Noise Schedulers in Denoising Diffusion Probabilistic Models

Dhruv Srikanth
Department of Computer Science
University of Chicago
`dhruvsrikanth@uchicago.edu`

Abstract

This work is an extension of the Denoising Diffusion Probabilistic Models [3] paper. We explore four different noise schedulers and their effects when training denoising diffusion probabilistic models. Different noise schedulers have been shown to produce varying results in terms of generated sample quality during inference [8]. Throughout this work, we will draw comparisons between the approach taken and the work by Ho et al. [3]. The goal of this work is to gain insights into the effects of the beta scheduler on the training time, loss obtained after the model converges, and sample quality during inference. The codebase for this work is available at `https://github.com/DhruvSrikanth/DenoisingDiffusionProbabilisticModels`

1 Introduction

Diffusion probabilistic models ("diffusion models" for brevity) [10] were introduced as a class of generative models that learn a data distribution by modeling the iterative reversal of the noising process over multiple steps. It has been shown that this class of generative models can produce high-quality outputs in multiple modalities - in the image space [3, 11, 4] as well as in the audio space [1, 6]. This work extends the diffusion models proposed by Ho et al. [3], however, as shown by Nichol and Dhariwal [8] in the image space and Chen et al. [1] in the audio space, we can efficiently generate outputs in far fewer sampling steps compared to that used by Ho et al. [3]. In this work, we explore different beta schedulers in the training procedure for denoising diffusion probabilistic models ("DDPMs" for brevity) and their effects while generating images during inference. We attempt to replicate the results shown by Ho et al. [3] on the CIFAR-10 [7] dataset. Though we are only performing experiments on the CIFAR-10 [7] dataset, Nichol and Dhariwal [8] has been able to show that DDMPs are capable of generating high-quality images even when trained on datasets with higher diversity such as ImageNet. We take inspiration from Nichol and Dhariwal [8] and Chen et al. [1] in reducing the number of steps in the sampling procedure and explore different noise schedules such as a linear, quadratic, sigmoid, and cosine [8]. The goal of this work is to gain insights into the effects of the noise schedule on the training time, in the hopes that we can reduce the computing time and resources required to train the model, as well as the effects of the noise schedule on generated samples during inference.

2 Denoising Diffusion Probabalistic Models

Denoising diffusion probabilistic models or DDPMs are a class of generative models used to learn a data distribution by modeling the gradual reversal in a noising process that takes place over multiple steps. This noising process is determined by a noise schedule. Ho et al. [3] run the noising process using a linear schedule over 1000 timesteps, however, in our experiments, we drastically reduce this number to 300. This allows us to save on computational resources during both training and inference.

We may view this forward noising process as a fixed Markov chain [3]. Given a data distribution $x_0 \sim q(x_0)$, we define a noising or forward diffusion process q as follows:

$$q(x_1, \cdots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

, where T represents the number of timesteps the forward diffusion process q is run for and $x_1, \cdots, x_T$ represent the latent samples in the noising process i.e. samples with increasing degrees of Gaussian noise added. This Gaussian noise has 0 mean and variance $\beta_t \in (0, 1)$. Ho et al. [3] show that given a large enough value of T and well behaved β_t, the final noised sample x_t produced by the forward process q is close to an isotropic Gaussian distribution.

It is possible to obtain the previous sample in the noising process by reversing q recursively as $q(x_{t-1}|x_t)$. DDPMs attempt to learn the reverse process of obtaining $q(x_0|x_t)$ by approximating $q(x_0)$ using a neural network:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3}$$

We may view this as a model learning a Markov chain of Gaussian transitions [3]. Though the combination of q and p resembles a variational auto-encoder [5] and we can learn the reverse process by optimizing over the variational lower bound (VLB), while training Ho et al. [3] show that minimizing the predicted noise ϵ on a sample is equivalent to optimizing over the variational lower bound. Therefore, we may write a simplified learning objective as follows:

$$L_{objective} = E_{t,x_0,\epsilon}||\epsilon - \epsilon_\theta(x_t, t)||^2 \tag{4}$$

, where ϵ refers to the known amount of noise and ϵ_θ refers to the predicted noise. Following Ho et al. [3], we will use the simplified objective described by equation 4 when training our models, however, we use *huber* loss instead of the mean-squared error loss shown in equation 4 due to it's robustness to outliers.

3 Noise Schedules

The forward process carried by Ho et al. [3] uses a linear schedule over $T = 1000$ timesteps. Chen et al. [1] show that we can reduce the number of timesteps and still obtain good results. Furthermore, Nichol and Dhariwal [8] demonstrate that the model does not get much worse if we skip up to 20% in the reverse diffusion process while using a linear schedule. This is due to the information present in the latent samples $x_1, \cdots, x_T$ being destroyed too quickly by the noise schedule $\beta_1, \cdots, \beta_T$.

Linear: A linear noise schedule to produce β_t where $t \in [1, \cdots, T]$ can be computed as follows-

$$\beta_t = \beta_1 + \frac{\beta_T - \beta_1}{T-1} \times (t-1) \tag{5}$$

Quadratic: A quadratic noise schedule to produce β_t where $t \in [1, \cdots, T]$ can be computed as follows-

$$\beta_t = \left(\sqrt{\beta_1} + \frac{\sqrt{\beta_T} - \sqrt{\beta_1}}{T-1} \times (t-1)\right)^2 \tag{6}$$

Sigmoid: A sigmoid noise schedule to produce β_t where $t \in [1, \cdots, T]$ can be computed as follows-

$$\beta_t = \beta_1 + \sigma\left(-6 + \frac{6-(-6)}{T-1} \times (t-1)\right) \times (\beta_T - \beta_1) \tag{7}$$

, where $\sigma(x) = \frac{1}{1+e^{-x}}$

Cosine: A cosine noise schedule to produce β_t where $t \in [1, \cdots, T]$ can be computed as follows-

$$f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2 \tag{8}$$

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \tag{9}$$

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \tag{10}$$

Following Nichol and Dhariwal [8], we set $s = 0.008$ and clip β_t to be between 0.0001 and 0.9999.

Note the differences between the effect of different noise schedules in the forward diffusion process from Figure 1. We see that the linear and cosine schedules destroy information in the image faster than the quadratic and sigmoid noise schedules. The effect of this on the model is that during training, we can skip a percentage of the start of the reverse diffusion process without significantly worsening the model's performance. [8].

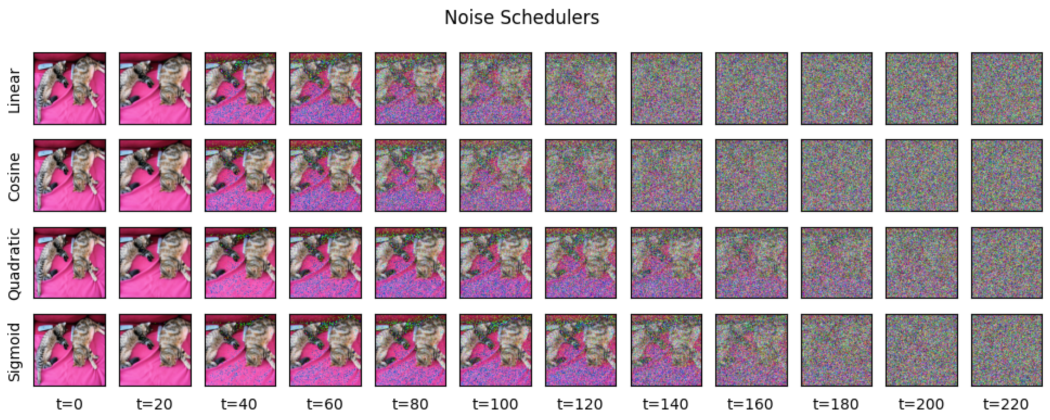

Figure 1: Latent samples for the linear, cosine, quadratic, and sigmoid noise schedules at linearly spaced values of t from 0 to T.

4 Experiments

Data: We carry out all experiments on the CIFAR-10 [7] dataset. This includes 60000 32x32 RGB images (50000 for training and 10000 for testing/validation) uniformly distributed across 10 classes. During training, the data is scaled to $[-1, 1]$.

Hardware: All training is performed on *Google Colab*. Google offers an Nvidia K80 GPU with 12GB of VRAM.

Model: We construct a model closely resembling the DDPM model by Ho et al. [3]. We train the model which is a U-Net architecture [9]. The network includes residual connections to improve gradient flow [2]. The parameters of the network are shared across time (noise level). In order to achieve this Ho et al. [3] utilize sinusoidal positional embeddings, inspired by the transformer architecture [12]. In addition, Ho et al. [3] interleave the convolution and attention layers of the U-Net with group normalization [13].

Hyperparameters	
batch size	128
learning rate	0.001
epochs	50
β_1	0.0001
β_T	0.02
$T(timesteps)$	300

Table 1: Hyperparameters used in training DDPMs using linear, quadratic, sigmoid, and cosine noise schedules.

Noise Schedule	Training Time (min)
Linear	137.5
Cosine	132
Quadratic	134
Sigmoid	137

Table 2: Training time in minutes for models trained for 50 epochs, each with a different noise schedule.

Hyperparameters: A list of the hyperparameters can be seen in Table 1.

We train four models, one for each noise schedule. The hyperparameters remain constant over all training cycles in order to compare the results between models. For each model, we observe the training and validation losses, as seen in Figure 2, the training times shown in Table 2, and the number of epochs required to converge as observed in Figure 2. *Google Colab* only allows a certain number of compute hours on their GPUs for free. Given the resource constraint, we restricted training all models for 50 epochs each. In addition, we observe the training time for each model. In the next section, we will discuss the results of the training.

5 Results

5.1 Quantitative Results

As shown in Figure 2, models trained using the sigmoid and quadratic schedulers converge around epoch 50 and 40 respectively. However, we can obtain better performance by training for longer. In the case of the linear and cosine schedules, the models appear to reach comparable loss values (when compared with the sigmoid and quadratic schedules) at around epoch 20. However, the loss curves for the models trained using the linear and cosine schedules continue to drop even after this, while the loss curves of the quadratic and sigmoid schedules flatten out. One possible inference that can be made from this is that the model is learning the distribution quicker. However, given that only the noise schedule is changing across these models, it is more likely that the noise schedule is influencing the model into learning faster. There exists the possibility that faster may not imply better performance. We can note from Figure 1, that the linear and cosine schedules destroy information faster. Therefore, it is possible that the model is finding it easier to learn as there are many steps at the start of the reverse diffusion process that are unnecessary and do not contribute much to learning the underlying data distribution [8].

From Table 2, we observe a similar training time on across all models trained. This is expected given the similar compute time for each scheduler in the forward process. Given that this is the only degree of freedom across all models trained, it is expected that the training time is similar. However, we can draw a conclusion based on the similar training time in Table 2, loss curves from Figure 2, and the effect of the forward process, seen in Figure 1. Though the training time is the same, the models trained using the linear and cosine schedules can be trained in less time by skipping the start of the reverse diffusion process as it does not contain enough useful information to improve model performance significantly [8].

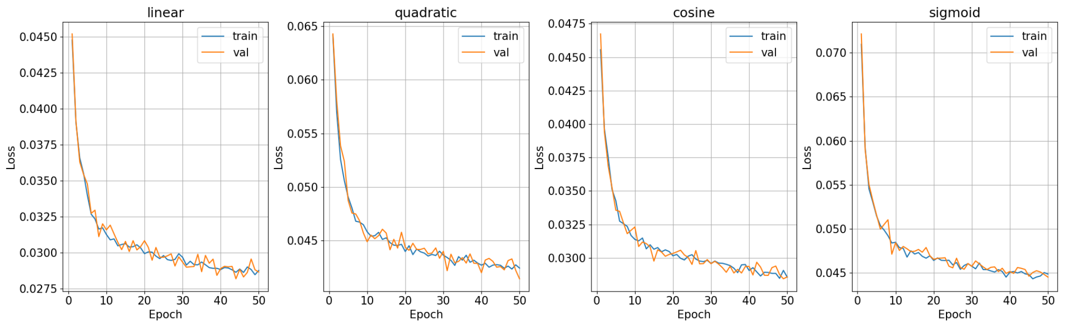

Figure 2: Training and validation losses for models trained with different noise schedules over 50 epochs.

5.2 Qualitative Results

It is clear from Figure 3 that the model using the cosine schedule did not learn the data distribution. This indicates that the model hyperparameters need to be tuned further. This includes allowing the model to train for several more epochs. However, the models utilizing linear, quadratic, and sigmoid noise schedules are generating relatively good samples. That being said, there are no qualitative differences that can be discerned between the generated samples for the latter models. One possible hypothesis that can be drawn is the effect of noise schedules is minimal during inference. However, it should be noted that the same noise schedule must be used in the forward process during training and inference. If a different noise schedule is used, the outputs resemble that of the generated samples by the model using the cosine schedule. Though this may seem obvious, it is not unreasonable to expect that similar noise schedules should be interchangeable in training and inference. An extension of this work could explore cycling noise schedules during training to produce a more robust model for inference. Another extension that approaches this topic is by Nichol and Dhariwal [8], in which the noise schedule is approximated using a neural network.

6 Conclusion

In this work, we explore the effect of different noise schedules on denoising diffusion probabilistic models during training and inference. We conclude that in order to make concrete inferences based on the qualitative results of the model, we must train for longer and tune the hyperparameters further. This is especially evident in the generated samples for the model using the cosine schedule. However, it is apparent that the effect of the noise schedule is largely during training and not inference. Furthermore, based on the quantitative results, we conclude that the model training time can be cut down by observing the effect of the noise schedule on the forward process [8]. This in turn will help to reduce the computational requirement of this class of generative models.

7 Acknowledgments

This paper is a part of my final project for a PhD course on Deep Learning. I'd like to thank Dr. David McAllester for making the course, *TTIC 31230: Fundamentals of Deep Learning*, so interesting.

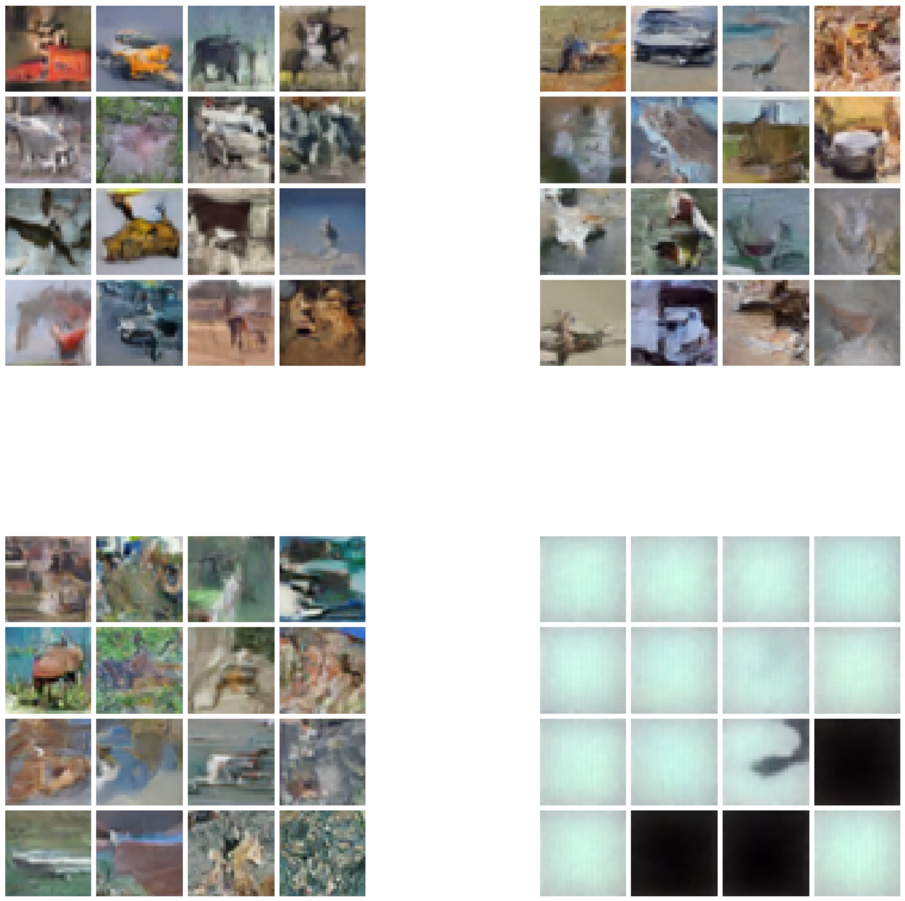

Figure 3: Generated samples using different noise schedules. The top left image grid corresponds to the model using a sigmoid schedule. The bottom left image grid corresponds to the model using a quadratic schedule. The top right image grid corresponds to the model using a linear schedule. The bottom right image grid corresponds to the model using a cosine schedule.

References

[1] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=NsMLjcFaO8O`.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

[4] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *CoRR*, abs/2009.05475, 2020. URL `https://arxiv.org/abs/2009.05475`.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL `https://arxiv.org/abs/1312.6114`.

[6] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=a-xFK8Ymz5J`.

[7] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[8] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/nichol21a.html`.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL `http://arxiv.org/abs/1505.04597`.

[10] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/sohl-dickstein15.html`.

[11] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *CoRR*, abs/2006.09011, 2020. URL `https://arxiv.org/abs/2006.09011`.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[13] Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018. URL `http://arxiv.org/abs/1803.08494`.