Optimizing Diffusion Models for Image Denoising

Adithya Iyer [1] **Man Shu** [1] **Rhitvik Sinha** [1]

Abstract

Diffusion Models have proved the ability to produce high quality images based on score function estimations, which in image modeling is often seen as a Gaussian noise removal process. We investigate whether this Gaussian noise removal trained model can be used for actual image denoising, or in some cases filling in missing pixels (imputation). We achieve this by conducting experiments where we feed in partially noised images to various stages of the diffusion model to observe the outputs of the de-noising behaviour. We also build a noise conditioned diffusion model to train the model to learn to de-noise and impute images.

1. Introduction

Diffusion models have recently grown in popularity for image generative tasks due to their nearly similar performance to GANs, and have resulted in their adoption to various use cases like Stable Diffusion[7]. The original DDPM [2] uses a sequential de-noising process to de-noise random Gaussian noise, and produces high quality outputs which have been proven recently to outperform GANs [1].

Diffusion models have also proved the ability to produce high quality images without the need for adversial training. One significant flaw of most diffusion models is that they need numerous iterations to create an excellent clear image, which makes diffusion models behave quite slower than GANs. The backward process for denoising diffusion probabilistic models (DDPM) approximates the reverse of the forward process which includes multiple steps. Another model Denoising Diffusion Implicit Models (DDIM) proposed by Song et al. [9] is based on non-Markovian noising process by altering reverse noise's variance compared to DDPM, which can shorten timesteps. Furthermore Nichol & Dhariwal [6] found that this model behaves well when sampling steps are less than 50.

In this paper, we want to discuss using diffusion models in the paradigm of noise removal or imputation. We do this is 2 broad ways :

1. By feeding in noised image to a regularly trained DDPM, considering class conditioned and unconditioned cases.

2. By training a diffusion model conditioned on a noisy or incomplete image.

We notice that the diffusion model noise-removal behaviour in the intermediate regions is significantly affected by the noise present in the input image. We also observe that noised-image conditioning on diffusion models works better when you remove pixels, than adding Gaussian noise.

2. Background

2.1. DDPMs

We briefly go over the formulation of DDPMs from Ho et al. [2]. Denoising diffusion probabilistic models (DDPMs) are latent variable models of the form below, where x_1,..., x_T are latents of the same dimensionality as x_0 given a data distribution $q(x_0)$.

$$p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T} \tag{1}$$

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta^{(t)}(x_{t-1}|x_t) \tag{2}$$

To fit the data distribution, optimize the variational lower bound (VLB) on negative likelihood:

$$\mathbb{E}\left[\log p_\theta(x_0)\right] \leq \mathop{\mathbb{E}}_{q}\left[-\log p_\theta(x_{0:T}) + \log q(x_{0:T}|x_0)\right] =: L \tag{3}$$

DDPM fix the approximate posterior $q(x_{1:T}|x_0)$ and use the following Markov Chain with Gaussian transition according to variance schedule β_1,...,β_T with notation $\alpha_t :=$

[1]Department of Computer Science, Courant Institute of Mathematical Science, New York University, New York. Correspondence to: Adithya Iyer <ai2257@nyu.edu>, Man Shu <ms12677@nyu.edu>, Rhitvik Sinha <rs8438@nyu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Algorithm 1 Training

repeat
  $x_0 \sim q(x_0)$
  $t \sim \text{Uniform}(\{1, ..., T\})$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  Take gradient descent step on

$$\nabla_\theta ||\epsilon - \epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon, t)||^2$$

until converged

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, ..., 1$ **do**
  $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $z = 0$
  $x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_\theta(x_t, t)\Big) + \sigma_t z$
end for
return x_0

$\prod_{s=1}^{t}(1-\beta_s)$:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{4}$$

$$q(x_t|x_{t-1}) := \mathcal{N}\Big(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}x_{t-1}, \Big(1 - \frac{\alpha_t}{\alpha_{t-1}}\Big)\mathbf{I}\Big) \tag{5}$$

The forward process has a remarkable property which is sampling x_t in closed form at any timestep:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_0, (1-\alpha_t)\mathbf{I}) \tag{6}$$

Then x_t could be written as a linear combination of x_0 with noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$:

$$x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon \tag{7}$$

We choose $p_\theta(x_{t-1}|x_t) = \mathcal{N}\Big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\Big)$ and fix $\Sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$. When setting α_T close enough to 0, $q(x_T|x_0)$ converges to a standard Gaussian for all x_0. Intuitively, we set $p_\theta(x_T) := \mathcal{N}(0, \mathbf{I})$. When the length T for the forward process is sufficiently large, the backward process will be much closer to a Gaussian, which means the generative process approximates well.

2.2. Conditioning DDPMs

Training diffusion models with class conditionality is important to obtain precise images of particular classes we want. Nichol & Dhariwal [6] train a classifier and use classifier gradients to train DDPM to fit a class. Although we agree with this approach, training a classifier seems unnecessary. The conditional model in this paper is based on Classifier-Free Diffusion Guidance [3] which includes a linear combination of the score estimates from conditional diffusion models $p_\theta(z, c)$ and jointly trained unconditional diffusion models $p_\theta(z)$. Conditional diffusion model involves two embedding timestep t and context c. Recall Baye's rule:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{8}$$

$$\nabla_x \log p(y|x) = \nabla_x \log p(x|y) - \nabla_x \log p(x) \tag{9}$$

Then the formula with classifier guidance:

$$\nabla_x \log p_\gamma(x|y) = (1-\gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x|y) \tag{10}$$

Define $\psi(z_t)$ for noise z_t at timestep t, and $\psi(z_t, c)$ for context c and timestep t. Based on this, the formula for the score estimator with weight $\omega \geq 0$:

$$\psi_t = (1+\omega)\psi(z_t, c) + \omega\psi(z_t) \tag{11}$$

Imputation or image transformation as a concept is not new to diffusion models. Saharia et al. [8] use diffusion models for in-painting, colorization and super resolution. VAEs particularly have been studied for image denoising, such as Pascal et al. [10], or image imputation as in Mattei & Frellsen [5]. Authors have also explored conditional models for sematic mixing [4].

3. Methods

We run 3 levels of experiments on MNIST and Fashion-MNIST to see how diffusion models perform in a de-noising context.

3.1. Adding noisy images to intermediate locations in the diffusion chain

We apply Gaussian noise to clean images, and feed these images to various time-steps in the diffusion process. The Gaussian noise scheduler is chosen to be the same as the one the models is trained on, to enable coherent inference of results. We apply noise in intervals of 30% and 60%, and feed this noisy image into the diffusion chain at various time-steps (T = 100%, 50%, 25%, 100% meaning the start of the noise removal and smaller percentages implying less noisier images.).

Furthermore, we also test the effect of running multiple cycles of the diffusion chain on the image. Basically, we get a de-noised output, and pass it again into the the model, multiple times.

3.2. Effect of conditioning the Neural Network on a noisy image

In Section 3.1, we largely train a regular and a class conditioned DDPM, and use it directly to denoise. In essense,

we are using the denoising aspect of a diffusion models as a feature and applying it for an alternate purpose. In this subsection, we specifically train a conditional DDPM for de-noising purpose. We do so by adding additional conditioning to a single link in the neural network.

Our traditional UNet, which is the building block of the neural network, in essence predicts $p(X_{t-1}|X_t, t, c)$, where t is the time-step and c is the context, or the class label. We add another condition to the UNet, namely the noised input image. This makes the UNet at timestep T become:

$$UNet_t = p(X_{t-1}|X_t, t, c, X_{NoisedInput}) \tag{12}$$

In a more score functional notation, this gives us the conditional score on the noised image which we want to denoise.

$$-\nabla_x log p(x_{t-1}|x_t, t, x, x_{NoisedInput}) \tag{13}$$

We modify the UNet which we use in our model by simply concatenating an additional channel to the input to each UNet. This additional channel contains the image which we want to denoise.

In an intuitive sense, we are showing a diffusion model a blurry image as a reference, and then training it to learn how to paint a de-noised image.

3.3. Classifier free conditioning with time-step governed weights

We noticed that the method in Section 3.2 worked particularly well where the conditional signal was strong. For instance, removing a fraction of pixels from an image still leaves the image with enough interpret-able latent space information to help the model train, since a majority of the pixels are from the population distribution p(x).

This is untrue for Gaussian noised images, since the entire image is from a different unseen distribution, that is, $p(x|x_{noisy})$ is not easily learn-able. We also observed this while conducting experiments, where essentially the model was learning very well to condition on the noised image, but was unable to learn the denoised distribution once it has learnt the noisy distribution.

Hence we suggest an alternative solution to this problem, where we essentially condition the model on the noisy input image less once it has learnt the noisy distribution. Basically, if the image X_t is very close to the noisy input $X_{NoisedInput}$, we condition the model less on the noisy image. We do this probabilistic-ally by the following trick, where we define a probability of conditioning on each time-step.

$$p_{conditioning}(t) = 1 - e^{\frac{||X_t - X_{NoisedImage}||_2}{T}} \tag{14}$$

What this basically means is that as the image in the diffusion chain is closer to the noised input image, we should decrease the classifier guidance and increase the score gradients from the actual image distribution.

In practice, we achieve this by masking the conditioning layer in the UNet with a probability $p_{conditioning}$. So the model significantly learns from the conditional distribution at the start of the denoising process, but learns less from the conditioning as it reaches the later time-steps in the reverse process. We notice that setting T = 1 seems to do a good job in modulating $p_{conditioning}$ between 0.2 and 0.8.

4. Experiments

We first run experiments by adding various levels of noisy images to various timesteps of the diffusion process.

4.1. Conditional - Adding varied noise to various timesteps

4.1.1. Adding X% noise to the timestep which expects exactly X% noise

We added a 30% noised image to an image, and fed it into the 70% time-step of the reverse diffusion process (where it expects 30% noise). As expected, the model worked perfectly for all images, which imply that the diffusion model has learnt the intermediate distributions perfectly, as can be seen in Fig. 1.

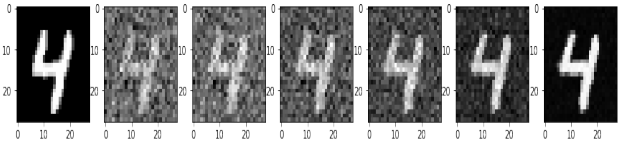

Figure 1. (From left to right: Original Image, Noisy Image Input, all diffusion time-steps till final generated image.) **Adding right amount of noise to the right location.**

4.1.2. Adding X% noise to the timestep which expects less than X% noise

We added a 30% noised image to an image, and fed it into the 75% time-step of the reverse diffusion process (where it expects 25% noise). We observe that the model does denoise, but does not give us the sharp clarity as it did in the previous image. This can be observed in Fig.2.

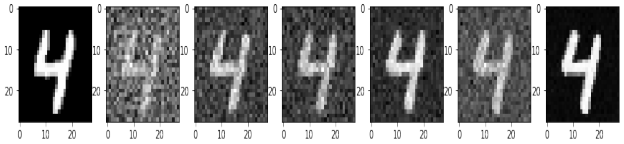

Figure 2. (From left to right: Original Image, Noisy Image Input, all diffusion time-steps till final generated image.) **Adding more noise to the time-step which expects less noise.**

4.1.3. Adding X% noise to the first time-step only, and running repeatedly (40 times)

We added a 30% noised image to an image, and fed it into the last time-step of the reverse diffusion process (T =0, where it expects a nearly noiseless image). We observe some de-noising, but largely as a consequence of Gaussian smoothing due to repeated loops of $UNet_{t_0}$. This can be observed in Fig.3.

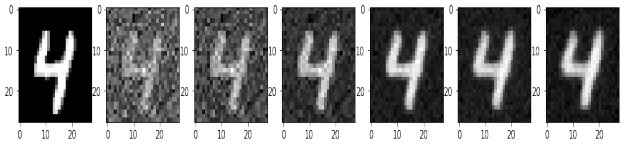

Figure 3. (From left to right: Original Image, Noisy Image Input, all diffusion time-steps till final generated image.) **Running only the last UNet (T = 1), 40 times to check effect of repetition.**

4.1.4. Adding X% noise to the time-step which expects more than X% noise

We added a 30% noised image to an image, and fed it into the first time-step of the reverse diffusion process (T = 200th step or 100%, where it expects pure noise). We observe severe addition of noise initially, followed by subsequent denoising. The important observation here is that the final generated image is not the exact denoised image, but a different sample from the same class. This can be observed in Fig.4.

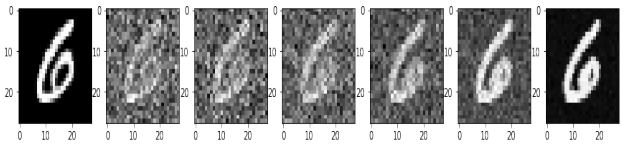

Figure 4. (From left to right: Original Image, Noisy Image Input, all diffusion time-steps till final generated image.) **Adding a less noisy image to a time-step which expects more noisy inputs- in a conditional DDPM.**

Performing the same experiment to FASHION-MNIST via an unconditional diffusion model gives even greater clarity on the fact that the diffusion models ends up de-noising to a completely different sample from the image distribution. This can be observed in Fig. 5, where an input noisy image of a pant leads to the model generating a dress.

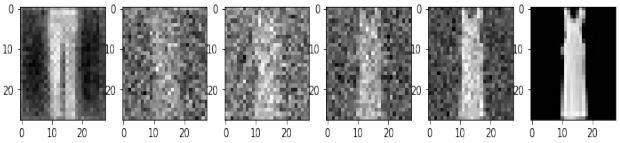

Figure 5. (From left to right: Noisy Image Input, all diffusion time-steps till final generated image.) **Adding a less noisy image to a time-step which expects more noisy inputs- in a un-conditional DDPM. Notice how the generated image is of an entirely different class.**

4.2. Image Imputation with conditional DDPM

We now train a conditional diffusion model on images with missing pixels, and force it to generate imputed outputs, as explained in Section 3.2. The model essentially learns the following distribution.

$$p(x|x_{MissingPixels}) \tag{15}$$

The model does a great job of filling in the missing pixels, which can be seen in Fig. 6

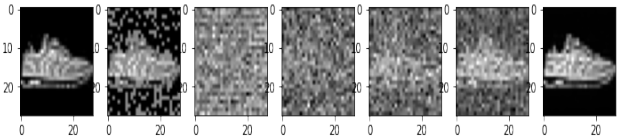

Figure 6. (From left to right: Original Image, Image Input with missing pixels, all diffusion time-steps till final generated image.) **Our conditional Diffusion model conditioned on the noisy image does a great job in pixel imputation.**

We apply the same method by conditioning on a Gaussian noised image, but the model did not seem to learn how to de-noise it, which resulted in us engineering the probabilistic method explained in section 3.3. The results of our probabilistic trick are shown in the next subsection.

4.3. Image Denoising with Conditional DDPM, probabilistic weights

We now train a conditional diffusion model on images with random Gaussian noise, and force it to generate de-noised outputs, as explained in Section 3.3. The model essentially learns the following distribution.

$$p(x|x_{NoisyImage}) \tag{16}$$

The model does a great job of denoising because of the probabilistic conditioning we apply which affects the later layers of the diffusion reverse process. The outputs can be seen in Fig. 7.

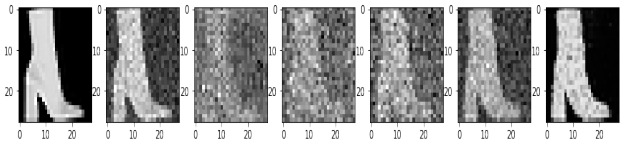

Figure 7. (From left to right: Original Image, Noisy Image Input, all diffusion time-steps till final generated image.) **Our conditional Diffusion model conditioned on the noisy image does a great job in denoising.**

5. Conclusion

We can observe that the de-noising behaviour of DDPMs works in a temporal order, where each link in the diffusion reverse process expects a certain noise level, and providing a less noisy image for de-noising leads to unnecessary noise addition and improper score function. This is clearly observable when we add a less noisy image to the first time-step of the DDPM- we get a completely different class as the output. The effect of giving a more noisy image to later in the diffusion chain leads to better outcomes, but is still not completely sufficient.

All of these mean that using a regularly trained diffusion model for de-noising works well only if we can estimate the **exact** noise level of the image, and then feed in the image at the right place in the diffusion process. This is impractical in a real world setting, since knowing the exact magnitude, and type of noise is difficult. We could apply Fourier based transforms methods to get an approximation of the amount of high frequency noise, but it is an inefficient way to get around the problem. Furthermore, Fourier transform methods for noise estimation would only work well for certain classes of noise such as Gaussian, and would fail with pixel removal type of noise.

Thus building a model **conditioned on the noise** is of importance, so the diffusion model can learn to generate complete images based on a memory of the noisy input. We prove that this methods works brilliantly when we remove pixels, and can be made to work with Gaussian noised images if we apply our probabilistic trick. Thus, we suggest that for de-noising purposes, a conditional diffusion model is the right way to go.

To confirm our assertion, we suggest applying the techniques we explain here to larger datasets like CIFAR-10 or Imagenet-Net as the obvious next step. The model will require certain modifications to account for additional channels, but the informational flow and observations on MNIST should transfer to these larger datasets.

References

[1] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[3] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *arXiv preprint arXiv:2207.12598* (2022).

[4] Jun Hao Liew et al. "MagicMix: Semantic Mixing with Diffusion Models". In: *arXiv preprint arXiv:2210.16056* (2022).

[5] Pierre-Alexandre Mattei and Jes Frellsen. "MIWAE: Deep generative modelling and imputation of incomplete data sets". In: *International conference on machine learning*. PMLR. 2019, pp. 4413–4423.

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

[7] Robin Rombach et al. "High-Resolution Image Synthesis With Latent Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695.

[8] Chitwan Saharia et al. "Palette: Image-to-image diffusion models". In: *ACM SIGGRAPH 2022 Conference Proceedings*. 2022, pp. 1–10.

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[10] Pascal Vincent. "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7 (2011), pp. 1661–1674.