Improvements to the Sampling Speed of Diffusion Models

Candidate Number: 1043516

Abstract

Diffusion Probabilistic Models are a type of generative machine learning model that first gradually adds noise to input data until it is completely corrupted then learns to reverse this process to produce new samples from pure noise. Diffusion Models have been shown [4] to improve upon existing generative models, however, their slow sampling speeds is a significant drawback to their use. In this paper I explore model adjustments aimed to improve on the sampling speed of Diffusion Models.

1 Introduction

Diffusion Probabilistic Models, first introduced by Sohl-Dickstein et.al. in 2015 [21], are a new powerful tool in the machine learning sphere. They have proved to be very effective in a wide range of applications such as image synthesis [4],[7],[22], audio processing [1],[13] and text-to-media. Stable Diffusion, a text-to-image latent Diffusion Model, has received widespread media attention for its application in software such as DALL-E [19].

However, although Diffusion Models can produce impressive results, they have the disadvantage of slow sampling speeds. This is mainly due to the multiple denoising steps needed to sample from Diffusion Models. There have been several methods explored to improve the sampling speeds, which fall into three main categories 1) Training algorithm improvements, 2) Training-free sampling methods which improve the sampling algorithm itself without adjusting the training algorithm and 3) Mixed-Models, which combine Diffusion Models with elements of other generative models. In this paper, I will outline several sampling speed improvement techniques, but will focus particularly on the method *Progressive Distillation*, a form of knowledge distillation introduced in the paper [20] which was presented at ICLR 2022.

2 Background

Diffusion Models consist of two phases, the forward diffusion process and the reverse process. Using the notation in [18] and based on the work of [7], the forward process takes in the provided training data x_0 which is distributed according to $q(x_0)$, and then adds Gaussian noise with variance β_t at times $t = 1, ..., T$. At each point this creates a latent variable x_t where $x_t|x_{t-1}$ is normally distributed as described in equation 1, forming a Markov Chain with T steps. Using the Chain Rule we can derive equation 2:

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

$$q(x_1, \ldots, x_T \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \tag{2}$$

If T is large enough, x_T can be approximated by an Isotropic Gaussian random variable. Therefore, if we learn the reverse process, $q(x_{t-1}|x_t)$ we can sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ and then create new data that could have been drawn from the original distribution, $q(x_0)$.

.

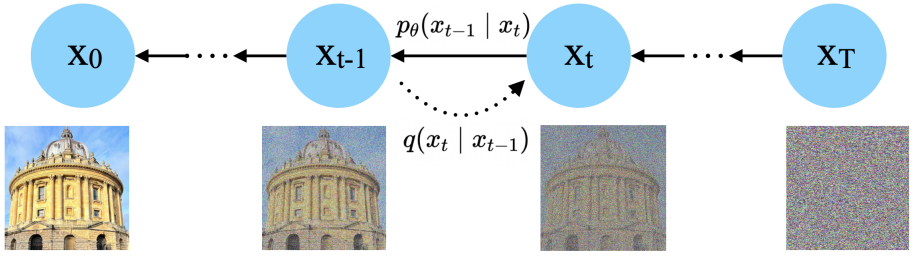

Figure 1: A diagram of the standard Denoising Diffusion Probabilistic Model

Using the reparameterization in [7], we let $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. This allows us, for any time t, to write the distribution of x_t conditioned on x_0:

$$q\left(x_t \mid x_0\right) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} x_0, \left(1 - \bar{\alpha}_t\right) \mathbf{I}\right) \tag{3}$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \tag{4}$$

For the reverse process, we cannot directly calculate $q(x_{t-1} \mid x_t)$ as it depends on the whole data distribution. Therefore, we approximate the distribution with a neural network, p_θ. If we restrict $\beta_t \in (0, 1)$, then the reverse distribution is approximately normally distributed. Using the Chain Rule again, we can write the joint probability distribution as shown in equation 6.

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{5}$$

$$p_\theta(x_1, \ldots, x_T) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t) \tag{6}$$

Combining q and p_θ we have something very similar to a Variational Auto-Encoder [10]. Therefore, we can write a lower bound on the log-likelihood using the Variational Lower Bound (VLB), where D_{KL} denotes the Kullback–Leibler divergence.

$$\begin{aligned} L_{\text{vlb}} &:= L_0 + L_1 + \ldots + L_{T-1} + L_T \\ L_0 &:= -\log p_\theta(x_0 \mid x_1) \\ L_{t-1} &:= D_{KL}(q(x_{t-1} \mid x_t, x_0) \| p_\theta(x_{t-1} \mid x_t)) \\ L_T &:= D_{KL}(q(x_T \mid x_0) \| p(x_T)) \end{aligned}$$

The KL Divergence of two Gaussian random variables can be evaluated in closed form. Using the Bayes Rule, it can be shown that $q(x_{t-1} \mid x_t, x_0)$ is normally distributed as described below:

$$\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$$

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t \tag{7}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}) \tag{8}$$

Therefore, the terms $L_t,\ t = 1, ..., T$ are computable and we can calculate L_0 by plugging $t = 1$ into equation 5. Our training objective, L_{vlb}, can be estimated by sampling $t \in 1, ..., T$ and then calculating $E_{t,x_0,\epsilon}[L_{t-1}]$. We can estimate the L_{t-1} by sampling x_t using equation 4 and then using the prior, 5, and posterior, 8.

To approximate the prior 5 we need to train a neural network to predict $\mu_\theta\left(x_t, t\right)$. There are several methods to achieve this; we could predict μ_θ directly, predict x_0 then use equation 7 or predict the noise ϵ then use equations 4 and 7 to calculate μ_θ. The paper [7] showed that predicting using ϵ worked best and used a simplified reweighted loss function:

$$L_{\text{simple}} = E_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

3 Sampling Speed Improvement Methods

Diffusion Models can produce high quality samples and have been shown to improve on existing generative models such as GANs in the seminal paper [4]. They can also generate diverse samples as shown by high-likelihood in [11],[23] and therefore fulfil two parts of what [25] termed the *Generative Learning Trilemma*. However, their slow sampling speed, the third part to the GLT, is a distinct drawback to their use.

One way to improve the sampling speed is to adjust the training algorithm of the Diffusion Model. This can be achieved by changing the noise schedule as proposed in [11],[18],[26] or by treating the number of diffusion steps as a training parameter [5]. Other methods include *Come-Closer Diffuse Faster* [3], *Truncation* [27] and *Early-Stopped DDPM* [17] which start the reverse process from partly noised data instead of pure Gaussian noise and reverses the process using alternative distributions. Knowledge Distillation is another technique used [16] and I will focus particularly on the method *Progressive Distillation* introduced in [20].

Another way to improve sampling speed is to adjust the sampling algorithm. One example of this is to use latent variables in a process called implicit sampling resulting in *Denoising Diffusion Implicit Models* (DDIM) [22]. Another method introduced in [12] is *FastDPM* which generalises the discrete-step diffusion model into one with continuous steps. In [22] they showed that the reverse process could be treated as a probability flow ODE [2] and therefore, there has been several techniques introducing ODE solver refinements that aim to improve sampling speed, such as: *Diffusion Exponential Integrator Sampler* [26], *Pseudo Numerical Methods* [14] and *DPM-solver* [15].

The third main method to speed-up sampling is to mix Diffusion Models with elements of other generative models such as GANs and VAEs [17],[25],[24].

4 Progressive Distillation

In the paper [20] they describe *Progressive Distillation* as a series of knowledge distillation algorithms (see Figure 3) where each algorithm takes an N-step teacher model and distills it into an $N/2$-step student model. The produced student model then becomes the teacher model for the next step in the series of distillations. Therefore, their method can take a pre-trained diffusion model and return a model with exponentially fewer steps, reducing the computational cost and resulting in faster sampling speeds.

Instead of the target for the denoising model being the original data x_0 as in [7], *Progressive Distillation* has the target, $\tilde{x}$, with the aim for one student DDIM step to match 2 teacher DDIM steps. Below is a derivation of $\tilde{x}$.

Let $t' = t - 0.5/N$ and $t'' = t - 1/N$, where N is the number of steps in the student model. In the teacher model, $x_{t''}$ is sampled by taking x_t and performing two steps of DDIM. The student model does this in one step. Let $\tilde{x}_{t''}$ be the one-step sample from the student model, let $\tilde{\alpha}_t = \sqrt{\bar{\alpha}_t}$ and $\sigma_t^2 = 1 - \alpha^2$. Then using the DDIM update rule:

$$\tilde{x}_{t''} = \tilde{\alpha}_{t''}\tilde{x} + \frac{\sigma_{t''}}{\sigma_t}(x_t - \tilde{\alpha}_t\tilde{x})$$

Setting the sample, $\tilde{x}_{t''}$, equal to $x_{t''}$, we get:

$$\begin{aligned} x_{t''} &= \tilde{\alpha}_{t''}\tilde{x} + \frac{\sigma_{t''}}{\sigma_t}(x_t - \tilde{\alpha}_t\tilde{x}) \\ &= \left(\tilde{\alpha}_{t''} - \frac{\sigma_{t''}}{\sigma_t}\tilde{\alpha}_t\right)\tilde{x} + \frac{\sigma_{t''}}{\sigma_t}x_t \\ \tilde{x} &= \frac{x_{t''} - \frac{\sigma_{t''}}{\sigma_t}x_t}{\tilde{\alpha}_{t''} - \frac{\sigma_{t''}}{\sigma_t}\tilde{\alpha}_t} \end{aligned}$$

In [20] they use a cosine noise schedule $\tilde{\alpha}_t = \cos(0.5\pi t)$ similar to that in [18] and variance such that the log signal-to-noise ratio (SNR) , $\lambda_t = \log[\tilde{\alpha}_t^2/\sigma_t^2]$, decreases monotonically with t. Additionally, the L_{simple} training objective outlined above is not suitable for distillation. For it to work, the implied prediction of $\hat{x}_\theta(x_t) \approx x_0$ needs to remain stable as λ_t varies. Therefore, [20] explores two alternative

training objectives: Truncated SNR weighting (9) and SNR+1 weighting (10):

$$L_\theta = \max\left(\|x_0 - \hat{x}_t\|_2^2, \|\epsilon - \hat{\epsilon}_t\|_2^2\right) = \max\left(\frac{\tilde{\alpha}_t^2}{\sigma_t^2}, 1\right)\|x_0 - \hat{x}_t\|_2^2 \tag{9}$$

$$L_\theta = \|v_t - \hat{v}_t\|_2^2 = \left(1 + \frac{\tilde{\alpha}_t^2}{\sigma_t^2}\right)\|x_0 - \hat{x}_t\|_2^2 \tag{10}$$

where $\hat{x}_t = \sigma_t^2 \tilde{x}_\theta\left(x_t\right) + \tilde{\alpha}_t\left(x_t - \sigma_t \tilde{\epsilon}_\theta\left(x_t\right)\right)$ or $\hat{x}_t = \tilde{\alpha}_t x_t - \sigma_t \hat{v}_\theta(x_t)$ and $v_t \equiv \tilde{\alpha}_t \epsilon - \sigma_t x_0$.

Algorithm 1 Standard Diffusion Training

Require: Model $\hat{x}_\theta(x_t)$ to be trained
Require: Data set $\mathcal{D}$
Require: Loss weight function $w(t)$

while not converged **do**
- $x \sim \mathcal{D}$ ▷ Sample data
- $t \sim U[0,1]$ ▷ Sample time
- $\epsilon \sim \mathcal{N}(0, I)$ ▷ Sample noise
- $x_t = \tilde{\alpha}_t x + \sigma_t \epsilon$ ▷ Add noise to data
- $\tilde{x} = x$ ▷ Clean data is target for $\hat{x}$
- $\lambda_t = \log[\tilde{\alpha}_t^2/\sigma_t^2]$ ▷ log-SNR
- $L_\theta = w(\lambda_t)\|\tilde{x} - \hat{x}_\theta(x_t)\|_2^2$ ▷ Loss
- $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ ▷ Optimisation

end while

Algorithm 2 Progressive Distillation

Require: Trained teacher model $\hat{x}_\eta(x_t)$
Require: Data set $\mathcal{D}$
Require: Loss weight function $w(t)$
Require: Student sampling steps N

for K iterations **do**
- $\theta \leftarrow \eta$ ▷ Init student from teacher

while not converged **do**
- $x \sim \mathcal{D}$
- $t = i/N, \quad i \sim Cat[1, 2, \ldots, N]$
- $\epsilon \sim \mathcal{N}(0, I)$
- $x_t = \tilde{\alpha}_t x + \sigma_t \epsilon$
- `# 2 steps of DDIM with teacher`
- $t' = t - 0.5/N, \quad t'' = t - 1/N$
- $x_{t'} = \tilde{\alpha}_{t'} \hat{x}_\eta(x_t) + \frac{\sigma_{t'}}{\sigma_t}(x_t - \tilde{\alpha}_t \hat{x}_\eta(x_t))$
- $x_{t''} = \tilde{\alpha}_{t''} \hat{x}_\eta(x_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(x_{t'} - \tilde{\alpha}_{t'} \hat{x}_\eta(x_{t'}))$
- $\tilde{x} = (x_{t''} - \frac{\sigma_{t''}}{\sigma_t} x_t)/(\tilde{\alpha}_{t''} - \frac{\sigma_{t''}}{\sigma_t}\tilde{\alpha}_t)$
- $\lambda_t = \log[\tilde{\alpha}_t^2/\sigma_t^2]$
- $L_\theta = w(\lambda_t)\,|\tilde{x} - \hat{x}_\theta(x_t)|_2^2$
- $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$

end while

- $\eta \leftarrow \theta$ ▷ Student becomes next teacher
- $N \leftarrow N/2$ ▷ Halve number of sampling steps

end for

Figure 2: Standard Diffusion and Progressive Distillation Algorithms based on those in [20]. Green highlight shows where the algorithms differ

5 Experimental Results

In the Literature In [20] they used the Cifar-10 dataset as a benchmark to compare *Progressive Distillation* with other methods using the Fréchet inception distance (FID) [6]. Table 1 demonstrates that *Progressive Distillation* achieves competitive FID scores with other Diffusion Models at much fewer models evaluations.

Table 1: Comparisons of *Progressive Distillation* [20] with other models on Cifar-10 dataset

Method	Model evaluations	FID
Progressive Distillation [20]	1	9.12
	2	4.51
	4	3.00
	8	2.57
Knowledge disstilation [16]	1	9.36
DDIM [22]	10	13.36
	20	6.84
	50	4.67
	100	4.16
IDDPM respacing [18] reimplementation by [20]	25	7.53
	50	4.99
Dynamic step-size extrapolation + VP-deep [9]	48	82.42
	151	2.73
	180	2.44
	274	2.60
	330	2.56
FastDPM [12]	10	9.90
	20	5.05
	50	3.20
	100	2.86
LSGM [24]	138	2.10

For the CIFAR-10 dataset they [20] started the progressive distillation with a teacher model with 8192 steps and used 50k parameter updates for each successive iteration of the distillation process. They used a UNet architecture with a fixed number of channels at resolution 256. The model internally downsampled the data twice, first to 16x16 and then to 8x8 and for each resolution the model applied 3 residual blocks [4]. They used single-headed attention for the model at the 16x16 and 8x8 resolutions. A dropout of 0.2 was used for training the original model, and no drop-out was used for the distillation process. As described in Chapter 4, the target for each student algorithm is the output of the teacher model instead of the original data.

My contribution The paper [20] used datasets Cifar-10, ImageNet, LSUN bedrooms and LSUN Church-Outdoor. However, they did not use datasets with a focus on human faces, so I have decided to also analyse the *Progressive Distillation* model on the 256x256 CelebA-HQ dataset. Due to computational cost, I decided to show qualitative results instead of the quantitative FID score.

Using the repository [8] which is a Pytorch implementation of the code for the *Progressive Distillation* paper [20], I made adjustments to return images for the original model trained with 1024 and 128 time-steps and to return images for the distilled model at 128 steps and 8 steps. The results shown in Figure 3 are non-cherry-picked. The quality of samples from the distilled model at 128 steps is similar to that of the original model at 1024 steps and noticeably better than the quality of samples from the original model at 128 steps.

Figure 3: *Progressive Distillation* on CelebA-HQ. Original model at (a) 1024 steps and (b) 128 steps. Distilled model at (c) 128 steps and (d) 8 steps

6 Conclusion

Diffusion Models have become a popular choice of generative model for their ability to produce high quality samples for a wide range of applications. However, they have the disadvantage of slow sampling speed. In this paper I outlined several methods aimed to improve the sampling speed of Diffusion Models and went into further detail on the technique, *Progressive Distillation*. I also applied the knowledge distillation model to the CelebA-HQ dataset, showing how *Progressive Distillation* can achieve similar sample quality with exponentially fewer steps, resulting in a significantly faster Diffusion Model.

References

[1] N. Chen et al. *WaveGrad: Estimating Gradients for Waveform Generation*. 2020. arXiv: 2009.00713.

[2] R. T. Q. Chen et al. *Neural Ordinary Differential Equations*. 2018. arXiv: 1806.07366.

[3] H. Chung, B. Sim, and J. C. Ye. *Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction*. 2021. arXiv: 2112.05146.

[4] P. Dhariwal and A. Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. arXiv: 2105.05233.

[5] G. Franzese et al. *How Much Is Enough? A Study on Diffusion Times in Score-based Generative Models*. 2022. arXiv: 2206.05173.

[6] M. Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2017. arXiv: 1706.08500.

[7] J. Ho, A. Jain, and P. Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239.

[8] V. Hramchenko. *PyTorch Implementation of "Progressive Distillation for Fast Sampling of Diffusion Models(v-Diffusion)"*. 2022. URL: https://github.com/Hramchenko/diffusion_distiller.

[9] A. Jolicoeur-Martineau et al. *Gotta Go Fast When Generating Data with Score-Based Models*. 2021. arXiv: 2105.14080.

[10] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. Version 1. 2013. arXiv: 1312.6114.

[11] D. P. Kingma et al. *Variational Diffusion Models*. 2021. arXiv: 2107.00630.

[12] Z. Kong and W. Ping. *On Fast Sampling of Diffusion Probabilistic Models*. 2021. arXiv: 2106.00132.

[13] Z. Kong et al. *DiffWave: A Versatile Diffusion Model for Audio Synthesis*. 2020. arXiv: 2009.09761.

[14] L. Liu et al. *Pseudo Numerical Methods for Diffusion Models on Manifolds*. 2022. arXiv: 2202.09778.

[15] C. Lu et al. *DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps*. 2022. arXiv: 2206.00927.

[16] E. Luhman and T. Luhman. *Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed*. 2021. arXiv: 2101.02388.

[17] Z. Lyu et al. *Accelerating Diffusion Models via Early Stop of the Diffusion Process*. 2022. arXiv: 2205.12524.

[18] A. Nichol and P. Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672.

[19] A. Ramesh et al. *Zero-Shot Text-to-Image Generation*. 2021. arXiv: 2102.12092.

[20] T. Salimans and J. Ho. *Progressive Distillation for Fast Sampling of Diffusion Models*. 2022. arXiv: 2202.00512.

[21] J. Sohl-Dickstein et al. *Deep Unsupervised Learning Using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585.

[22] J. Song, C. Meng, and S. Ermon. *Denoising Diffusion Implicit Models*. 2020. arXiv: 2010.02502.

[23] Y. Song et al. *Maximum Likelihood Training of Score-Based Diffusion Models*. 2021. arXiv: 2101.09258.

[24] A. Vahdat, K. Kreis, and J. Kautz. *Score-Based Generative Modeling in Latent Space*. 2021. arXiv: 2106.05931.

[25] Z. Xiao, K. Kreis, and A. Vahdat. *Tackling the Generative Learning Trilemma with Denoising Diffusion GANs*. 2021. arXiv: 2112.07804.

[26] Q. Zhang and Y. Chen. *Fast Sampling of Diffusion Models with Exponential Integrator*. 2022. arXiv: 2204.13902.

[27] H. Zheng et al. *Truncated Diffusion Probabilistic Models and Diffusion-based Adversarial Auto-Encoders*. 2022. arXiv: 2202.09671.