Generating Images using DDPM and DDIM

nvzf61

Abstract

This paper summarises the theory and mathematical background of two related models: Denoising Diffusion Probabilistic Models (DDPM) and Denoising Diffusion Implicit Models (DDIM). We also provide generated samples and interpolations, trained on the CIFAR-10 (32x32) and STL-10 (96x96) datasets, to support the comparison of our two models.

1 Methodology

1.1 Overview

Both models follow the same overall structure: Gaussian noise is repeatedly added to a real sample such that it tends towards true Gaussian noise. The models are then able to learn a reverse (generative) process to produce true samples from Gaussian noise inputs.

Denoising Diffusion Probabilistic Models (DDPM) [1] define both forward and reverse processes as Markov chains. Denoising Diffusion Implicit Models (DDIM) [2] instead generalise this forward process as non-Markovian whilst retaining the same training objective. As such, the corresponding generative process is also deterministic.

DDIMs have three benefits over DDPMs. First, the reverse process can be defined to require fewer generative steps, increasing the speed of sampling without compromising sample quality. Second, since the generative process is determinsitic, DDIMs have "consistency": given the same initial Gaussian noise input, generated samples with Markov chains of various lengths will retain similar high-level features. Third, due to consistency, we can perform semantically meaningful interpolation.

1.2 Denoising Diffusion Probabilistic Models

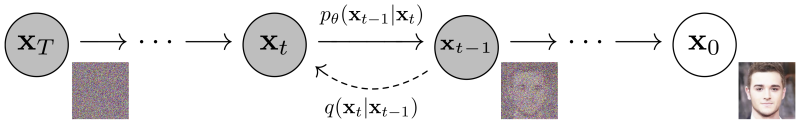


Figure 1: Forward and reverse processes. Image from [1]

Given an image sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, we define a forward diffusion process q which adds Gaussian noise to the sample for $T = 1000$ steps with variance increasing linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad where \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

Given sufficiently large T, $\mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution. Hence, if we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we can produce a true sample from a Gaussian noise input $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, since $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ depends on the entire data distribution, we approximate it as:

$$p_\theta(\mathbf{x}_{0:T}) \coloneqq p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2}$$

$$where \quad p(\mathbf{x}_T) \coloneqq \mathcal{N}(\mathbf{x}; 0, \mathbf{I}) \quad and \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{3}$$

For brevity, we summarise that we can sample $\mathbf{x}_t$ at an arbitrary timestep and skip the derivation of the loss function, concluding by stating the simplified loss function:

$$L_{simple} \coloneqq \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}[\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \|^2] \quad where \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4}$$

1.3 Denoising Diffusion Implicit Models

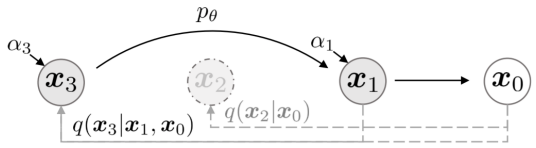


Figure 2: Accelerated generation for $\tau = [1, 3]$. Image from [2]

We define a forward process that is non-Markovian since each $\mathbf{x}_t$ could depend on both $\mathbf{x}_{t-1}$ and $\mathbf{x}_0$. The magnitude of σ controls how stochastic the forward process is; when $\sigma = 0$, the forward process becomes deterministic.

$$q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) \coloneqq q_\sigma(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \tag{5}$$

$$where \quad q_\sigma(\mathbf{x}_T|\mathbf{x}_0) \coloneqq \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1-\alpha_T)\mathbf{I}) \tag{6}$$

$$and \quad q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \coloneqq \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\alpha_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\mathbf{I}\right) \tag{7}$$

We define the corresponding non-Markovian reverse process. Given a noisy image $\mathbf{x}_t$, we first make a prediction of the corresponding $\mathbf{x}_0$, and then use it to obtain a sample $\mathbf{x}_{t-1}$ through the forward process defined above.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \coloneqq \begin{cases} \mathcal{N}(f_\theta(\mathbf{x}_1, 1), \sigma_1^2\mathbf{I}) & if\ t = 1 \\ q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, f_\theta(\mathbf{x}_t, t)) & otherwise \end{cases} \tag{8}$$

$$where \quad f_\theta(\mathbf{x}_t, t) \coloneqq (\mathbf{x}_t - \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))/\sqrt{\alpha_t} \tag{9}$$

We conclude by defining the objective:

$$J_\sigma(\boldsymbol{\epsilon}_\theta) \coloneqq \mathbb{E}_{\mathbf{x}_{0:T}}[log\ q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) - log\ p_\theta(\mathbf{x}_{0:T})] \tag{10}$$

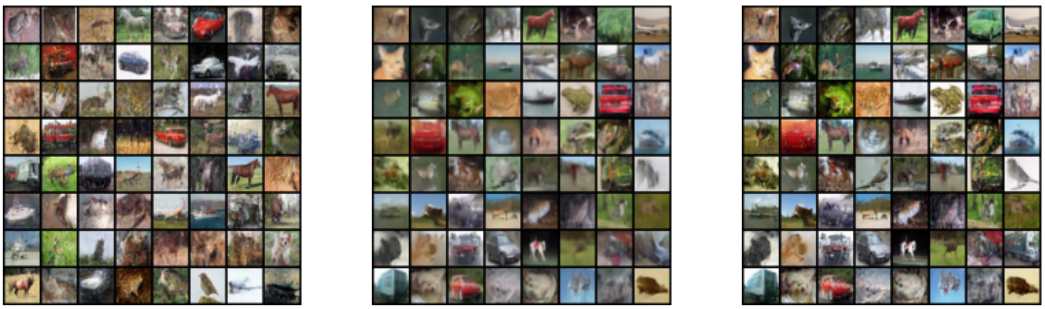
It has been shown by Song et. al [2] that the objective function J_σ from DDIM is identical to L_{simple} from DDPM. Since L_{simple} does not depend on the forward process as long as $q_\sigma(\mathbf{x}_t|\mathbf{x}_0)$ is fixed, we may define the forward process on a subset τ of T steps. The corresponding generative process is also dependent on this subset. Therefore, we can train a model with an arbitrary number of forward steps, but only sample a subset of them in the generative process, greatly increasing the sampling speed.

2 Results

We train our models on CIFAR-10 and STL-10 for 672 and 642 epochs respectively using the source code provided by labml.ai [3] for DDPM and Song et. al [2] for DDIM as a basis. We make a lot of alterations to simplify the code and to allow both models to train and sample using the same functions. We restrict STL-10 to the train and test sets only to reduce the training time on colab. To satisfy the restrictions imposed by colab's GPU, we use a batch size of 64 for CIFAR-10 and 16 for STL-10.

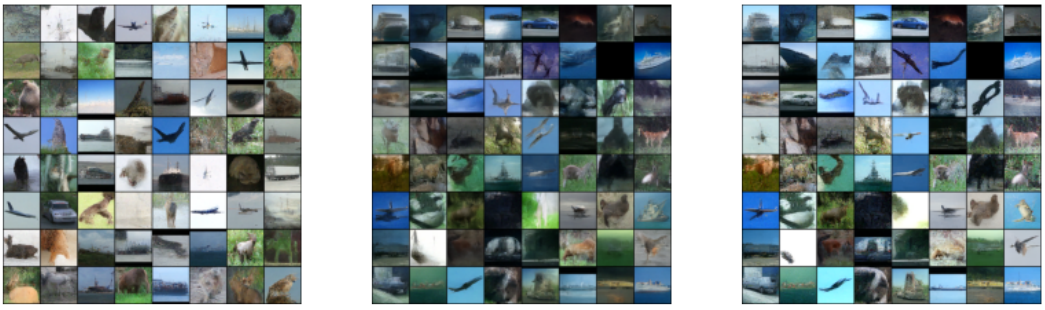
2.1 Samples

From the samples below, we notice that we can easily generate realistic images using our models. By comparing our DDIM samples for varying τ, we can clearly see that we can generate comparable images whilst severely reducing the number of sampling steps, only losing out on some definition and depth of colour. To highlight the time saved, it took 15 minutes to sample figure 4a but only 15 seconds to sample 4b with only a small batch of sixty-four 92x92 images. If we had to generate a larger amount of images at higher resolution, the difference would be greater still. We also observe the consistency property of DDIMs as the generated images with varying τ retain similar high-level features.



(a) DDPM ($T = 1000$) (b) DDIM ($\tau = 10$) (c) DDIM ($\tau = 1000$)

Figure 3: Non cherry-picked samples from CIFAR-10



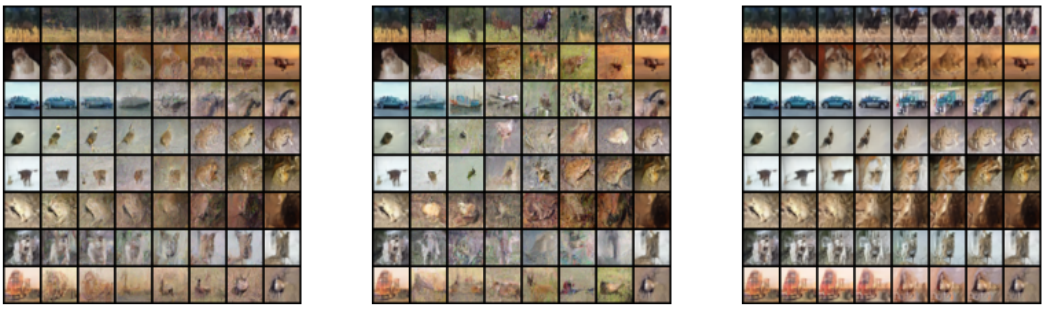
(a) DDPM ($T = 1000$) (b) DDIM ($\tau = 10$) (c) DDIM ($\tau = 1000$)

Figure 4: Non cherry-picked samples from STL-10

2.2 Interpolations

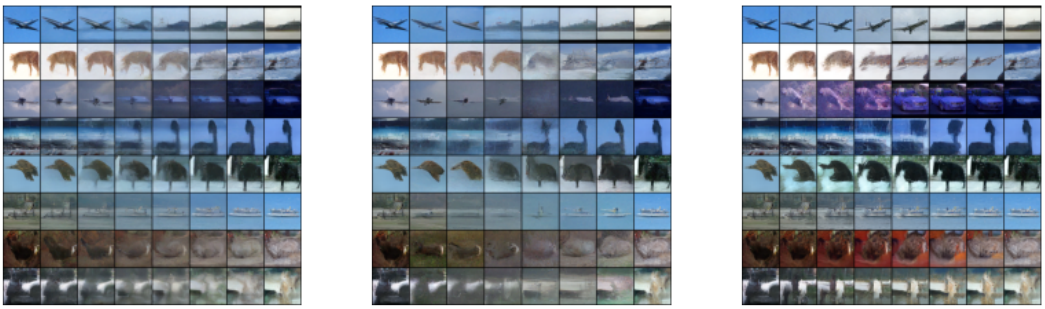
For DDPM, given two images in the image space $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and $\mathbf{x}'_0 \sim q(\mathbf{x}'_0)$, using the forward process we get $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ and $\mathbf{x}'_t \sim q(\mathbf{x}'_t|\mathbf{x}'_0)$. We can then decode the linearly interpolated latents $\bar{\mathbf{x}}_t = (1-\lambda)\mathbf{x}_t + \lambda\mathbf{x}'_t$ for varying λ back into the image space using the reverse process $\bar{\mathbf{x}}_0 \sim p(\mathbf{x}_0|\bar{\mathbf{x}}_t)$. As such, we can choose how coarse our interpolations are by varying t, where larger t results in more varied interpolations, as shown by the middle interpolations for $t = 250$ looking more distinct from their neighbours than for $t = 100$.

Since DDIMs have consistency, we can perform spherical linear interpolation on two Gaussian noise inputs, and then decode the latents to produce semantically meaningful interpolations. This is in contrast to DDPM, which has to interpolate closer to the image space (smaller t) due to its stochastic nature. We cam observe the semantically meaningful interpolations in comparison to DDPM, by noticing the middle interpolations for $\tau = 100$ look more like realistic generated images than in $t = 100$ (e.g. the car in the third row of 5c).



(a) DDPM ($t = 100$) (b) DDPM ($t = 250$) (c) DDIM ($\tau = 100$)

Figure 5: Interpolations between 8 pairs of generated CIFAR-10 samples



(a) DDPM ($t = 100$) (b) DDPM ($t = 250$) (c) DDIM ($\tau = 100$)

Figure 6: Interpolations between 8 pairs of generated STL-10 samples

2.3 Diffusion Sequences

We briefly show the reverse process of our models used to generate an image.

Figure 7: CIFAR-10 and STL-10 diffusion sequence

2.4 Cherry-Picked

We show cherry-picked samples that highlight the best images our models have generated. Our models more easily generate man-made objects such as cars over more complex images such as animals. This would be resolved given more training time.

(a) CIFAR-10 (b) STL-10

Figure 8: Cherry picked samples from both DDIM and DDPM

3 Limitations

The glaring issue of DDPMs is the number of steps required to sample an image since it becomes impractical when sampling a large numbers of images at high resolution. This is amended via DDIMs, whilst acquiring the added benefits of consistency and semantically meaningful interpolations.

The simplest and most obvious improvement would be to train for more epochs given a more powerful machine. Similarly, we could train on the whole of the STL-10 dataset and increase the number of steps from $T = 1000$ to improve image quality. This would be impractical for DDPMs as it will only increase the sampling time, however, we can retain sampling efficiency using DDIMs by only sampling on a subset of steps in the generative process.

We could further improve our paper by the introduction and comparison with Improved DDPMs [4], which also increase sampling speed and achieve better log-likelihoods with little impact on sample quality.

Bonuses

This submission has a total bonus of +4 marks as it is trained on STL-10 at 96x96 pixels.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2021. arXiv: 2010.02502 [cs.LG].

[3] Nipun Wijerathne Varuna Jayasiri. *labml.ai Annotated Paper Implementations*. 2020. URL: https://nn.labml.ai/.

[4] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].