Improved Noise Schedule for Diffusion Training

Supplementary Material

1. Detailed Implementation for Noise Schedule

We provide a simple PyTorch implementation for the Laplace noise schedule and its application in training. This example can be adapted to other noise schedules, such as the Cauchy distribution, by replacing the `laplace_noise_schedule` function. The model accepts noisy samples $\mathbf{x}_t$, timestep t, and an optional condition tensor $\mathbf{c}$ as inputs. This implementation supports prediction of $\{\mathbf{x}_0, \mathbf{v}, \boldsymbol{\epsilon}\}$.

2. Details for Proposed Laplace and Cauchy Design

For a Laplace distribution with location parameter μ and scale parameter b, the probability density function (PDF) is given by:

$$p(\lambda) = \frac{1}{2b} \exp\left(-\frac{|\lambda - \mu|}{b}\right) \tag{1}$$

The cumulative distribution function (CDF) can be derived as follows:

$$\begin{aligned} 1 - t &= \int_{-\infty}^{\lambda} p(x)\,\mathrm{d}x \\ &= \int_{-\infty}^{\lambda} \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right)\,\mathrm{d}x \\ &= \frac{1}{2}\left(1 + \mathrm{sgn}(\lambda - \mu)\left(1 - \exp\left(-\frac{|\lambda - \mu|}{b}\right)\right)\right) \end{aligned}$$

To obtain λ as a function of t, we solve the inverse function:

$$\lambda = \mu - b\mathrm{sgn}(0.5 - t)\ln(1 - 2|t - 0.5|)$$

For a Cauchy distribution with location parameter μ and scale parameter γ, the PDF is given by:

$$f(\lambda; \mu, \gamma) = \frac{1}{\pi\gamma}\left[1 + \left(\frac{\lambda - \mu}{\gamma}\right)^2\right]^{-1} \tag{2}$$

The corresponding CDF is:

$$F(\lambda; \mu, \gamma) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\lambda - \mu}{\gamma}\right) \tag{3}$$

To derive $\lambda(t)$, we proceed as follows:

$$1 - t = F(\lambda; \mu, \gamma) \tag{4}$$

$$1 - t = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{\lambda - \mu}{\gamma}\right) \tag{5}$$

$$t = \frac{1}{2} - \frac{1}{\pi}\arctan\left(\frac{\lambda - \mu}{\gamma}\right) \tag{6}$$

Solving for λ, we obtain:

$$\lambda(t) = \mu + \gamma\tan\left(\frac{\pi}{2}(1 - 2t)\right) \tag{7}$$

3. Combination between Noise schedule and Timestep Importance Sampling

We observe that incorporating importance sampling of timesteps into the cosine schedule bears similarities to the Laplace schedule. Typically, the distribution of timestep t is uniform $\mathcal{U}[0, 1]$. To increase the sampling frequency of middle-level timesteps, we propose modifying the sampling distribution to a simple polynomial function:

$$p(t') = \begin{cases} C \cdot t'^n, & t' < \frac{1}{2} \\ C \cdot (1 - t')^n, & t' \geq \frac{1}{2}, \end{cases} \tag{8}$$

where $C = (n + 1)2^n$ is the normalization factor ensuring that the cumulative distribution function (CDF) equals 1 at $t = 1$.

To sample from this distribution, we first sample t uniformly from $(0, 1)$ and then map it using the following function:

$$t' = \begin{cases} \left(\frac{1}{2}\right)^{\frac{n}{n+1}} t^{\frac{1}{n+1}}, & t < \frac{1}{2} \\ 1 - \left(\frac{1}{2}\right)^{\frac{n}{n+1}} (1 - t)^{\frac{1}{n+1}}, & t \geq \frac{1}{2}, \end{cases} \tag{9}$$

We incorporate the polynomial sampling of t into the cosine schedule $\lambda = -2\log\tan\frac{\pi t}{2}$, whose inverse function is $t = \frac{2}{\pi}\arctan\exp\left(-\frac{\lambda}{2}\right)$. Let us first consider the situation where $t < \frac{1}{2}$:

$$\left(\frac{1}{2}\right)^{\frac{n}{n+1}} t^{\frac{1}{n+1}} = \frac{2}{\pi}\arctan\exp\left(-\frac{\lambda}{2}\right) \tag{10}$$

$$t = 2^n\left(\frac{2}{\pi}\arctan\exp\left(-\frac{\lambda}{2}\right)\right)^{n+1} \tag{11}$$

We then derive the expression with respect to $\mathrm{d}\lambda$:

```python
import torch


def laplace_noise_schedule(mu=0.0, b=0.5):
    # refer to Table 1
    lmb = lambda t: mu - b * torch.sign(0.5 - t) * \
        torch.log(1 - 2 * torch.abs(0.5 - t))
    snr_func   = lambda t: torch.exp(lmb(t))
    alpha_func = lambda t: torch.sqrt(snr_func(t) / (1 + snr_func(t)))
    sigma_func = lambda t: torch.sqrt(1 / (1 + snr_func(t)))

    return alpha_func, sigma_func


def training_losses(model, x, timestep, condition, noise=None,
                    predict_target="v", mu=0.0, b=0.5):

    if noise is None:
        noise = torch.randn_like(x)

    alpha_func, sigma_func = laplace_noise_schedule(mu, b)
    alphas = alpha_func(timestep)
    sigmas = sigma_func(timestep)

    # add noise to sample
    x_t = alphas.view(-1, 1, 1, 1) * x + sigmas.view(-1, 1, 1, 1) * noise
    # velocity
    v_t = alphas.view(-1, 1, 1, 1) * noise - sigmas.view(-1, 1, 1, 1) * x

    model_output = model(x_t, timestep, condition)
    if predict_target == "v":
        loss = (v_t - model_output) ** 2
    elif predict_target == "x0":
        loss = (x - model_output) ** 2
    else: # predict_target == "noise":
        loss = (noise - model_output) ** 2

    return loss.mean()
```

$$\frac{\mathrm{d}t}{\mathrm{d}\lambda} = 2^n \left(\frac{2}{\pi}\right)^{n+1} (n+1) \left(\arctan\exp\left(-\frac{\lambda}{2}\right)\right)^n \tag{12}$$

$$\times \frac{1}{1+\exp(-\lambda)} \frac{1}{-2} \exp(-\lambda/2) \tag{13}$$

$$p(\lambda) = (n+1)\frac{4^n}{\pi^{(n+1)}} \arctan^n \exp\left(-\frac{\lambda}{2}\right) \frac{\exp(-\frac{1}{2}\lambda)}{1+\exp(-\lambda)} \tag{14}$$

Considering symmetry, we obtain the final distribution with respect to λ as follows:

$$p(\lambda) = (n+1)\frac{4^n}{\pi^{(n+1)}} \arctan^n \exp\left(-\frac{|\lambda|}{2}\right) \frac{\exp(-\frac{1}{2}|\lambda|)}{1+\exp(-|\lambda|)} \tag{15}$$

We visualize the schedule discussed above and compare it with Laplace schedule in Figure 1. We can see that $b = 1$ for Laplace and $n = 2$ for cosine-ply matches well. We also conduct experiments on such schedule and present results in Table 1. They perform similar and both better than the standard cosine schedule.

We visualize the schedules discussed above and compare them with the Laplace schedule in Figure 1. The results demonstrate that Laplace with $b = 1$ and cosine-ply with $n = 2$ exhibit a close correspondence. To evaluate the

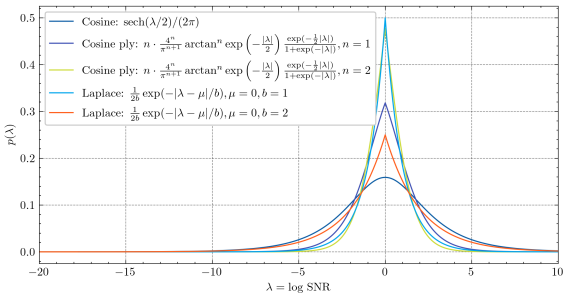

Figure 1. Visualization of $p(\lambda)$ for Laplace schedule and cosine schedule with polynomial timestep sampling.

performance of these schedules, we conducted experiments and present the results in Table 1. Both the Laplace and cosine-ply schedules show similar performance, and both outperform the standard cosine schedule.

4. Flow Matching with Logit-Normal Sampling

In Stable Diffusion 3 [1] and Movie Gen [5], logit-normal sampling is applied to improve the training efficiency of flow models. To better understand this approach, we present a detailed derivation from the logit-normal distribution to the probability density function of logSNR λ.

Let the Logit transformation $X = \text{logit}(t)$ of random variable t follow a normal distribution:

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{16}$$

Then, the probability density function of t is:

$$p(t;\mu,\sigma) = \frac{1}{\sigma \cdot t \cdot (1-t) \cdot \sqrt{2\pi}} \exp\left(-\frac{(\text{logit}(t)-\mu)^2}{2\sigma^2}\right), \tag{17}$$

where $\text{logit}(t) = \log\left(\frac{t}{1-t}\right)$, and μ and σ are constants.

Consider the variable transformation:

$$\lambda = 2\log\left(\frac{1-t}{t}\right) \tag{18}$$

Our goal is to find the probability density function $p(\lambda)$ of random variable λ.

First, we solve for t in terms of λ:

$$\begin{aligned}
\frac{\lambda}{2} &= \log\left(\frac{1-t}{t}\right) \\
e^{\frac{\lambda}{2}} &= \frac{1-t}{t} \\
1-t &= te^{\frac{\lambda}{2}} \\
1 &= t\left(1+e^{\frac{\lambda}{2}}\right) \\
t(\lambda) &= \frac{1}{1+e^{\frac{\lambda}{2}}}
\end{aligned}$$

Next, we calculate the Jacobian determinant $\left|\frac{\mathrm{d}t}{\mathrm{d}\lambda}\right|$:

$$\begin{aligned}
t(\lambda) &= \frac{1}{1+e^{\frac{\lambda}{2}}} \\
\frac{\mathrm{d}t}{\mathrm{d}\lambda} &= -\frac{e^{\frac{\lambda}{2}} \cdot \frac{1}{2}}{(1+e^{\frac{\lambda}{2}})^2} \\
\left|\frac{\mathrm{d}t}{\mathrm{d}\lambda}\right| &= \frac{e^{\frac{\lambda}{2}}}{2(1+e^{\frac{\lambda}{2}})^2}
\end{aligned}$$

Using the variable transformation formula:

$$p(\lambda) = p(t(\lambda);\mu,\sigma) \cdot \left|\frac{\mathrm{d}t}{\mathrm{d}\lambda}\right| \tag{19}$$

We calculate $p(t(\lambda);\mu,\sigma)$:

$$\text{logit}(t(\lambda)) = \log\left(\frac{t(\lambda)}{1-t(\lambda)}\right) = \log\left(\frac{\frac{1}{1+e^{\frac{\lambda}{2}}}}{\frac{e^{\frac{\lambda}{2}}}{1+e^{\frac{\lambda}{2}}}}\right) = -\frac{\lambda}{2}$$

$$p(t(\lambda);\mu,\sigma) = \frac{(1+e^{\frac{\lambda}{2}})^2}{\sigma e^{\frac{\lambda}{2}}\sqrt{2\pi}} \exp\left(-\frac{(\mu+\frac{\lambda}{2})^2}{2\sigma^2}\right)$$

Multiplying by the Jacobian determinant:

$$\begin{aligned}
p(\lambda) &= \frac{(1+e^{\frac{\lambda}{2}})^2}{\sigma e^{\frac{\lambda}{2}}\sqrt{2\pi}} \exp\left(-\frac{(\mu+\frac{\lambda}{2})^2}{2\sigma^2}\right) \cdot \frac{e^{\frac{\lambda}{2}}}{2(1+e^{\frac{\lambda}{2}})^2} \\
&= \frac{1}{2\sigma\sqrt{2\pi}} \exp\left(-\frac{(\lambda+2\mu)^2}{8\sigma^2}\right)
\end{aligned}$$

Therefore, the probability density function of λ is:

$$p(\lambda) = \frac{1}{2\sigma\sqrt{2\pi}} \exp\left(-\frac{(\lambda+2\mu)^2}{8\sigma^2}\right), \quad \lambda \in (-\infty, +\infty)$$
$$t \in (0,1) \tag{20}$$

This shows that λ follows a normal distribution with mean -2μ and variance $4\sigma^2$:

$$\lambda \sim \mathcal{N}(-2\mu, 4\sigma^2) \tag{21}$$

The mean and variance are:

$$\begin{aligned}
\mathbb{E}[\lambda] &= -2\mu \\
\text{Var}(\lambda) &= 4\sigma^2
\end{aligned}$$

To verify normalization, we integrate $p(\lambda)$ over its domain:

$$\begin{aligned}
\int_{-\infty}^{+\infty} p(\lambda)\,\mathrm{d}\lambda &= \int_{-\infty}^{+\infty} \frac{1}{2\sigma\sqrt{2\pi}} \exp\left(-\frac{(\lambda+2\mu)^2}{8\sigma^2}\right)\mathrm{d}\lambda \\
\text{Let } z = \frac{\lambda+2\mu}{2\sqrt{2}\sigma} &\Rightarrow \mathrm{d}\lambda = 2\sqrt{2}\sigma\,\mathrm{d}z \\
&= \frac{2\sqrt{2}\sigma}{2\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-z^2}\mathrm{d}z \\
&= \frac{1}{\sqrt{\pi}} \cdot \sqrt{\pi} = 1
\end{aligned}$$

Iterations	100,000	200,000	300,000	400,000	500,000
Cosine-ply ($n = 2$)	28.65	13.77	10.06	8.69	7.98
Laplace ($b = 1$)	28.89	13.90	10.17	8.85	8.19

Table 1. Performance comparison of cosine-ply ($n = 2$) and Laplace ($\mu = 1$) schedules over different iteration counts

Thus, $p(\lambda)$ satisfies the normalization condition for probability density functions.

We compare the standard cosine scheudle [4], Flow Matching [2, 3], and Flow Matching with Logit-normal sampling [1, 5]. The probability density functions of these schedules are visualized in Figure 2. Our analysis reveals that Flow Matching with Logit-normal sampling concentrates more probability mass around $\lambda = 0$ compared to both the standard Cosine and Flow Matching schedules, resulting in improved training efficiency [1, 5].

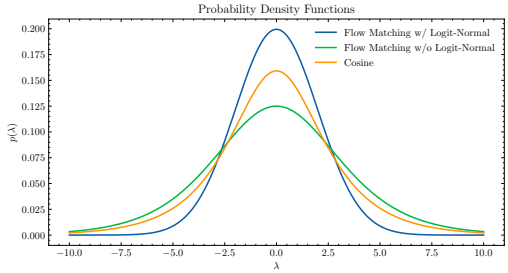

Figure 2. Comparison of probability density functions for different flow matching approaches. The plot shows three distributions: Flow Matching with Logit-Normal sampling (blue), Flow Matching without Logit-Normal sampling (green), and the Cosine schedule (orange).

5. Importance of Time Intervals

To investigate the significance of training intervals, we conducted controlled experiments using a simplified setup. We divided the time range $(0, 1)$ into four equal segments: $\text{bin}_i = \left(\frac{i}{4}, \frac{i+1}{4}\right), i = 0, 1, 2, 3$. We first trained a base model $\mathbf{M}$ over the complete range $(0, 1)$ for 1M iterations, then fine-tuned it separately on each bin for 140k iterations to obtain four specialized checkpoints $\mathbf{m}_i, i = 0, 1, 2, 3$.

For evaluation, we designed experiments using both the base model $\mathbf{M}$ and fine-tuned checkpoints $\mathbf{m}_i$. To assess the importance of each temporal segment, we selectively employed the corresponding fine-tuned checkpoint during its specific interval while maintaining the base model for remaining intervals. For example, when evaluating bin_0, we used $\mathbf{m}_0$ within its designated interval and $\mathbf{M}$ elsewhere.

The FID results across these four experimental configurations are presented in Figure 3. Our analysis reveals that optimizing intermediate timesteps (bin1 and bin2) yields superior performance, suggesting the critical importance of these temporal regions in the diffusion process.

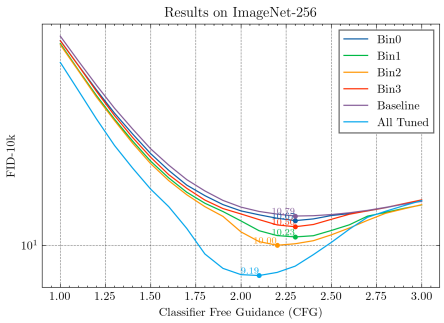

Figure 3. Comparative analysis of interval-specific fine-tuning effects. When sampling within interval $\left(\frac{1}{4}, \frac{2}{4}\right)$, "Bin1" indicates the use of fine-tuned weights $\mathbf{m}_1$, while $\mathbf{M}$ is used for other intervals. "Baseline" represents the use of base model $\mathbf{M}$ throughout all intervals, and "All Tuned" denotes the application of interval-specific fine-tuned models within their respective ranges.

References

[1] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024. 3, 4

[2] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022. 4

[3] Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022. 4

[4] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 4

[5] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 3, 4