate the compounding error problem by *conditioning augmentation*.

Conditioning augmentation: Training

The super-resolution model takes in a noise-corrupted Z_s and the degree of corruption s.

The error (score) network ε_θ is conditioned on Z_s, not Z_0.

Instead of conditioning on the clean upsampled image Z_0, we are using the noise-corrupted (augmented) version Z_s for conditioning. Hence, the name conditioning augmentation.

Algorithm 1 Training a two-stage CDM with Gaussian conditioning augmentation

repeat ▷ Train base model
  $(\mathbf{z}_0, \mathbf{c}) \sim p(\mathbf{z}, \mathbf{c})$ ▷ Sample low-resolution image and label
  $t \sim \mathcal{U}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}\right\|^2$ ▷ Simple loss (can be replaced with a hybrid loss)
until converged
repeat ▷ Train super-resolution model (in parallel with the base model)
  $(\mathbf{x}_0, \mathbf{z}_0, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{z}, \mathbf{c})$ ▷ Sample low- and high-resolution images and label
  $s, t \sim \mathcal{U}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon}_\mathbf{z}, \boldsymbol{\epsilon}_\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Note: $\boldsymbol{\epsilon}_\mathbf{z}, \boldsymbol{\epsilon}_\mathbf{x}$ should have the same shapes as $\mathbf{z}_0, \mathbf{x}_0$, respectively
  Z_s ~~$\mathbf{z}_t$~~ $= \sqrt{\bar{\alpha}_s}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_s}\boldsymbol{\epsilon}_\mathbf{z}$ ▷ Apply Gaussian conditioning augmentation
  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\mathbf{x}$
  $\theta \leftarrow \theta - \eta\nabla_\theta \left\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{z}_s, s, \mathbf{c}) - \boldsymbol{\epsilon}_\mathbf{x}\right\|^2$
until converged

CDM sampling

The sampling requires choosing s as a hyper parameter.

The sampling

$$X_{t-1} \sim p_\theta(\cdot\,|X_t, X_s, C)$$

depends on t and s. (The error (score) network ε_θ takes t and s in as input and uses them through time embeddings.)

Algorithm 2 Sampling from a two-stage CDM with Gaussian conditioning augmentation

Require: $\mathbf{c}$: class label
Require: s: conditioning augmentation truncation time
$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c})$
end for
$\mathbf{z}_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s, \mathbf{c})$
end for
return $\mathbf{x}_0$