A PROBABILITY FLOW ODE IN TERMS OF LOG-SNR

Song et al. (2021c) formulate the forward diffusion process in terms of an SDE of the form

$$d\mathbf{z} = f(\mathbf{z}, t)dt + g(t)dW, \tag{10}$$

and show that samples from this diffusion process can be generated by solving the associated *probability flow* ODE:

$$d\mathbf{z} = [f(\mathbf{z}, t) - \frac{1}{2}g^2(t)\nabla_z \log p_t(\mathbf{z})]dt, \tag{11}$$

where in practice $\nabla_z \log p_t(\mathbf{z})$ is approximated by a learned denoising model using

$$\nabla_z \log p_t(\mathbf{z}) \approx \frac{\alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t) - \mathbf{z}_t}{\sigma_t^2}. \tag{12}$$

Following Kingma et al. (2021) we have $f(\mathbf{z}, t) = \frac{d \log \alpha_t}{dt}\mathbf{z}_t$ and $g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{d \log \alpha_t}{dt}\sigma_t^2$. Assuming a variance preserving diffusion process with $\alpha_t^2 = 1 - \sigma_t^2 = \text{sigmoid}(\lambda_t)$ for $\lambda_t = \log[\alpha_t^2/\sigma_t^2]$ (without loss of generality, see Kingma et al. (2021)), we get

$$f(\mathbf{z}, t) = \frac{d \log \alpha_t}{dt}\mathbf{z}_t = \frac{1}{2}\frac{d \log \alpha_\lambda^2}{d\lambda}\frac{d\lambda}{dt}\mathbf{z}_t = \frac{1}{2}(1 - \alpha_t^2)\frac{d\lambda}{dt}\mathbf{z}_t = \frac{1}{2}\sigma_t^2\frac{d\lambda}{dt}\mathbf{z}_t. \tag{13}$$

Similarly, we get

$$g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{d \log \alpha_t}{dt}\sigma_t^2 = \frac{d\sigma_\lambda^2}{d\lambda}\frac{d\lambda}{dt} - \sigma_t^4\frac{d\lambda}{dt} = (\sigma_t^4 - \sigma_t^2)\frac{d\lambda}{dt} - \sigma_t^4\frac{d\lambda}{dt} = -\sigma_t^2\frac{d\lambda}{dt}. \tag{14}$$

Plugging these in to the probability flow ODE then gives

$$d\mathbf{z} = [f(\mathbf{z}, t) - \frac{1}{2}g^2(t)\nabla_z \log p_t(\mathbf{z})]dt \tag{15}$$

$$= \frac{1}{2}\sigma_\lambda^2[\mathbf{z}_\lambda + \nabla_z \log p_\lambda(\mathbf{z})]d\lambda. \tag{16}$$

Plugging in our function approximation from Equation 12 gives

$$d\mathbf{z} = \frac{1}{2}\sigma_\lambda^2\left[\mathbf{z}_\lambda + \left(\frac{\alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) - \mathbf{z}_\lambda}{\sigma_\lambda^2}\right)\right]d\lambda \tag{17}$$

$$= \frac{1}{2}[\alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) + (\sigma_\lambda^2 - 1)\mathbf{z}_\lambda]d\lambda \tag{18}$$

$$= \frac{1}{2}[\alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) - \alpha_\lambda^2\mathbf{z}_\lambda]d\lambda. \tag{19}$$

B DDIM IS AN INTEGRATOR OF THE PROBABILITY FLOW ODE

The DDIM update rule (Song & Ermon, 2020) is given by

$$\mathbf{z}_s = \frac{\sigma_s}{\sigma_t}[\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t)] + \alpha_s \hat{\mathbf{x}}_\theta(\mathbf{z}_t), \tag{20}$$

for $s < t$. Taking the derivative of this expression with respect to λ_s, assuming again a variance preserving diffusion process, and using $\frac{d\alpha_\lambda}{d\lambda} = \frac{1}{2}\alpha_\lambda\sigma_\lambda^2$ and $\frac{d\sigma_\lambda}{d\lambda} = -\frac{1}{2}\sigma_\lambda\alpha_\lambda^2$, gives

$$\frac{\mathbf{z}_{\lambda_s}}{d\lambda_s} = \frac{d\sigma_{\lambda_s}}{d\lambda_s}\frac{1}{\sigma_t}[\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t)] + \frac{d\alpha_{\lambda_s}}{d\lambda_s}\hat{\mathbf{x}}_\theta(\mathbf{z}_t) \tag{21}$$

$$= -\frac{1}{2}\alpha_s^2\frac{\sigma_s}{\sigma_t}[\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t)] + \frac{1}{2}\alpha_s\sigma_s^2\hat{\mathbf{x}}_\theta(\mathbf{z}_t). \tag{22}$$

Evaluating this derivative at $s = t$ then gives

$$\frac{\mathbf{z}_{\lambda_s}}{d\lambda_s}|_{s=t} = -\frac{1}{2}\alpha_\lambda^2[\mathbf{z}_\lambda - \alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda)] + \frac{1}{2}\alpha_\lambda\sigma_\lambda^2\hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) \tag{23}$$

$$= -\frac{1}{2}\alpha_\lambda^2[\mathbf{z}_\lambda - \alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda)] + \frac{1}{2}\alpha_\lambda(1 - \alpha_\lambda^2)\hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) \tag{24}$$

$$= \frac{1}{2}[\alpha_\lambda \hat{\mathbf{x}}_\theta(\mathbf{z}_\lambda) - \alpha_\lambda^2\mathbf{z}_\lambda]. \tag{25}$$

Comparison with Equation 19 now shows that DDIM follows the probability flow ODE up to first order, and can thus be considered as an integration rule for this ODE.

C EVALUATION OF INTEGRATORS OF THE PROBABILITY FLOW ODE

In a preliminary investigation we tried several numerical integrators for the probability flow ODE. As our model we used a pre-trained class-conditional 128x128 ImageNet model following the description in Ho et al. (2020). We tried a simple Euler integrator, RK4 (the "classic" 4th order Runge–Kutta integrator), and DDIM (Song et al., 2021a). In addition we compared to a Gaussian sampler with variance equal to the lower bound given by Ho et al. (2020). We calculated FID scores on just 5000 samples, hence our results in this experiment are not comparable to results reported in the literature. This preliminary investigation gave the results listed in Table 3 and identified DDIM as the best integrator in terms of resulting sample quality.

Sampler	Number of steps	FID
Stochastic	1000	**13.35**
Euler	1000	16.5
RK4	1000	16.33
DDIM	1000	15.98
Stochastic	100	18.44
Euler	100	23.67
RK4	100	18.94
DDIM	100	**16.35**

Table 3: Preliminary FID scores on 128×128 ImageNet for various integrators of the probability flow ODE, and compared against a stochastic sampler. Model specification and noise schedule follow Ho et al. (2020).

D EXPRESSION OF DDIM IN ANGULAR PARAMETERIZATION

We can simplify the DDIM update rule by expressing it in terms of $\phi_t = \arctan(\sigma_t/\alpha_t)$, rather than in terms of time t or log-SNR λ_t, as we show here.

Given our definition of ϕ, and assuming a variance preserving diffusion process, we have $\alpha_\phi = \cos(\phi)$, $\sigma_\phi = \sin(\phi)$, and hence $\mathbf{z}_\phi = \cos(\phi)\mathbf{x} + \sin(\phi)\epsilon$. We can now define the velocity of $\mathbf{z}_\phi$ as

$$\mathbf{v}_\phi \equiv \frac{d\mathbf{z}_\phi}{d\phi} = \frac{d\cos(\phi)}{d\phi}\mathbf{x} + \frac{d\sin(\phi)}{d\phi}\epsilon = \cos(\phi)\epsilon - \sin(\phi)\mathbf{x}. \tag{26}$$

Rearranging $\epsilon, \mathbf{x}, \mathbf{v}$, we then get

$$\sin(\phi)\mathbf{x} = \cos(\phi)\epsilon - \mathbf{v}_\phi \tag{27}$$

$$= \frac{\cos(\phi)}{\sin(\phi)}(\mathbf{z} - \cos(\phi)\mathbf{x}) - \mathbf{v}_\phi \tag{28}$$

$$\sin^2(\phi)\mathbf{x} = \cos(\phi)\mathbf{z} - \cos^2(\phi)\mathbf{x} - \sin(\phi)\mathbf{v}_\phi \tag{29}$$

$$(\sin^2(\phi) + \cos^2(\phi))\mathbf{x} = \mathbf{x} = \cos(\phi)\mathbf{z} - \sin(\phi)\mathbf{v}_\phi, \tag{30}$$

and similarly we get $\epsilon = \sin(\phi)\mathbf{z}_\phi + \cos(\phi)\mathbf{v}_\phi$.

Furthermore, we define the predicted velocity as

$$\hat{\mathbf{v}}_\theta(\mathbf{z}_\phi) \equiv \cos(\phi)\hat{\epsilon}_\theta(\mathbf{z}_\phi) - \sin(\phi)\hat{\mathbf{x}}_\theta(\mathbf{z}_\phi), \tag{31}$$

where $\hat{\epsilon}_\theta(\mathbf{z}_\phi) = (\mathbf{z}_\phi - \cos(\phi)\hat{\mathbf{x}}_\theta(\mathbf{z}_\phi))/\sin(\phi)$.

Rewriting the DDIM update rule in the introduced terms then gives

$$\mathbf{z}_{\phi_s} = \cos(\phi_s)\hat{\mathbf{x}}_\theta(\mathbf{z}_{\phi_t}) + \sin(\phi_s)\hat{\epsilon}_\theta(\mathbf{z}_{\phi_t}) \tag{32}$$

$$= \cos(\phi_s)(\cos(\phi_t)\mathbf{z}_{\phi_t} - \sin(\phi_t)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\phi_t})) + \sin(\phi_s)(\sin(\phi_t)\mathbf{z}_{\phi_t} + \cos(\phi_t)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\phi_t})) \tag{33}$$

$$= [\cos(\phi_s)\cos(\phi_t) - \sin(\phi_s)\sin(\phi_t)]\mathbf{z}_{\phi_t} + [\sin(\phi_s)\cos(\phi_t) - \cos(\phi_s)\sin(\phi_t)]\hat{\mathbf{v}}_\theta(\mathbf{z}_{\phi_t}). \tag{34}$$

Finally, we use the trigonometric identities

$$\cos(\phi_s)\sin(\phi_t) - \sin(\phi_s)\cos(\phi_t) = \cos(\phi_s - \phi_t) \tag{35}$$
$$\sin(\phi_s)\cos(\phi_t) - \cos(\phi_s)\sin(\phi_t) = \sin(\phi_s - \phi_t), \tag{36}$$

to find that

$$\mathbf{z}_{\phi_s} = \cos(\phi_s - \phi_t)\mathbf{z}_{\phi_t} + \sin(\phi_s - \phi_t)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\phi_t}). \tag{37}$$

or equivalently

$$\mathbf{z}_{\phi_t - \delta} = \cos(\delta)\mathbf{z}_{\phi_t} - \sin(\delta)\hat{\mathbf{v}}_\theta(\mathbf{z}_{\phi_t}). \tag{38}$$

Viewed from this perspective, DDIM thus evolves $\mathbf{z}_{\phi_s}$ by moving it on a circle in the $(\mathbf{z}_{\phi_t}, \hat{\mathbf{v}}_{\phi_t})$ basis, along the $-\hat{\mathbf{v}}_{\phi_t}$ direction. The relationship between $\mathbf{z}_{\phi_t}, \mathbf{v}_t, \alpha_t, \sigma_t, \mathbf{x}, \epsilon$ is visualized in Figure 5.

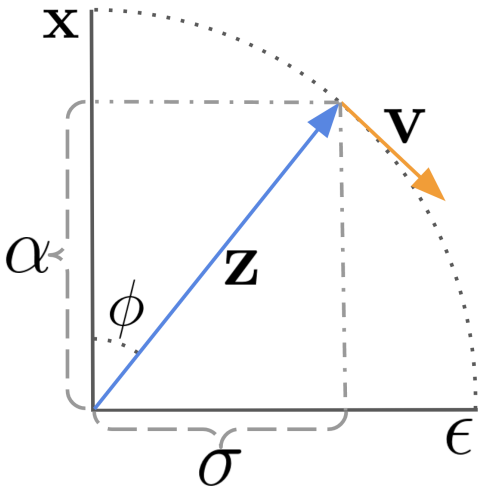


Figure 5: Visualization of reparameterizing the diffusion process in terms of ϕ and $\mathbf{v}_\phi$.

E SETTINGS USED IN EXPERIMENTS

Our model architectures closely follow those described by Dhariwal & Nichol (2021b). For 64×64 ImageNet we use their model exactly, with 192 channels at the highest resolution. All other models are slight variations with different hyperparameters.

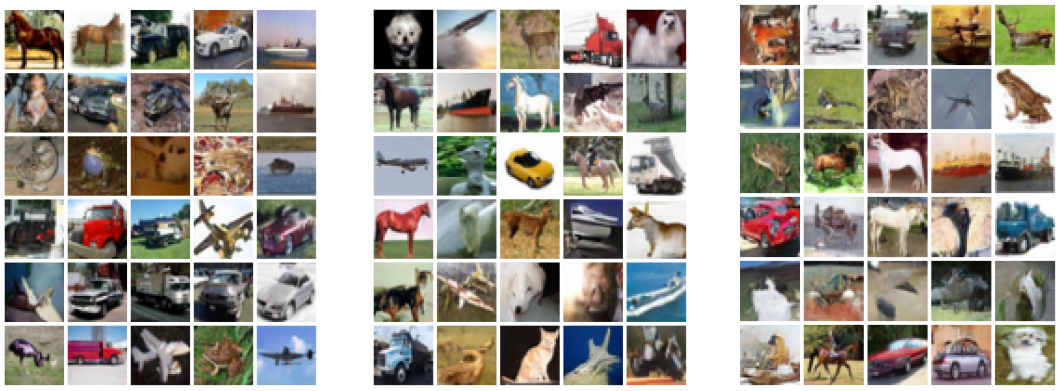
For CIFAR-10 we use an architecture with a fixed number of channels at all resolutions of 256. The model consists of a UNet that internally downsamples the data twice, to 16×16 and to 8×8. At each resolution we apply 3 residual blocks, like described by Dhariwal & Nichol (2021b). We use single-headed attention, and only apply this at the 16×16 and 8×8 resolutions. We use dropout of 0.2.

For LSUN we use a model similar to that for ImageNet, but with a reduced number of 128 channels at the 64×64 resolution. Compared to ImageNet we have an additional level in the UNet, corresponding to the input resolution of 128×128, which we process using 3 residual blocks with 64 channels. We only use attention layers for the resolutions of 32×32 and lower.

For CIFAR-10 we take the output of the model to represent a prediction of $\mathbf{x}$ directly, as discussed in Section 4. For the other data sets we used the combined prediction of $(\mathbf{x}, \epsilon)$ like described in that section also. All original models are trained with Adam with standard settings (learning rate of $3 * 10^{-4}$), using a parameter moving average with constant 0.9999 and very slight decoupled weight decay (Loshchilov & Hutter, 2017) with a constant of 0.001. We clip the norm of gradients to a global norm of 1 before calculating parameter updates. For CIFAR-10 we train for 800k parameter updates, for ImageNet we use 550k updates, and for LSUN we use 400k updates. During distillation we train for 50k updates per iteration, except for the distillation to 2 and 1 sampling steps, for which we use 100k updates. We linearly anneal the learning rate from 10^{-4} to zero during each iteration.

We use a batch size of 128 for CIFAR-10 and 2048 for the other data sets. We run our experiments on TPUv4, using 8 TPU chips for CIFAR-10, and 64 chips for the other data sets. The total time required to first train and then distill a model varies from about a day for CIFAR-10, to about 5 days for ImageNet.

F ADDITIONAL RANDOM SAMPLES

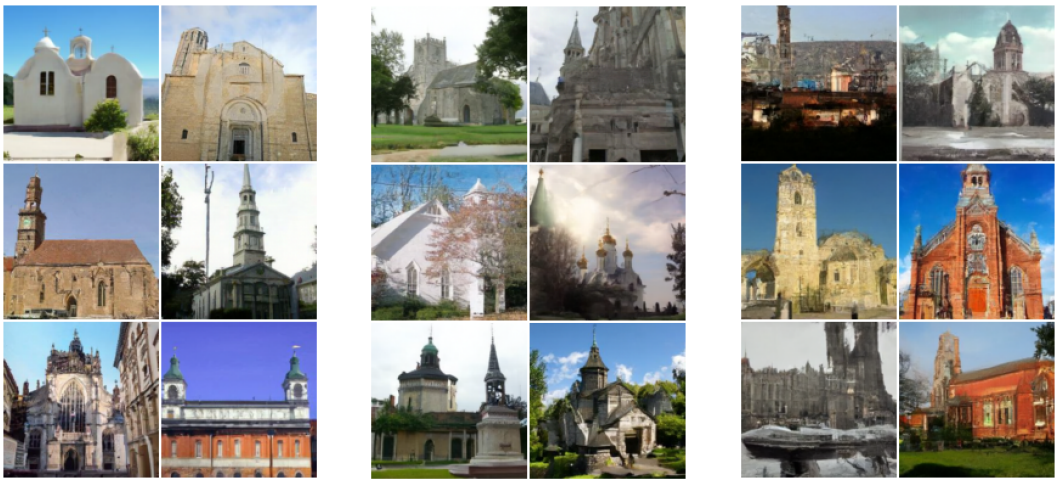


(a) 128 sampling steps (b) 4 sampling steps (c) 1 sampling step

Figure 6: Random samples from our distilled CIFAR-10 model, for varying numbers of sampling steps.

(a) 128 sampling steps (b) 4 sampling steps (c) 1 sampling step

Figure 7: Random samples from our distilled LSUN bedrooms model, for varying numbers of sampling steps.



(a) 128 sampling steps (b) 4 sampling steps (c) 1 sampling step

Figure 8: Random samples from our distilled LSUN church-outdoor model, for varying numbers of sampling steps.