Application of Diffusion Models to Inverse Imaging Problems

Isaac Criddle

Abstract—Inverse problems in imaging often require reconstructing data from incomplete information. Since diffusion models implicitly learn image distributions, they can be leveraged to reconstruct such missing data. To apply a diffusion model to an inverse problem, the reverse diffusion process must be altered so as to be conditioned on given data. In this paper, we compare three contemporary techniques for conditioning diffusion models on given data for inverse imaging problems: SDEdit, ScoreALD, and DPS. These approaches are evaluated on inpainting and deblurring tasks. This paper shows that generative priors are a powerful tool for inverse imaging problems and a promising direction for future research.

Index Terms—Computational Photography

1 Introduction

In image processing and computational photography, it is often desirable to reconstruct a signal from samples in order to interpolate or simulate data. Many processes beyond our control can remove or hide information in images, including motion blur, diffraction, shot noise, and many others. Recovering the missing information by extrapolating from the rest of the image is an inherently difficult task.

Applications that require an implicitly reconstructed signal are called **inverse imaging problems**, including deblurring, deconvolution, inpainting, super-resolution, and other related problems. Inverse imaging problems can often be modeled with an image formation model, as follows:

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\eta}$$

Let $\boldsymbol{x}$ be our underlying signal, $\boldsymbol{A}$ some linear corruption process, $\boldsymbol{\eta}$ a random noise factor, and $\boldsymbol{b}$ our measured image. The problem is then to extract $\boldsymbol{x}$, and $\boldsymbol{\eta}$ given either only $\boldsymbol{b}$ or given both $\boldsymbol{b}$ and $\boldsymbol{A}$ (note that $\boldsymbol{A}$ need not be invertible). Since such problems are inherently under-constrained, it is useful to utilize data-driven techniques to learn to reconstruct the underlying signal.

Generative models [1] have turned out to be powerful tools for such problems, as they implicitly represent priors for image distributions. Rather than training task-specific models with complex bespoke techniques, methods for solving inverse problems ideally would not require reinforcement learning or fine-tuning of models. In this paper, we implement and compare three techniques for leveraging diffusion models for inverse imaging problems: SDEdit [2], annealed langevin dynamics [3], or ScoreALD, and Diffusion Posterior Sampling [4], or DPS. SDEdit solves inverse problems by pipelining the given data as a guide image into a diffusion model as though it was a partially-generated image. ScoreALD and DPS approach inverse imaging problems as a Bayesian sampling problem, where one samples a likely image while using a diffusion model as a generative prior. They differ primarily in their approaches to posterior sampling throughout a diffusion process.

- *Isaac Criddle is with the Department of Computer Science at Stanford University, CA, 94305.*

2 Background

2.1 Half Quadratic Splitting

Half quadratic splitting [5] turns the inverse problem into a Bayesian maximum *a posteriori* problem, which can then be solved iteratively, alternating between a least-squares term for data fidelity and a regularizer representing a prior on the output distribution. In this way we can use arbitrary priors in conjunction with convex optimization techniques to estimate a maximum a posteriori solution to the problem, sidestepping the under-constrained nature of these inverse problems.

Traditional approaches to inverse problems have relied on hand-crafted priors such as total variation or other sparsity constraints. These methods, while mathematically well-founded, often struggle to capture the complexity of natural image distributions.

2.2 Generative Adversarial Networks

Deep learning techniques for image generation took off with the advent of **generative adversarial networks** [6], [7]. The central idea was to train two neural networks with competing aims. One network would learn to discriminate between images inside and outside of a distribution, and another would generate samples from that distribution.

2.3 Variational Autoencoders

Variational autoencoders implicitly learn data distributions by learning to compress a high-dimensional data space into a much lower-dimensional space and then re-project into the high-dimensional space. By sampling carefully within the low-dimensional space and then running the sample through the second half of the model, one can effectively draw samples from the high-distribution space.

2.4 Diffusion Models

Diffusion models [1] learn to draw samples from a distribution over a high-dimensional manifold by learning the

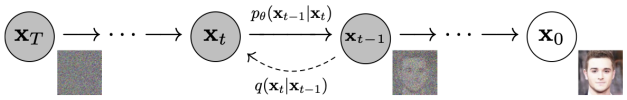

Fig. 1: The DDPM denoising reverse process, as shown in [1].

reverse process of an iterative noising walk. To train, one adds a certain amount of noise to an initial image according to some schedule, and the model learns to perform the reverse process. In the DDPM paper, they use a model which predicts the noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ instead of the score $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$, though the two formulations are mathematically equivalent. See the appendix for details on conversion between noise-based and score-based formulations.

Algorithm 1 Training denoising diffusion models

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}, t \right) \right\|^2$$

6: **until** converged

Once the model has implicitly learned our likelihood distribution, we can sample from the likelihood distribution by performing the reverse process, iteratively denoising from pure Gaussian noise T consecutive times, according to our noise schedule $\hat{\alpha}_t \Leftrightarrow \beta_t$.

Algorithm 2 Unconditioned sampling from diffusion models

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

Since we are implicitly sampling from a learned distribution of images seen in the training set, a diffusion model will produce very different outputs depending on the noise with which the unconditioned sampling is initialized.

Fig. 2: Three images sampled from the model from [4]. Trained on the Flickr-Faces-HQ (FFHQ) dataset.

Fig. 3: The denoising reverse process at $t = 0.8T$, $t = 0.2T$, and $t = 0$, where $t = T$ is sampled from a unit normal distribution.

3 Methods

For all of our approaches, we use the DPS paper [4]'s unconditioned image diffusion model, which was trained on the FFHQ Flickr faces dataset [7]. The dataset consists of 70,000 faces from a wide variety of ethnicities and settings.

3.1 SDEdit

SDEdit [2] applies pretrained diffusion models by taking a guide image, applying some noise according to the forward noise schedule, and then running that noised guide image through the reverse process of the diffusion model with no other alterations. The unmasked version of the algorithm, Algorithm 3, is written as follows in their paper.

Algorithm 3 SDEdit

Require: $\mathbf{x}^{(g)}$ (guide), t_0 (SDE hyper-parameter), N (total denoising steps), K (total repeats)
1: $\Delta t \leftarrow \frac{t_0}{N}$
2: $\alpha(t_0) \leftarrow \prod_{n=1}^{N} \left(1 - \beta\left(\frac{n t_0}{N}\right) \Delta t\right)$
3: **for** $k \leftarrow 1$ to K **do**
4: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad \mathbf{x} \leftarrow \sqrt{\alpha(t_0)}\mathbf{x} + \sqrt{1 - \alpha(t_0)}\mathbf{z}$
6: $\quad$ **for** $n \leftarrow N$ to 1 **do**
7: $\quad\quad t \leftarrow t_0 \frac{n}{N}$
8: $\quad\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
9: $\quad\quad \mathbf{x} \leftarrow \frac{1}{\sqrt{1-\beta(t)\Delta t}} \left(\mathbf{x} + \beta(t)\Delta t\, \mathbf{s}_\theta(\mathbf{x}, t)\right) + \sqrt{\beta(t)\Delta t}\, \mathbf{z}$
10: $\quad$ **end for**
11: **end for**
12: **return** $\mathbf{x}$

A slightly more complex version for masked images appears in the appendix to their paper which simply forces the model only to generate new data at missing pixels.

SDEdit requires as a hyperparameter a time t_0 at which to inject the guide image into the reverse process. A larger t_0 will correspond to less fidelity to the guide image, as more noise will be injected and the model will iterate through the process more times. A smaller t_0 corresponds to a greater fidelity to the guide image, though it gives the model less iterations to reconstruct missing data.

3.2 ScoreALD

In their paper, Jalal et al. [3] train a score-based model and utilize a Bayesian posterior sampling approach. Together the input image and a model define a posterior distribution

Fig. 4: The SDEdit pipeline. Left: Input corrupted image. Middle: Initialized input at $t = 0.5T$. Right: SDEdit outuput.

from which one can sample. Rather than starting with a partially-noised image as seen in SDEdit, ScoreALD begins with complete noise but conditions the diffusion model's output on the input image at each timestep t. More specifically, at each timestep we subtract the score of our input image $\boldsymbol{y}$ given our partially denoised diffusion image $\boldsymbol{x}_t$.

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t) \approx -\frac{1}{2(\sigma^2 + \gamma_t^2)} \nabla_{\boldsymbol{x}_t} \|\boldsymbol{y} - A(\boldsymbol{x}_t)\|_2^2$$

Here σ^2 is the variance in our image formation model, γ is a hyperparameter describing the strength of the conditioning, and A is our image formation process itself. For the purposes of this paper, A represents the blurring of the image followed by the addition of additional noise, or the masking off of certain pixels. By increasing γ as t decreases, we can simulate an annealing process where we converge to a more probable sample.

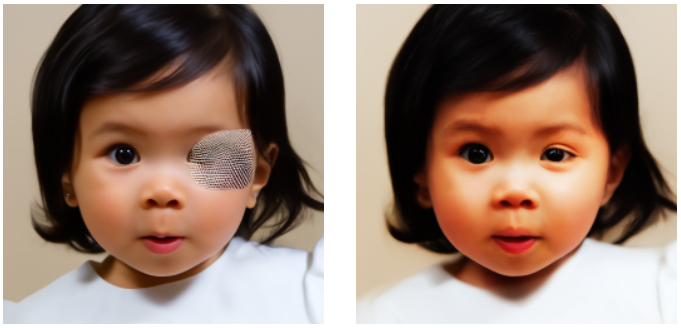

Fig. 5: ScoreALD sampled with $\gamma = 10.0$ on the left for more faithfulness to the input image and $\gamma = 20.0$ on the right for more realism.

Note that for ScoreALD to be viable, we need an approximation of our image formation model, which is trivial for inpainting and deblurring, but may be impossible to procure for other inverse problems.

In a manner similar to half quadratic splitting, we can alternate between taking diffusion steps and applying the ScoreALD conditioning to guide the model's output using our input data. ScoreALD is also sensitive to the initialization noise.

3.3 Diffusion Posterior Sampling

DPS shares with ScoreALD an approach that samples from the posterior distribution of images conditioned on our input data $\boldsymbol{y}$. However, rather than estimating the score $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y} \mid \boldsymbol{x}_t)$ via the image at the current timestep,

Algorithm 4 ScoreALD

Require: $N, y, \{\zeta_t\}_{t=1}^N, \{\tilde{\sigma}_t\}_{t=1}^N$
1: $\mathbf{x}_N \sim \mathcal{N}(0, I)$
2: **for** $t = T - 1$ to 0 **do**
3: $\quad \hat{\mathbf{s}} \leftarrow \mathbf{s}_\theta(\mathbf{x}_t, t)$
4: $\quad \hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t + (1 - \bar{\alpha}_t)\hat{\mathbf{s}})$
5: $\quad \mathbf{z} \sim \mathcal{N}(0, I)$
6: $\quad \mathbf{x}'_{t-1} \leftarrow \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \hat{\mathbf{x}}_0 + \tilde{\sigma}_t \mathbf{z}$
7: $\quad \mathbf{x}_{t-1} \leftarrow \mathbf{x}'_{t-1} - \frac{1}{2(\sigma^2+\gamma_t^2)} \nabla_{x_t} \|A(x_t) - \mathbf{y}\|^2$
8: **end for**
9: **return** $\hat{\mathbf{x}}_0$

Fig. 6: The ScoreALD-conditioned denoising reverse process for the box inpainting task at $t = 0.8T$, $t = 0.2T$, and $t = 0$.

diffusion posterior sampling estimates the score via *the predicted reconstructed image* $\hat{\mathbf{x}}_0$. This image can be extracted from both score- and noise-based models, as shown in the appendix. We then split the solving of the stochastic differential equation into two steps as was done in ScoreALD, and the diffusion process otherwise remains identical to the unconditioned generation.

Algorithm 5 Diffusion Posterior Sampling

Require: $N, y, \{\zeta_i\}_{i=1}^N, \{\tilde{\sigma}_i\}_{i=1}^N$
1: $\mathbf{x}_N \sim \mathcal{N}(0, I)$
2: **for** $i = N - 1$ to 0 **do**
3: $\quad \hat{\mathbf{s}} \leftarrow \mathbf{s}_\theta(\mathbf{x}_i, i)$
4: $\quad \hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}} (\mathbf{x}_i + (1 - \bar{\alpha}_i)\hat{\mathbf{s}})$
5: $\quad \mathbf{z} \sim \mathcal{N}(0, I)$
6: $\quad \mathbf{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i} \mathbf{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i} \hat{\mathbf{x}}_0 + \tilde{\sigma}_i \mathbf{z}$
7: $\quad \mathbf{x}_{i-1} \leftarrow \mathbf{x}'_{i-1} - \zeta_i \nabla_{\mathbf{x}_i} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$
8: **end for**
9: **return** $\hat{\mathbf{x}}_0$

Here ζ_i is a hyperparameter weighting the influence of the input data $\boldsymbol{y}$. It can be seen as a step size of the score, and we choose ζ_i to be $\frac{\zeta_{\text{user}}}{\nabla_{\mathbf{x}_i} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2}$ in our comparisons, normalizing our gradient steps.

4 Results

To compare outputs, we use two metrics: PSNR and LPIPS.

4.1 PSNR definition

Peak signal-to-noise ratio is a simple metric with broad application in the sciences for its simple calculation and evaluation:

$$\text{PSNR} = 10 \log_{10}(\frac{\text{MAX}^2}{\text{MSE}})$$

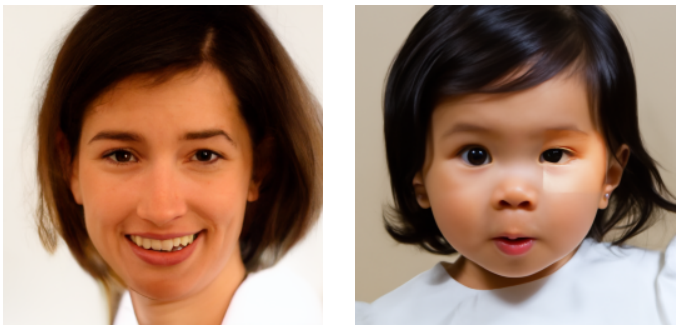

Fig. 7: DPS sampled with $\zeta_{\text{user}} = 0.1$ on the left and $\zeta_{\text{user}} = 1.0$ on the right.

Here MAX is the maximum value of a pixel, and MSE is the mean squared error between the measurement and the ground truth. A higher PSNR indicates higher fidelity. PSNR is not perceptual in nature, and as a result sometimes does not align with perceived visual quality.

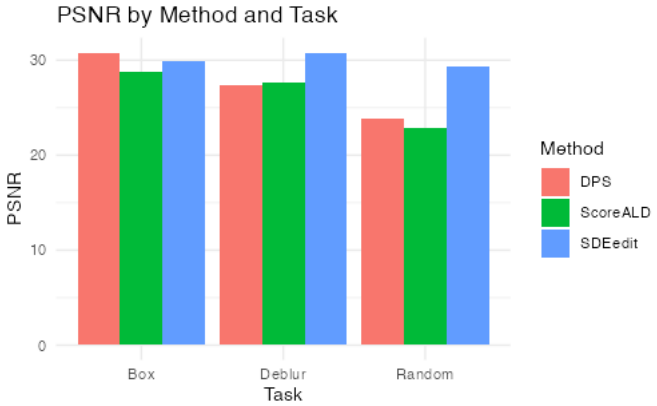

Fig. 8: PSNR comparison across methods and tasks ($n = 9$ test images).

4.2 LPIPS definition

LPIPS, proposed by Zhang et al. [8], compares two images not by comparing their values pixel-to-pixel but rather by running the two images through several layers of a pretrained neural network, and then extracting high-level features from the neural network to compare patch-by-patch. This approach is closer to a perceptual similarity, as the model compares the images not by values but by distance within a learned feature space. A lower LPIPS corresponds to a higher visual fidelity.

4.3 Inpainting Comparison

To compare the three approaches, we'll use two types of inverse problems: box inpainting and random pixel inpainting, where one in three pixels has been randomly removed.

On the box removal task, the SDEdit almost always did very well using the masked formulation. ScoreALD often lacked facial symmetry, and the DPS approach often yielded an eye with a clear border around the missing patch, which was often highly saturated.

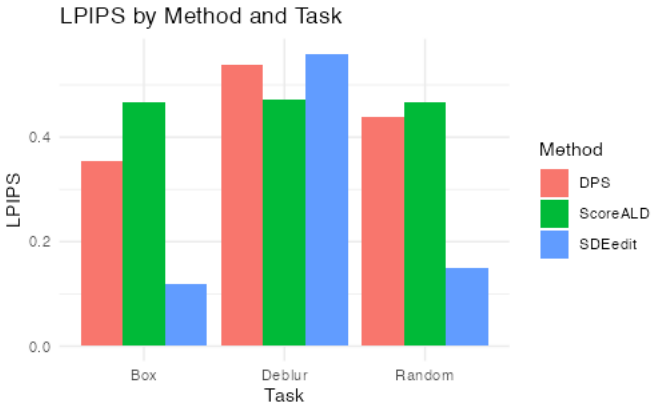

Fig. 9: $n = 9$. Note that SDEdit does best on inpainting tasks and worst on deblurring tasks.

TABLE 1: Quantitative Comparison on Young Boy Box Inpainting

Method	PSNR (dB) ↑	LPIPS ↓
SDEdit	29.5	0.06
ScoreALD	27.3	0.10
DPS	30.1	0.07

On the random pixel inpainting task, the SDEdit approach yielded the lowest LPIPS but the highest PSNR. ScoreALD and DPS achieved similar results. SDEdit was able to preserve the valid pixels, but both DPS and ScoreALD contributed hallucinations to the valid pixels.

4.4 Deconvolution Comparison

In general, ScoreALD produced the best results on the deblur task. In the case of the young boy image, DPS did unusually well, as can be seen in the table 2 and figure 12.

TABLE 2: Quantitative Comparison on Deconvolution

Method	PSNR (dB) ↑	LPIPS ↓
SDEdit	19.1	0.28
ScoreALD	21.8	0.20
DPS	26.7	0.08

4.5 Generalization

None of the approaches is able to generalize perfectly well outside of the training distribution. For instance, if asked to deblur the Mona Lisa, SDEdit, ScoreALD and DPS achieve LPIPS scores of 0.56, 0.47, 0.54, a 2.2x increase over their scores for the same task on the boy. Perceptually, the results appear far less convincing than they do on the face photograph outputs. This is a limitation of the model itself more than of these approaches, but some ability to generalize out of distribution is important for real-world applications.

5 Discussion

The posterior sampling approaches did well on the deblurring tasks, and less well on the inpainting tasks. SDEdit, on the other hand, could mask out its generation to only the needed pixels. When it came to the deblurring tasks, SDEdit

(a) Input Image

(b) SDEdit

(c) ScoreALD

(d) DPS

Fig. 10: Box Inpainting results

(a) Input Image

(b) SDEdit

(c) ScoreALD

(d) DPS

Fig. 11: Random Pixel Inpainting results

(a) Input Image

(b) SDEdit

(c) ScoreALD

(d) DPS

Fig. 12: Deblurring results

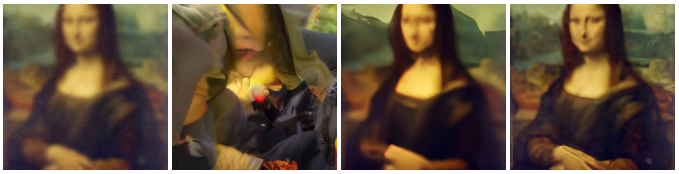

Fig. 13: Results on an out-of-distribution deblur task. From left to right: input, SDEdit, ScoreALD, and DPS. DPS obtains the best visual quality, but ScoreALD achieves the lowest LPIPS and highest PSNR.

tended to hallucinate more detail than DPS and ScoreALD did.

SDEdit's approach is simple yet powerful. SDEdit often achieved the best results on inpainting tasks, and more consistently preserved skin tone fidelity. SDEdit did significantly worse on the deblurring task, where it would often cause the model to hallucinate fine details that were not present in the ground truth image.

In contrast, ScoreALD and DPS achieved generally consistent and similar results. In general, they seem to provide more control and to be more generalizable than SDEdit. DPS seems to give slightly more stable generation than ScoreALD. Nevertheless, they struggled with inpainting somewhat because not only did they have to construct data for the missing pixels, but they also had to reconstruct the entire image.

5.1 Limitations

Several limitations warrant discussion here. First, all experiments were conducted using a single pretrained model on a limited selection of images from the FFHQ dataset. For a

proper comparison, a larger gamut of images, models, and hyperparameters should be tested.

We also only test these methods on two inpainting tasks: inpainting and deblurring. While these methods achieve good results on these tasks, it may be that the same approaches suffer for other inverse problem categories. For instance, Chung et al. [4] mention that ScoreALD suffers in non-linear problem spaces.

5.2 Practical Considerations and Applications

While performance and computational intensity are generally outside of the scope of this paper, it is worth noting that SDEdit requires significantly less compute than the other methods, requiring up to 70% fewer diffusion timesteps, depending on hyperparameter choice. SDEdit also does not require an additional expensive gradient calculation step.

5.3 Future Research

Several directions for future work are evident. It may be possible to optimize hyperparameter choices and schedules with more advanced adaptive learning algorithms.

DPS assumes that one can approximate the score by running an estimate of $\hat{\boldsymbol{x}}_0$ through the forward process and comparing that image to the measured input, but the technique does not empirically validate that assumption. Future research might identify other ways to estimate the score in specific problem domains.

Future work may also consider application of these principles for other problem domains, beyond imaging, to signal processing, artist workflows, and scientific applications.

5.4 Conclusion

This work shows that diffusion models can represent powerful generative priors for under-constrained inverse imaging problems, and that data-driven approaches to such problems have enormous potential. By conditioning diffusion models on given data, it is possible to sample from distributions that are difficult to model directly.

Acknowledgments

The author would like to thank Gordon Wetzstein and Sonia Kim for their instruction and feedback. The author also acknowledges the Wikimedia foundation for providing the Mona Lisa image.

Isaac Criddle is a master's student in computer science at Stanford University with interests in computer graphics and film.

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[2] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," 2022. [Online]. Available: https://arxiv.org/abs/2108.01073

[3] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. I. Tamir, "Robust compressed sensing mri with deep generative priors," 2021. [Online]. Available: https://arxiv.org/abs/2108.01368

[4] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," 2024. [Online]. Available: https://arxiv.org/abs/2209.14687

[5] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *Trans. Img. Proc.*, vol. 4, no. 7, p. 932–946, Jul. 1995. [Online]. Available: https://doi.org/10.1109/83.392335

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[7] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2019. [Online]. Available: https://arxiv.org/abs/1812.04948

[8] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018. [Online]. Available: https://arxiv.org/abs/1801.03924

Appendix

Mathematical Equivalencies

.1 Markov Chain vs. Conditional Distribution Formulations

DDPM introduces a markov chain formulation of the forward process:

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\,\boldsymbol{z}_{t-1}, \quad t = 1, 2, \ldots, T$$

Another formulation, dependent not on the last time step but only on the original image and a vector sampled from a multivariate unit normal distribution, can be expressed as follows:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\,\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{z}$$

The two formulations are equivalent, as shown by induction. First, it can be shown that A and A are true for the base case of $t = 1$.

$$\begin{aligned}\boldsymbol{x}_1 &= \sqrt{\bar{\alpha}_1}\,\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_1}\,\boldsymbol{z} \\ &= \sqrt{\alpha_1}\,\boldsymbol{x}_0 + \sqrt{1-\alpha_1}\,\boldsymbol{z} \\ &= \sqrt{1-\beta_1}\,\boldsymbol{x}_0 + \sqrt{\beta_1}\,\boldsymbol{z}\end{aligned}$$

It can then be shown that if the two statements are equivalent for t, they are also equivalent for $t+1$.

$$\begin{aligned}&\boldsymbol{x}_{t+1} = \sqrt{1-\beta_{t+1}}\,\boldsymbol{x}_t + \sqrt{\beta_{t+1}}\,\boldsymbol{z}_t \\ &= \sqrt{1-\beta_{t+1}}\,(\sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\,\boldsymbol{z}_{t-1}) + \sqrt{\beta_{t+1}}\,\boldsymbol{z}_t \\ &= \sqrt{(1-\beta_{t+1})(1-\beta_t)}\,\boldsymbol{x}_{t-1} + \\ &\qquad \sqrt{(1-\beta_{t+1})\beta_t}\,\boldsymbol{z}_{t-1} + \sqrt{\beta_{t+1}}\,\boldsymbol{z}_t\end{aligned}$$

Since $\boldsymbol{z}_t$ and $\boldsymbol{z}_{t+1}$ are both distributed according to a standard normal distribution, we can substitute their combination with the product of a scalar and another random variable $\boldsymbol{z}_{temp}$ which has a standard normal distribution.

$$\begin{aligned}
&\boldsymbol{x}_{t+1} = \sqrt{(1-\beta_{t+1})(1-\beta_t)}\,\boldsymbol{x}_{t-1} + \\
&\quad \sqrt{1-(1-\beta_{t+1}-\beta_t+\beta_{t+1}\beta_t)}\,\boldsymbol{z}_{temp} \\
&= \sqrt{\alpha_{t+1}\alpha_t}\,\boldsymbol{x}_{t-1} + \\
&\quad \sqrt{1-(1-\beta_{t+1})(1-\beta_t)}\,\boldsymbol{z}_{temp} \\
&= \sqrt{\alpha_{t+1}\alpha_t}\,(\sqrt{\bar{\alpha}_{t-1}}\,\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{z}_{t-1}) + \\
&\quad \sqrt{1-\alpha_{t+1}\alpha_t}\,\boldsymbol{z}_{temp} \\
&= \sqrt{\alpha_{t+1}\alpha_t\bar{\alpha}_{t-1}}\,\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_{t-1})\alpha_{t+1}}\,\boldsymbol{z}_{t-1} + \\
&\quad \sqrt{1-\alpha_{t+1}\alpha_t}\,\boldsymbol{z}_{temp}
\end{aligned}$$

Again, since $\boldsymbol{z}_{t-1}$ and $\boldsymbol{z}_{t+1}$ each have a standard normal distribution, we can express their combination as the scale of another random variable $\boldsymbol{z}$ with a standard normal distribution.

$$\begin{aligned}
&\boldsymbol{x}_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\,\boldsymbol{x}_0 + \\
&\quad \sqrt{1-\alpha_{t+1}\alpha_t+\alpha_{t+1}\alpha_t-\alpha_{t+1}\alpha_t\bar{\alpha}_{t-1}}\,\boldsymbol{z} \\
&= \sqrt{\bar{\alpha}_{t+1}}\,\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_{t+1}}\,\boldsymbol{z}
\end{aligned}$$

Thus, by induction, the two formulations are equivalent for all $t \in \mathbb{Z}^+$.

.2 Equivalence of Predicting Score vs. Predicting the Final Image

Two formulations of the problem include modeling the score, or the gradient of the log likelihood of the image, and predicting the image at the beginning of the forward process, $\hat{\boldsymbol{x}}_0$. The two formulations are equivalent, as shown below.

$$\begin{aligned}
&\boldsymbol{x}_{t-1} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\alpha_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\boldsymbol{x}}_0 \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\alpha_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} \\
&\quad \left(\frac{1}{\sqrt{\alpha_t\bar{\alpha}_{t-1}}}(\boldsymbol{x}_t + (1-\bar{\alpha}_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t,t))\right)
\end{aligned}$$

Simplifying, we obtain

$$\begin{aligned}
&\boldsymbol{x}_{t-1} = \frac{\alpha_t(1-\bar{\alpha}_{t-1})+1-\alpha_t}{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}\alpha_t)}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) \\
&= \frac{1-\bar{\alpha}_{t-1}\alpha_t+\alpha_t-\alpha_t}{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1}\alpha_t)}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) \\
&= \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}} + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) \\
&= \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + (1-\alpha_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t,t))
\end{aligned}$$

.3 Equivalence of Predicting Score vs. Predicting Noise

Some of the literature utilizes models which predict the noise component of the image directly, while other models predict the gradient of the log likelihood. The two formulations are mathematically equivalent and interchangeable, as shown below.

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + (1-\alpha_t)\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t))$$

$$\Leftrightarrow$$

$$\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) = -\frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t))}{\sqrt{1-\bar{\alpha}_t}}$$

$$\begin{aligned}
&\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) = \frac{\sqrt{\bar{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] - \boldsymbol{x}_t}{1-\bar{\alpha}_t} \\
&= \boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) = \frac{\sqrt{\bar{\alpha}_t}\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}{1-\bar{\alpha}_t} \\
&= \boldsymbol{s}_\theta(\boldsymbol{x}_t,t)) = \frac{\sqrt{\bar{\alpha}_t}(\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t] - \boldsymbol{x}_0) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}{1-\bar{\alpha}_t} \\
&= -\frac{\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t,t))}{\sqrt{1-\bar{\alpha}_t}}
\end{aligned}$$

Ergo the two formulations are interchangeable.