Denoising Diffusion Probabilistic Models: Seminar Report

Soumava Paul

Visual Computing Master, Saarland University, 66123 Saarbrücken, Germany

Abstract. In this seminar report, we will be discussing the **Denoising Diffusion Probabilistic Models** (DDPM) paper published in Neurips 2020. We will be going over the inspiration behind the DDPM model, its overall methodology, experimental results, and finally, some of its advantages and limitations.

1 Introduction

In recent years, researchers have made remarkable progress in conditional/unconditional generation across many data modalities like audio, images, and videos with diffusion models. Some prominent examples include Stable Diffusion [8], Imagen [10], GLIDE [5] etc. In this report, however, we will be discussing one of the earlier works - Denoising Diffusion Probabilistic Models [3] or **DDPM** for short, that proved to be one of the foundational ideas for many future works.

Prior to diffusion models, GANs [2], VAEs [4], and Flow-based models [7] have proven effective in generating high-quality samples. GANs are based on a game theoretic approach where two models, a generator and a discriminator, compete with each other to optimize a minimax objective function. However, GANs are quite difficult to train due to general instability and slow convergence. VAEs are generative models that employ neural networks and a probabilistic framework to learn data representations while optimizing a loss function that balances reconstruction accuracy and a regularization term. However, they often produce blurry reconstructions of input data, and the assumption that the latent space follows a simple Gaussian often hinders its ability to capture complex distributions. Also, both GANs and VAEs suffer from **mode collapse** where they fail to capture the full diversity of the data distribution and often produce highly similar samples. Flow-based generative models are constructed by a sequence of invertible transformations. The invertible requirement can limit the types of transformations that can be used, potentially leading to a less expressive model.

Diffusion models, on the other hand, are inspired by non-equilibrium thermodynamics introduced first by Sohl-Dickstein [12] in 2015, and then independently improved by Song et al. [13] in 2019, and Ho et al. [3] in 2020. The rest of the report is organized as follows: In Section 2, we dive into the math behind the objective function of DDPM. In Sections 3 and 4, we discuss some of the architecture details and key qualitative results presented in the paper. In Section 5, we list some drawbacks of DDPM and related future works, and finally, in Section 6, we give a conclusion.

2 Methodology

The DDPM paper showed for the first time that a diffusion probabilistic model or a “diffusion model” for short, is capable of generating high-quality samples. The idea is quite similar to Normalizing Flows, GANs or VAEs in that it converts noise from some simple distribution (like isotropic Gaussian) to a data sample. For this, we define a parameterized Markov chain that gradually destroys data by adding random noise. This fixed or predefined noising process is known as the forward diffusion process (2.1). The objective is to learn to reverse this process to recover a data sample. For this, a neural network is trained to gradually denoise an image starting from pure random noise in what is known as the reverse diffusion process (2.2). When the diffusion consists of small amounts of Gaussian noise, it is sufficient to set the sampling chain transitions to conditional Gaussians too, allowing for a particularly simple neural network parameterization (2.3).

2.1 Forward Process

Let $\mathbf{x}_0$ be a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$. The *forward diffusion process* adds small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$ where $\beta_1 < \beta_2 < \cdots < \beta_T$. This implies we can afford a larger update step when the sample gets noisier.

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

$\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger and for $T \to \infty$, $\mathbf{x}_T$ is pure isotropic Gaussian noise for an appropriately chosen schedule for β_t (Section 3). Using the reparameterization trick [4], $\mathbf{x}_t$ at any arbitrary

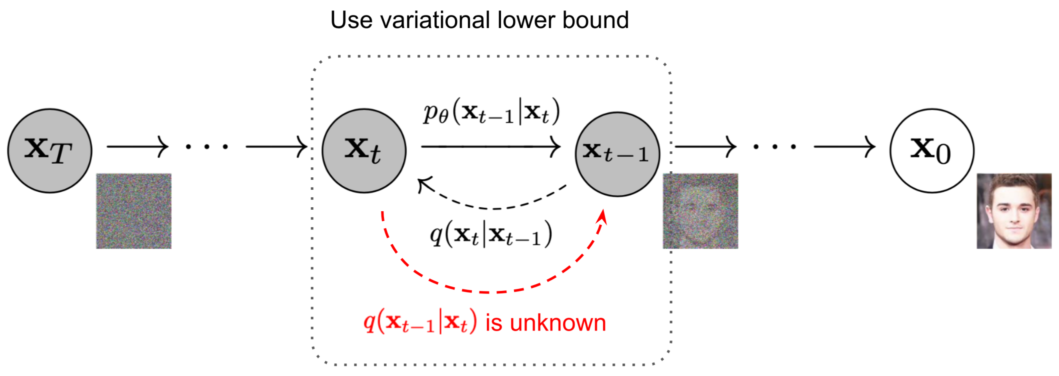

Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown

Fig. 1. The directed graphical model used in DDPM [16]

time step t can be written directly in terms of $\mathbf{x}_0$ so that we don't have to repeatedly apply q. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && ;\text{where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && ;\text{where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\
q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}$$

2.2 Reverse Process

If we can reverse the above process and sample from the conditional probabilities $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ (Figure 1), we can get a sample from $q(\mathbf{x_0})$ starting from Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, we cannot do that as that means knowing the distribution of all possible images to calculate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. So instead, the authors propose to learn a model p_θ updated by gradient descent to approximate these conditional probabilities.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

If β_t is small enough, the reverse conditional will be Gaussian as well and tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

Using Bayes' rule, one obtains ([16]):

$$\begin{aligned}
\tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\
\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0
\end{aligned}$$

Using the result in 2.1, we have $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t)$ and hence:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t) \\
&= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big)
\end{aligned}$$

The authors observe that the combination of q and p_θ can be interpreted as a VAE [4], and hence one can use the variational lower bound (or ELBO) to optimize the negative log-likelihood.

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\le -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \ge -\mathbb{E}_{q(\mathbf{x}_0)}\log p_\theta(\mathbf{x}_0)
\end{aligned}$$

The objective above can be further rewritten as a combination of KL-divergence and entropy terms (Appendix A of [3])

$$\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log\Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\cdot\frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \\
&= \mathbb{E}_q[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}$$

Every KL term above (except for L_0) compares two Gaussian distributions, and therefore they can be computed in closed form. The reverse process learns a neural network to approximate $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t,t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t))$. Hence $\boldsymbol{\mu}_\theta$ should be able to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_t\Big)$. As $\mathbf{x}_t$ is available as input at training time, one can reparameterize the Gaussian noise term instead to make it predict $\boldsymbol{\epsilon}_t$ from the input $\mathbf{x}_t$ at time step t:

$$\begin{aligned}
\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\Big) \\
\text{Thus } \mathbf{x}_{t-1} &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\Big), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t))
\end{aligned}$$

With this reparameterization, the network becomes a noise predictor rather than a mean predictor. The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$ [16]:

$$\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\Big)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar\alpha_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar\alpha_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]
\end{aligned}$$

2.3 Simplification

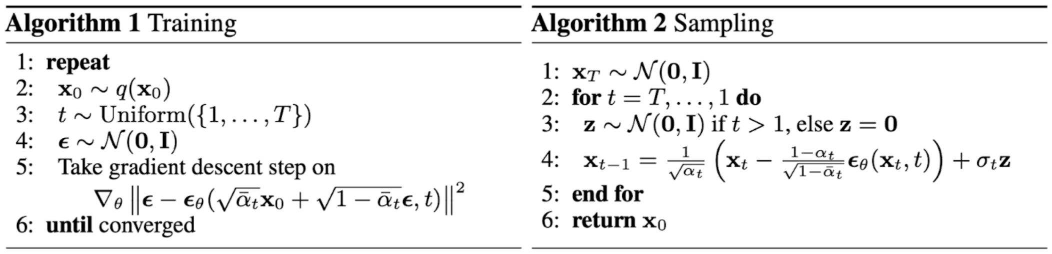
Empirically, [3] found that training the diffusion model works better with a simplified objective that ignores the weighting term in front:

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right] \end{aligned}$$

The final simple objective is:

$$L_{\text{simple}} = L_t^{\text{simple}} + C$$

where C is a constant not depending on θ. Hence, the overall algorithm is as below:



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1,\dots,T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: for $t = T,\dots,1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Fig. 2. The training and sampling algorithms in DDPM [3]

3 Experimental Details

The authors set $T = 1000$ for all their experiments to match the number of neural network evaluations needed during sampling matches with previous works [12][13]. They chose the β_t schedule from a set of constant, linear, and quadratic schedules, all constrained so that $L_T \approx 0$. In their experiments, they finally chose a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.

For modeling the reverse process, the neural network needs to take in a noised image at a particular time step and return the predicted noise. Since the predicted noise has the same resolution as the input image, the authors chose a U-Net backbone similar to an unmasked PixelCNN++ [11][9] with group normalization throughout. Parameters are shared across time and the diffusion time t is specified to the network using the Transformer sinusoidal position embedding [15].

4 Experimental Results

In Figures 3 and 4 taken from [3], we have unconditional image generation results for LSUN *256* $\times$ *256* [17] Church and Bedroom datasets, that achieves better sample quality than most generative models in literature, including class conditional models. The sample quality is measured using the FID or Frechet Inception Distance score [14]. FID compares the distribution of generated images with the distribution of real images that were used to train the generative model using the squared Wasserstein metric. DDPM also achieves state-of-the-art FID score (**3.17**) in unconditional image generation on the CIFAR10 dataset. In Figure 5, the authors present the result of the reverse process, $\hat{\mathbf{x}}_0$, while sampling from the reverse process using Algorithm 2. It shows the resulting sample quality of $\hat{\mathbf{x}}_0$ over the course of the reverse process. For more details, please refer to Section **4.3** of the main paper.

5 Limitations and Follow-up Work

DDPM was one of the first papers to show promise for diffusion models in (un)conditional image generation. The DDPM formulation is, however, not free from a few drawbacks.

– The reverse denoising process can require around 1000 steps and hence can be quite slow. But, with some follow-up works like [18], this problem has been alleviated, and diffusion models can now do high-fidelity generation in as few as 10 denoising steps.

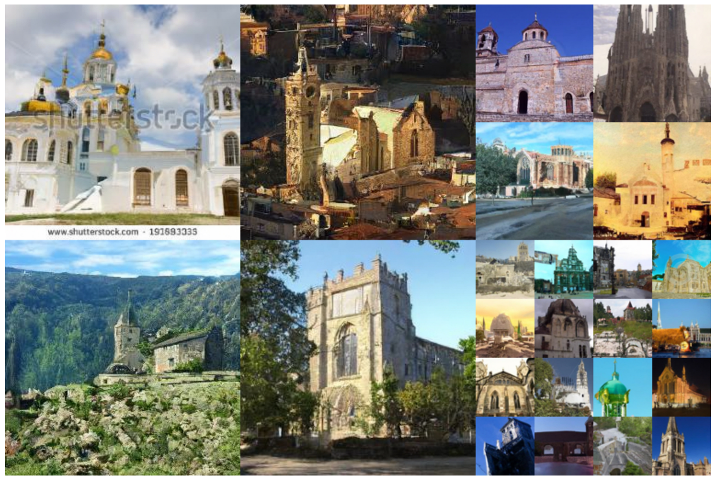


Fig. 3. LSUN Church samples. FID=7.89

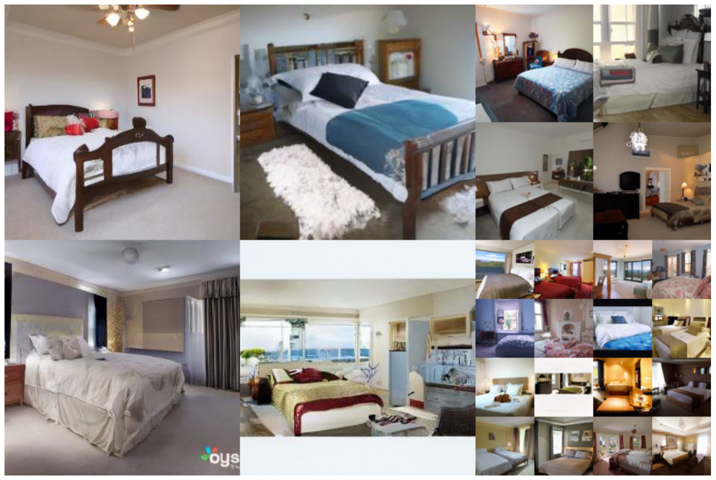


Fig. 4. LSUN Bedroom samples. FID=4.90

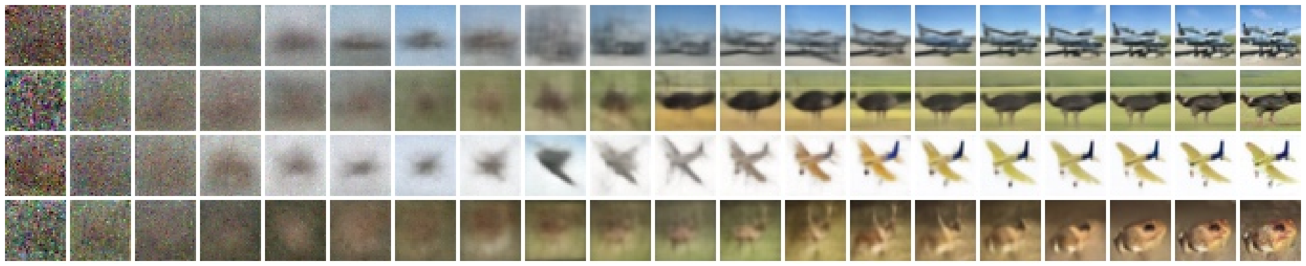


Fig. 5. Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right)
.

- In DDPM, the latents $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are of the same dimensionality as the input data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. Since, $\mathbf{x}_0$ can be high-dimensional images, the diffsuion process is quite computationally expensive. This issue was alleviated in follow-up works like [8] where the noising and denoising processes operate in the low-dimensional latent space.
- Even though the authors used a fixed variance schedule here, follow-up works like [6] have shown that using a neural network to learn the variance of the backward conditional distribution helps improve performance. Also, [6] found that using a cosine instead of a linear schedule leads to better performance.
- DDPM and many other related works emphasize upon conditional Gaussian modelling for the forward and backward processes. However, recently [1] introduced *cold diffusion* which doesn't require Gaussian noise (or any randomness) during training or testing, and in fact, *generalized diffusion models* can be constructed with arbitrary deterministic degradations like blurring, masking, downsampling etc.
- Like other generative models, diffusion models are also not immune to malicious applications like fake image generation. Since they are trained on unlabeled datasets, the models reflect the biases of the datasets on which they are trained. Such biases get amplified as more and more samples generated by these models spread over the internet.

6 Conclusion

In this report, we discussed the DDPM model, which is essentially a parameterized Markov chain where latent variables only depend on the previous or following timestep. We found that maximizing the likelihood of the training data as the training objective is equivalent to minimizing the variational upper bound of the negative log-likelihood. While the forward process requires only a variance schedule, the reverse process parameters are learned. The assumption that the transition distributions are Gaussians enables the KL-Divergence loss to have a closed-form solution. The authors obtain the best and most stable results with a simplified training objective that predicts the noise component of the latent variable.

References

1. Bansal, A., Borgnia, E., Chu, H.M., Li, J.S., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., Goldstein, T.: Cold diffusion: Inverting arbitrary image transforms without noise. arXiv preprint arXiv:2208.09392 (2022)
2. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS (2014), `https://api.semanticscholar.org/CorpusID:1033682`

3. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. CoRR **abs/2006.11239** (2020), `https://arxiv.org/abs/2006.11239`
4. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
5. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models (2022)
6. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. pp. 8162–8171. PMLR (2021)
7. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International conference on machine learning. pp. 1530–1538. PMLR (2015)
8. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models (2021)
9. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
10. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding (2022)
11. Salimans, T., Karpathy, A., Chen, X., Kingma, D.P.: Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. arXiv preprint arXiv:1701.05517 (2017)
12. Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics (2015)
13. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems **32** (2019)
14. Strika, L.: Fid: Fréchet inception distance. `https://strikingloo.github.io/wiki/fid` (2022), last visited August 2023
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
16. Weng, L.: What are diffusion models? `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/` (2021), last visited August 2023
17. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
18. Zhang, Q., Chen, Y.: Fast sampling of diffusion models with exponential integrator. arXiv preprint arXiv:2204.13902 (2022)