Diffusion on Nearest Target Neighbor

Luzhe Sun*
Computer Science Department, University of Chicago, Chicago, US
Takuma Yoneda*
RIPL, Toyota Technological Institute at Chicago

2022-12-08

Abstract

Diffusion model is a popular generative models recently. Based on diffusion planning, this paper explores several aspects of the diffusion model for conditional sampling that need attention. (1) the unreasonableness of conditional sampling by fixing some regions of the sample. (2) the necessity of conditional sampling by diffusion first and then denoising. (3) method to find the nearest target sample.

Keywords: diffusion model; reinforcement learning; conditional sampling

1 Introduction

A diffusion probabilistic model (which we will call a "diffusion model" for brevity) is a parameterized Markov chain trained using variational inference to produce samples matching the data after finite time. (Ho, Jain, & Abbeel, 2020) Transitions of this chain are learned to reverse a diffusion process, which is a Markov chain that gradually adds noise to the data in the opposite direction of sampling until signal is destroyed. 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.
The Diffusion Planning paper(Janner, Du, Tenenbaum, & Levine, 2022) introduces the method of trajectory complementation, where conditional sampling is performed by continuously replacing a portion of the sample during the backpropagation of the diffusion model. However, this method has the obvious drawback that different parts of the same sample come from different distributions. A sample $S = \{A, B\}$, conditional sampling by repeatedly replacing A as part of the target sample will result in a sample S that does not actually belong to any reasonable distribution. This is verified in this paper based on the Stable Diffusion model.
To improve the rationality of conditional sampling, this paper adopts a process of diffusion followed by denoising of user input samples, which protects the user input information while moving the input samples to the nearest target samples and better completes the conditional sampling.

2 Score Matching and Diffusion Model

2.1 Score Matching

The idea of score matching was originally proposed by Hyvarinen et al. (Hyvärinen, 2005), Parameters are estimated by minimizing the **expected squared distance** of the **gradient of the log-density** of the model and the gradient of the log-density of the observed data.
We first define the gradient of the log-density data as Stein score function:

$$\mathcal{F}_\theta(\mathbf{x}) = \nabla_x \log p(x) \tag{1}$$

However, to perform training, we would need to minimize the error of our model $\mathcal{F}_\theta(\mathbf{x})$ at predicting the gradient $\nabla_\mathbf{x} \log p(\mathbf{x})$, which amounts to minimize the Fisher divergence, or simply the MSE

$$\mathcal{L}_{mse} = E_{\mathbf{x}\sim p(\mathbf{x})} \left[\mathcal{F}_\theta(\mathbf{x}) - \nabla_\mathbf{x} \log p(\mathbf{x}) {}_2^2 \right] \tag{2}$$

The real $\nabla_\mathbf{x} \log p(\mathbf{x})$ is usually unknown, but under regularity assumptions on $p(\mathbf{x})$, it can be shown that the minimum of $\mathcal{L}_{mse}$ can be found through a tractable objective

$$\mathcal{L}_{matching} = E_{\mathbf{x}\sim p(\mathbf{x})} \left[tr\left(\nabla_\mathbf{x} \mathcal{F}_\theta(\mathbf{x})\right) + \frac{1}{2} \left\| \mathcal{F}_\theta(\mathbf{x}) {}_2^2 \right] . \tag{3}$$

where $\nabla_\mathbf{x} \mathcal{F}_\theta(\mathbf{x})$ denotes the Jacobian of $\mathcal{F}_\theta(\mathbf{x})$ with respect to $\mathbf{x}$, and $tr(\cdot)$ is the trace operation. The gradient of the log-density is just like an gradient map of the distribution.

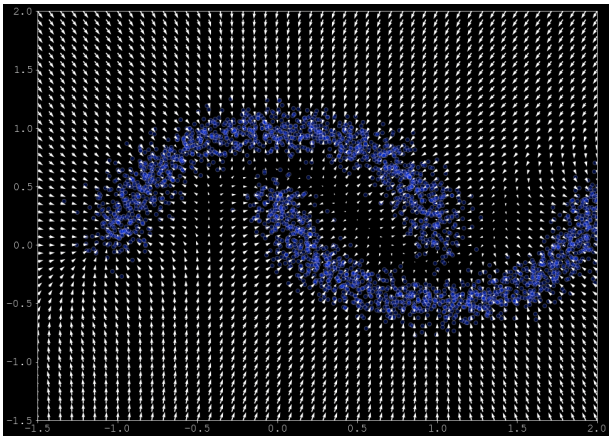

Figure 1: Diagram of gradient of the log-density
This image shows the gradient diagram of the moon distribution. Starting from any sample you can get, following the direction of the arrow you can get a sample inside the distribution.

2.2 Langevin Dynamics

Langevie dynamics can work on any score based models, it is a general procedure to draw samples from the score-based model. (Wikipedia contributors, 2022)
After training, our model is able to produce an approximation of the gradient of the probabiliity, such that $\mathcal{F}_\theta(\mathbf{x}) \approx \nabla_x \log p(x)$. Therefore, we could use this to generate data by relying on a simple gradient ascent from a given point by using an initial sample $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then using the gradient information to find a local maximum of $p(\mathbf{x})$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon \nabla_{\mathbf{x}_t} \log p\left(\mathbf{x}_t\right) \tag{4}$$

where ϵ defines the size of the step we take in the direction of the gradient (akin to the learning rate).

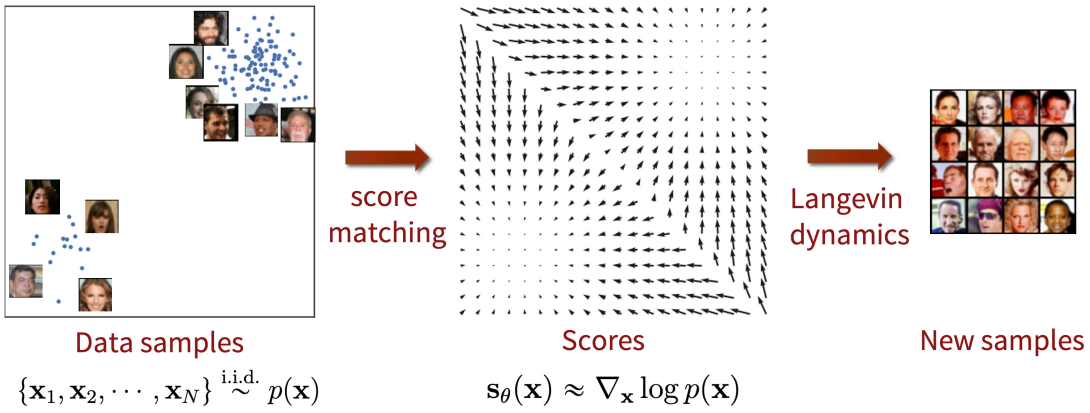

Figure 2: Langevin Dynamics Sampling Process(Song, n.d.)

2.3 Denoising Diffusion Probabilistic Model(DDPM)

Given a data distribution $x_0 \sim q\left(x_0\right)$, we define a forward noising process q which produces latents x_1 through x_T by adding Gaussian noise at time t with variance $\beta_t \in (0,1)$ as follows(Ho et al., 2020):

$$q\left(x_1, \ldots, x_T \mid x_0\right) := \prod_{t=1}^{T} q\left(x_t \mid x_{t-1}\right) \tag{5}$$

$$q\left(x_t \mid x_{t-1}\right) := \mathcal{N}\left(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}\right) \tag{6}$$

Given sufficiently large T and a well behaved schedule of β_t, the latent x_T is nearly an isotropic Gaussian distribution. Thus, if we know the exact reverse distribution $q\left(x_{t-1} \mid x_t\right)$, we can sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ and run the process in reverse to get a sample from $q\left(x_0\right)$. However, since $q\left(x_{t-1} \mid x_t\right)$ depends on the entire data distribution, we

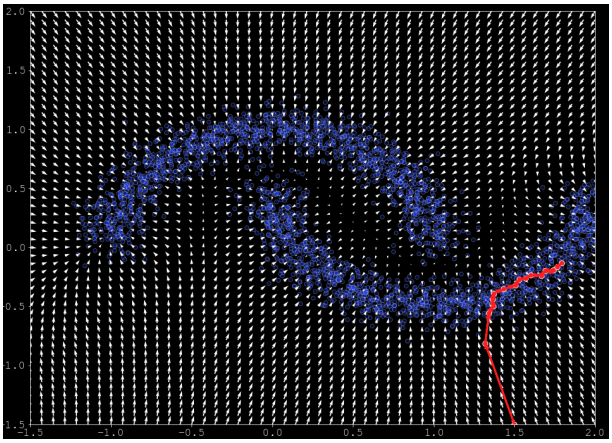

Figure 3: Langevin Dynamics Sample Process

approximate it using a neural network as follows: $p_\theta\left(x_{t-1} \mid x_t\right) := \mathcal{N}\left(x_{t-1}; \mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right)$ The combination of q and p is a variational auto-encoder(Kingma & Welling, 2013), and we can write the variational lower bound (VLB) as follows:

$$L_{\text{vlb}} := L_0 + L_1 + \ldots + L_{T-1} + L_T \tag{7}$$

$$L_0 := -\log p_\theta\left(x_0 \mid x_1\right) \tag{8}$$

$$L_{t-1} := D_{KL}\left(q\left(x_{t-1} \mid x_t, x_0\right) \| p_\theta\left(x_{t-1} \mid x_t\right)\right) \tag{9}$$

$$L_T := D_{KL}\left(q\left(x_T \mid x_0\right) \| p\left(x_T\right)\right) \tag{10}$$

Aside from L_0, each term of Equation 7 is a KL divergence between two Gaussians, and can thus be evaluated in closed form. To evaluate L_0 for images, we assume that each color component is divided into 256 bins, and we compute the probability of $p_\theta\left(x_0 \mid x_1\right)$ landing in the correct bin (which is tractable using the CDF of the Gaussian distribution). Also note that while L_T does not depend on θ, it will be close to zero if the forward noising process adequately destroys the data distribution so that $q\left(x_T \mid x_0\right) \approx \mathcal{N}(0, \mathbf{I})$.

As noted in (Ho et al., 2020), the noising process defined in Equation 2 allows us to sample an arbitrary step of the noised latents directly conditioned on the input x_0. With $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$, we can write the marginal

$$q\left(x_t \mid x_0\right) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} x_0, \left(1 - \bar{\alpha}_t\right) \mathbf{I}\right) \tag{11}$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \tag{12}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Here, $1 - \bar{\alpha}_t$ tells us the variance of the noise for an arbitrary timestep, and we could equivalently use this to define the noise schedule instead of β_t.

Using Bayes theorem, one can calculate the posterior $q\left(x_{t-1} \mid x_t, x_0\right)$ in terms of $\tilde{\beta}_t$ and $\tilde{\mu}_t\left(x_t, x_0\right)$ which are defined as follows:

$$\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \tag{13}$$

$$\tilde{\mu}_t\left(x_t, x_0\right) := \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}\left(1 - \bar{\alpha}_{t-1}\right)}{1 - \bar{\alpha}_t} x_t \tag{14}$$

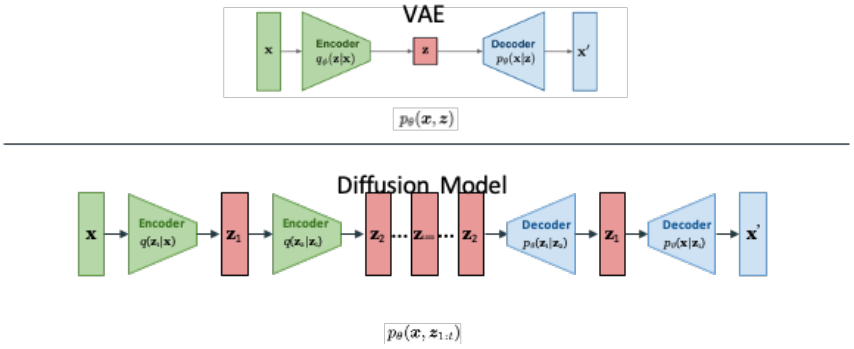

Figure 4: VAE and Diffusion Model

$$q\left(x_{t-1} \mid x_t, x_0\right)=\mathcal{N}\left(x_{t-1} ; \tilde{\mu}\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right) \tag{15}$$

We can conclude the algorithm in below format:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
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** $\mathbf{x}_0$

Figure 5: DDPM Algorithm

3 Conditional Sampling

Inspired by Diffusion Planning(Janner et al., 2022) method, this section mainly talks about how to reproduce diffusion planning on trajectories and how to do conditional sampling.

3.1 Diffusion Planning

In Diffusion Planning paper, they propose an alternative approach to data-driven trajectory optimization. The core idea is to train a model that is directly amenable to trajectory optimization, in the sense that sampling from the model and planning with it become nearly identical.

Diffuser samples plans by iteratively denoising two dimensional arrays consisting of a variable number of state-action pairs. A small receptive field constrains the model to only enforce local consistency during a single denoising step. By composing many denoising steps together, local consistency can drive global coherence of a sampled plan.

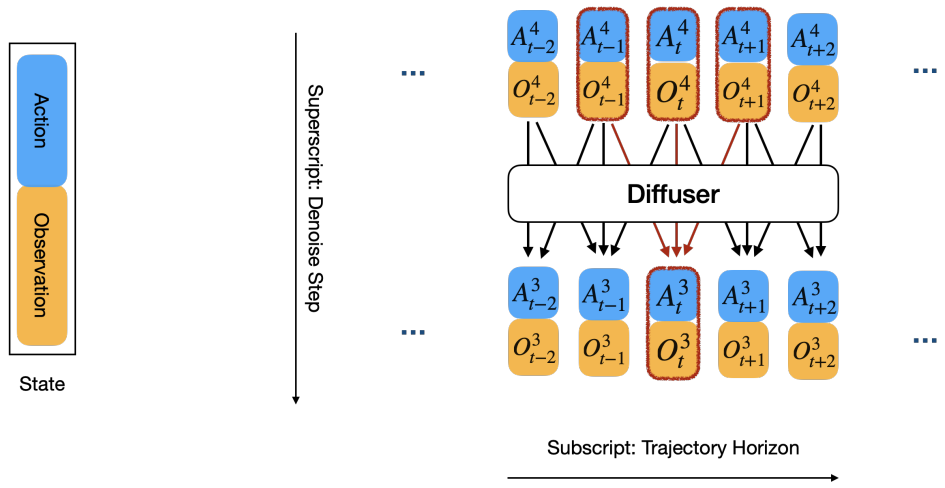

Figure 6: Diffusion Planning Structure

To predict a longer horizon, the diffuser is trained as a U-net Convolution Kernel instead of a whole network. In this manner, no matter how long the input trajectories is, the model can always behave denoising process.

3.2 Introduction to maze problem

The Maze2D domain involves moving force-actuated ball (along the X and Y axis) to a fixed target location. The observation consists of the (x, y) location and velocities.(Farama, n.d.)

$$Action = (x_v, y_v)$$

$$Observation = (x, y, x_v, y_v)$$

The task objective is to complete the trajectory prediction from the starting point to the end point within the specified length of steps, $L = 384$.

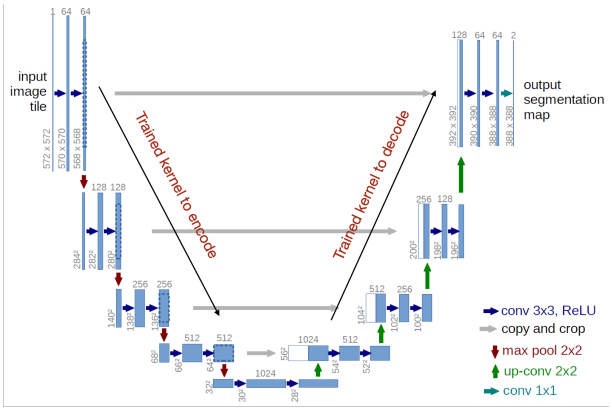

Figure 7: U-Net Structure

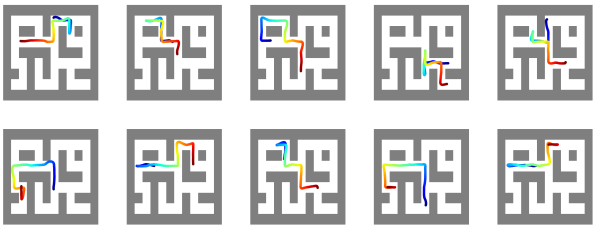

Figure 8: Demonstration Trajectory for Maze2D Experiments

3.3 Trajectory inpainting (Conditional Sampling)

The diffusion planning process aims to find a path between the start state S_0^* and target state S_T^*. To implement the conditional sampling process in the reverse process of diffusion model, we need to constantly replace the start and target state manually. The reverse process format shows below:

$$
\begin{aligned}
t = 0 &: \left[S_0^*, S_1^0, S_2^0, \ldots, S_{t-1}^0, S_T^*\right] \\
t = 1 &: \left[S_0^*, S_1^1, S_2^1, \ldots, S_{t-1}^1, S_T^*\right] \\
&\ldots \\
t = k &: \left[S_0^*, S_1^k, S_2^k, \ldots, S_{t-1}^k, S_T^*\right]
\end{aligned}
$$

The training process happens like below:

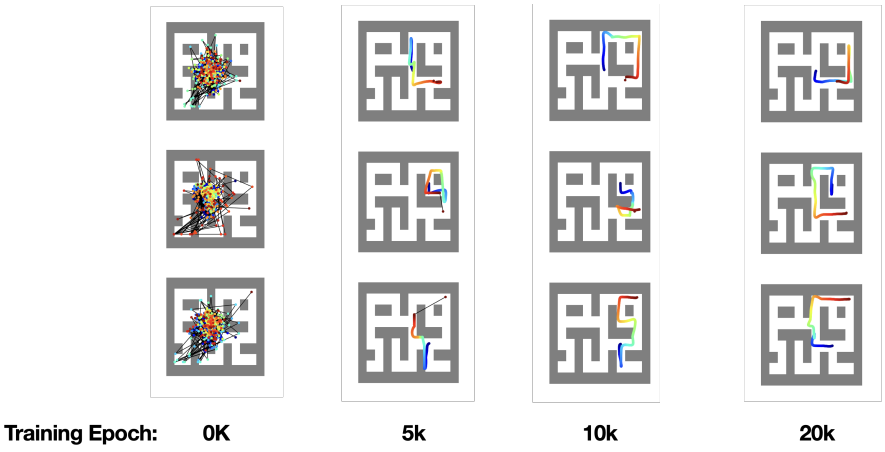

Figure 9: Training Process of Trajectory Inpainting

3.4 Trajectory stitching

Benefiting from the special structure of U-Net, we are able to have trajectories of different lengths for the trained trajectories and the predicted trajectories. That is, in the case of presenting only trajectories of length l, the model can stitch these short trajectories and obtain a longer trajectory with length $L(L \geq l)$.

Figure 10: Trajectory Stitching

Figure12 shows the scores obtained by training using trajectories of different lengths and predicting trajectories of length $L = 384$. We can see that the test scores obtained by training using the dataset of length 288 ($\frac{3}{4}L$, purple line) even exceed the benchmark(green line), which is train with L and test with L.

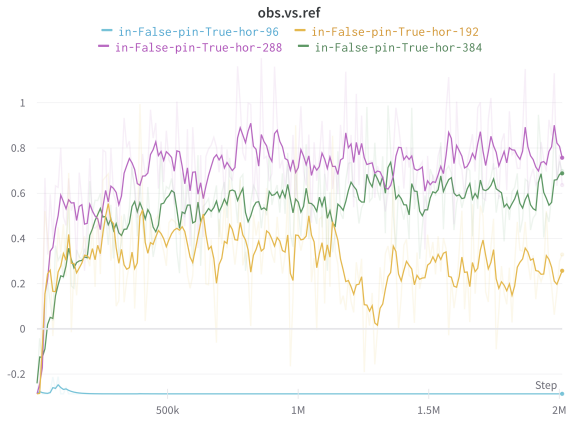

Figure 11: Model Evaluation

The main reason for this result is that the scoring mechanism of the D4RL maze model is continuously dense. That is, the closer to the end point the higher the score, and even if one just wanders near the end point, the relatively higher the score will be. Reference provide a reasonable solution, but maybe not best.

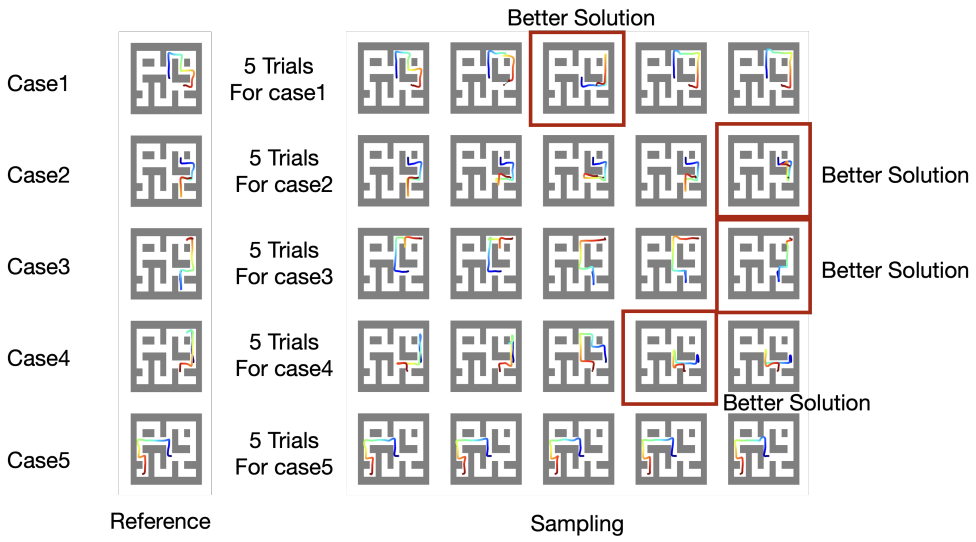

Figure 12: Evaluation Beat Benchmark

3.5 Unreasonableness

Diffusion Models have only seen samples from Gaussian distribution, what we are doing is not theoretically reasonable. We cannot expected Diffusion Model to start from a stitched sample like Fig13 a stitching cat image to generate an full image exactly the same ears. Since this image doesn't belong to any distribution that diffusion model have ever seen.

Figure 13: stitching cat image

Following the idea in Diffusion Planning, I reproduced the stable diffusion model and put the stitched cat image into it for conditional sampling. The cat image fits perfectly into the diffusion planning setup, with part of the same sample coming from the target distribution and part from the Gaussian distribution. The result of conditional sampling is shown in Figure14, where the stable diffusion model abandons the rationality of sampling and instead preserves the user's input.

Figure 14: conditional sampling of cat image

This simple conditional sampling model wants to show that fixing a portion of the sample and putting it into the diffusion model is not the right way to make conditional sampleing. This causes the sample to never exist in any distribution known to the diffusion model. As a result, the diffusion model loses information about the gradient of that sample, and the denoising process cannot be performed. This point is also mentioned in YangSong's model on score matching(Song et al., 2020). The following figure15 shows the forward and backward propagation of score matching from a two-point distribution to a Gaussian distribution. The dark blue area in the figure represents the sample outside the distribution, losing gradient/energy information.

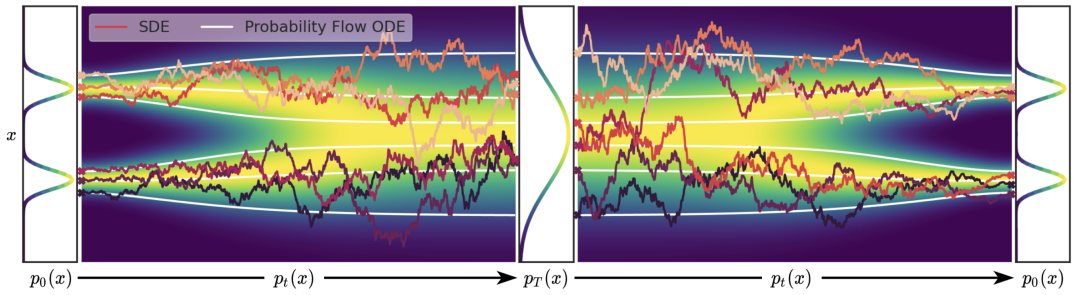

Figure 15: Model Energy/Gradient Field

3.6 Diffuse-conditional sampling

To conditionally sample a sample outside a distribution and make it fall into the target distribution. We need to first modify the input sample so that it falls into a distribution known to the diffusion model. To achieve this,

we need to diffuse the input samples until they fall into the known distribution before performing the denoising process.

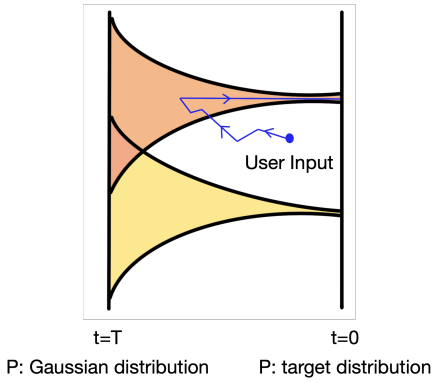

Figure 16: Necessity of adding noise to the user input

The whole training process can be summarized in the following diagram in Figure 1617. Whenever we receive a noiseless user input sample ($s \in P_{user}$), we need to perform diffuse, adding Gaussian noise until the sample confusion approximates the distribution learned by the diffusion model ($s' \in P_{NoisyTarget}$).

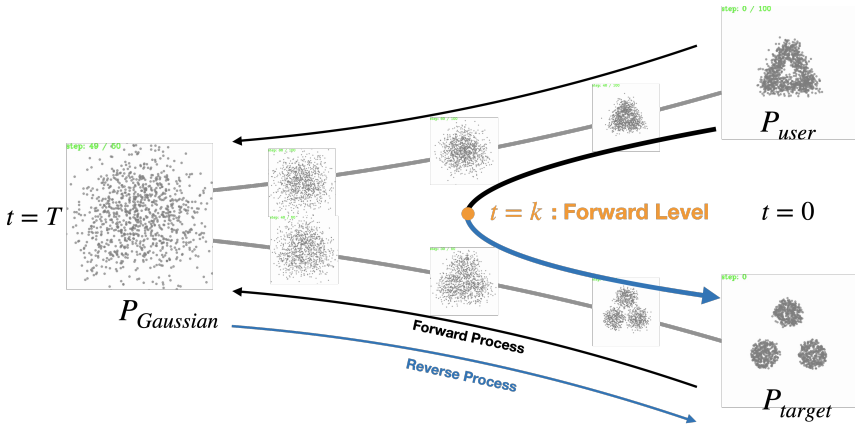

Figure 17: Diffuse-conditional sampling

In this way we can do conditional sampling in a more reasonable way, but then comes another problem, the choice of k-value. Currently I still do not know how to obtain a suitable k-value, which means that the distance of the current sample from the distribution needs to be measured.

4 Nearest Expert Neighbor

For this purpose, I conducted another experiment on whether the diffusion model can automatically find the nearest expert distribution. A three-point distribution was diffused for 100 steps and denoised for 100 steps. The samples were stained at 100, 70, and 50 steps of denoising process, and the final attribution of the stained samples was observed. Note that this is not a clustering problem. If the samples can automatically find the nearest target, then we should get 3 pure color target regions, and vice versa, the degree of error can be determined based on the degree of mixing of the 3 target regions.

According to the conclusion of the figure18, we can learn the following two points.

- The diffusion model aims to provide a reasonable sample, but does not directly find the nearest target sample.
- The greater the diffusion for user input, the less likely it is to find the nearest target sample in the denoising process.

Based on this conclusion, we can conclude in figure19 that the selection of k-values is a trade-off process. the larger the k-value, the more user input information we lose, but the more reasonable our generated results are (belonging to the target distribution). The smaller the k-value, the more user input information we retain, but the further our generated results may be from the target distribution.

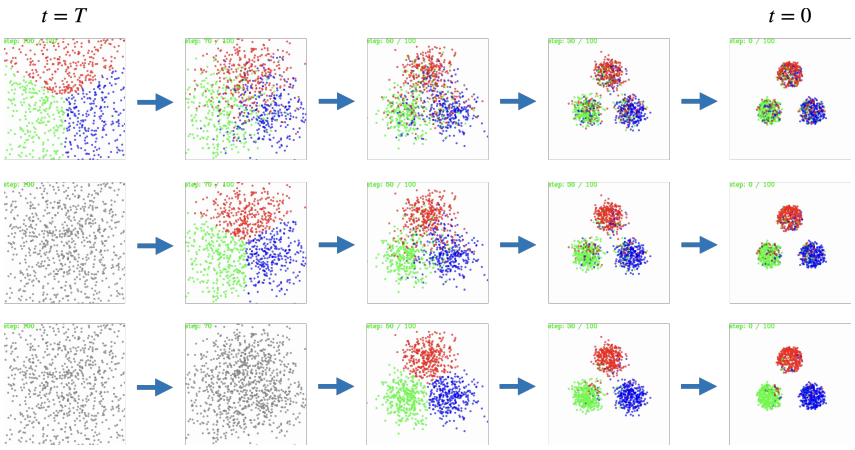

Figure 18: Hardness for finding nearest target distribution

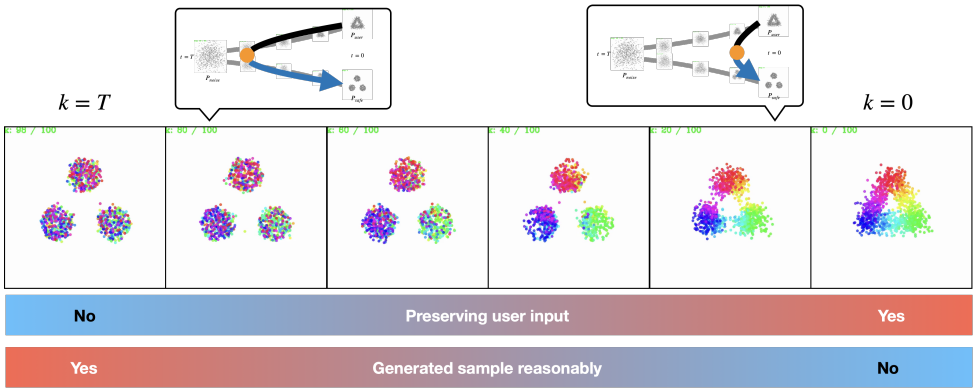

Figure 19: Switch Over K-value

By setting k values wisely, (which I can only do by constant testing at the moment), we can get impressive conditional sampling results.
The following experiment uses the pre-trained stable diffusion parameter and figure20 shows, with a real cat image as input, how we can get a cartoon cat image without guidance. When the value of k is reasonable, we can get the target cartoon cat. And when the value of k is too large, the information of the cat has been lost, and the sample information is only left with some color blocks and positions. The diffusion model modifies the image to the image of the cartoon character that appears most in the training data by default.

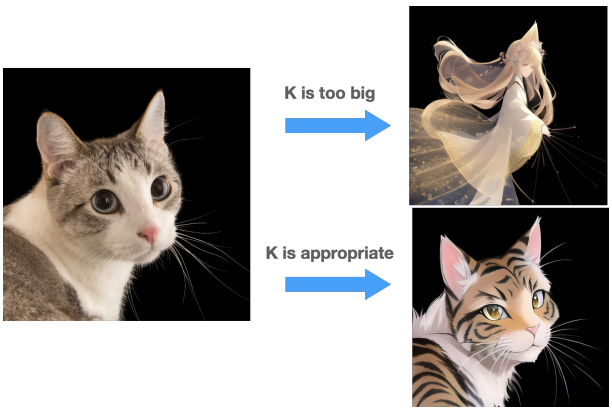

Figure 20: K-value for user intention

5 Future Work

There are 3 main areas that can be explored as future work.

5.1 K-value Detection

Based on our discussion above, we find that the choice of k value is crucial. Depending on the "clarity" of the user input, we need to be very flexible in adjusting the k-value. For example, if the user enters "table" but needs a "cat", then obviously we need a larger k value, keeping only the color block and location information of the table. If the user enters a "dog" and wants to get a "cat", then obviously we only need to set a very small k value to blur the features that belong to the dog.

5.2 In-distribution-ness

Knowing the distance of the current sample from the target distribution is important for the selection of the k value. As the figure21 shows, how do we determine how far forward the blue sample diffuse before it becomes fall into a known distribution?

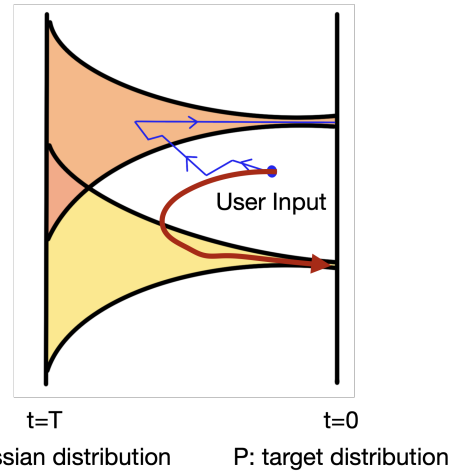

Figure 21: Find Nearest Neighbor

For example, if a user inputs a picture of a "cat" and desires a picture of a "cat", the model should perceive that the current sample is one of the samples of the target distribution and should not make any changes to the input. But the degree of sample obedience to the distribution, or distance measurement, is not a very straightforward matter. The first problem we encounter is that the "distribution" is a implicit variable built into the diffusion model. The diffusion model only trained on $\nabla_{\mathbf{x}} \log P(\mathbf{x})$ which is represented as an neural network. It's really hard to extract $P(\mathbf{x})$ out.

5.3 Nearest Neighbour

The last question is how can we guarantee that the model gives us the nearest (most relevant) target sample, rather than a random target sample. For example, train a model with a dataset based on a huge number of photos of dogs and humans. Now feeding a table with four legs, we should expect the model to generate photos of dogs with the same four legs, not photos of humans walking upright. Furthermore, if a sketch of a person is inputed, the model should return a similarly accurate picture of a person rather than a dog. The sketch and the table are exactly like the blue sample points in the figure21, so how do we ensure that it follows the blue trajectory instead of the red one? A simple idea is to learn the method in stable diffusion and set a text guide message. The message tells the model which target distribution we want to generate. This is a good idea, but should not be the best idea, the more precise the description, the more restrictive the model is and the more demanding it is for the user.

References

Farama. (n.d.). *D4rl.* https://github.com/Farama-Foundation/D4RL/wiki/Tasks.

Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising diffusion probabilistic models.* arXiv. Retrieved from https://arxiv.org/abs/2006.11239 DOI: 10.48550/ARXIV.2006.11239

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, *6*(24), 695–709. Retrieved from http://jmlr.org/papers/v6/hyvarinen05a.html

Janner, M., Du, Y., Tenenbaum, J. B., & Levine, S. (2022). *Planning with diffusion for flexible behavior synthesis.* arXiv. Retrieved from https://arxiv.org/abs/2205.09991 DOI: 10.48550/ARXIV.2205.09991

Kingma, D. P., & Welling, M. (2013). *Auto-encoding variational bayes.* arXiv. Retrieved from https://arxiv.org/abs/1312.6114 DOI: 10.48550/ARXIV.1312.6114

Song, Y. (n.d.). Retrieved from https://yang-song.net/

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2020). *Score-based generative modeling through stochastic differential equations.* arXiv. Retrieved from https://arxiv.org/abs/2011.13456 DOI: 10.48550/ARXIV.2011.13456

Wikipedia contributors. (2022). *Langevin dynamics — Wikipedia, the free encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Langevin_dynamics&oldid=1107558883 ([Online; accessed 5-December-2022])

Thanks for Xiaodan Du, Haochen Wang at TTIC for sharing idea and advanced knowledge of diffusion model with me.