CS182/282A: Deep Neural Nets December 7, 2022

Diffusion, Denoising, and Deep Networks

Instructor: Anant Sahai *Jamie Hong, Michael Lam, Mark Lindblad, Anze Liu*

0 Introduction

Denoising Diffusion Probabilistic Models.

In this assignment, we'll investigate how diffusion models work, involving:

1. the forward process where we add randomly sampled Gaussian noise to an input via a Markov Chain of diffusion steps,
2. analyzing different loss functions,
3. and followed by exploring how to reverse the diffusion process by denoising and generate new samples from the original predicted distribution.

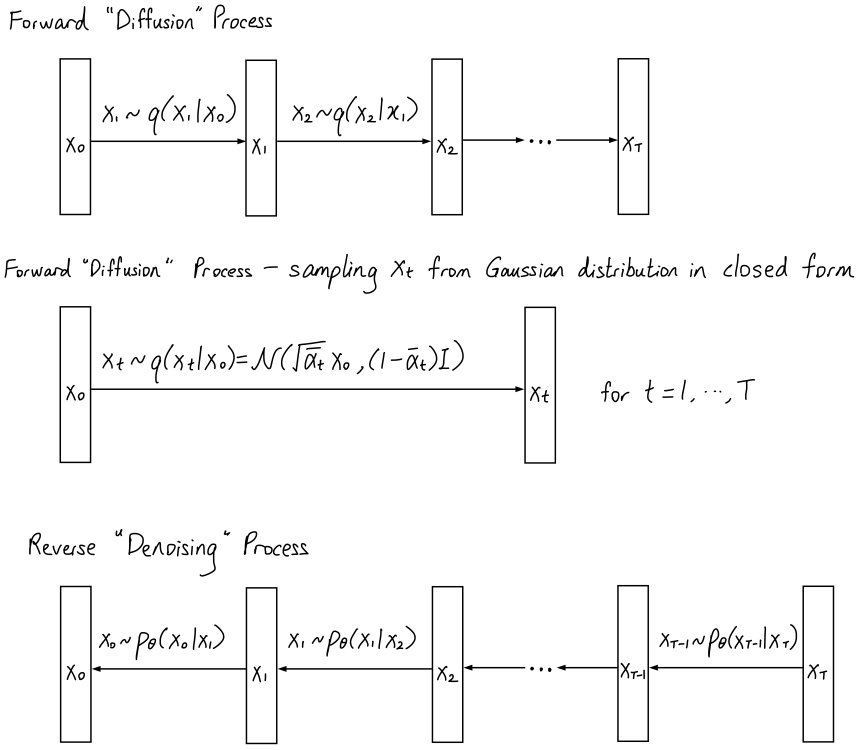

Figure 1: Forward Diffusion Process and Reverse Denoising Process

1 Forward "Noising" Process

Diffusion networks are neural networks that progressively "denoise" an input, e.g. image, that has been corrupted with Gaussian noise until the original input is reconstructed. This can be thought of as a two-pass Markov chain with T nodes. In the forward (first) pass, noise is added from state to state, which can be described by the following posterior:

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

where $\mathbf{x}_0$ is the initial image, and each subsequent step $\mathbf{x}_t$ is a "noised" version of the image at the previous time step. $\beta_1 ... \beta_T$ is the *variance schedule* because it controls how much each image at each time step is diffused; qualitatively, the higher the sampling variance β_t, the more unrecognizable an image will be after the diffusion at that time step.

In this problem, we will be exploring how to implement this forward diffusion pass through the Markov chain. The rest of the problems will be dedicated to the *reverse* pass, where the image will be progressively *denoised* until the resulting image will be close to the initial image $\mathbf{x}_0$.

1. **What happens when all the variances are 0? What do the images look like at each time step? What about when all the variances are 1?**
2. In the `diffusion.ipynb` Jupyter notebook, implement the `compute_mean` and `compute_cov` functions. Run the following cells to test your implementation. Then follow the instructions in the notebook to define your own input and variance schedules to visualize.
3. Once you have implemented the functions above, the code allows you to run the forward diffusion pass with a number of different variance schedules. Play around with these parameters and try testing different kinds of variance schedules. **How does the variance scheduling affect the rate at which the initial image becomes unrecognizable? What patterns do you notice?**
4. In practice, we often want to calculate the Gaussian noise at a time step t conditioned on the initial time step rather than the previous time step. **Given equation (1), derive a closed-form expression for** $q(\mathbf{x}_T|\mathbf{x}_0)$ and **show your work**.

   (Hint: Define variables $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.)

2 Optimization Loss Function

We will now study the reverse "denoising" pass of the diffusion network. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is the reverse process, represented by a Markov chain with learned Gaussian transitions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ for $t = 1, ..., T$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := N(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{2}$$

To optimize the reverse pass, we need to define an appropriate loss function for the final "denoised" state. In other words, given an end state $\mathbf{x}_t$, we would like to estimate the most likely initial state, i.e. the state $\mathbf{x}_0$ with the highest probability $p_\theta(\mathbf{x}_0)$. This is equivalent to finding the minimum of the negative log likelihood:

$$\mathbb{E}[-log\ p_\theta(\mathbf{x}_0)] \tag{3}$$

We would like to find a distribution that approximates the random variable so we can generate data via sampling to predict probabilities at different time steps. *Variational bound* (also known as the *evidence lower bound*) allows us to upper bound the above negative log likelihood as such:

$$\mathbb{E}\left[-log\ p(\mathbf{x}_0)\right] \ \leq\ \mathbb{E}_q\left[-log\ \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L \tag{4}$$

We define the variational bound as our loss function.

1. **Show that the following expression is equivalent to the variational bound on the negative log likelihood:**

$$\mathbb{E}_q\left[-log\ p_\theta(\mathbf{x}_T) - \sum_{t=1}^{T} log\ \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \tag{5}$$

2. For performance reasons, we would like to rewrite the upper bound derived in (2.1) as the *KL divergence* between the posterior distribution of Gaussian noise and the probability of a given state. The KL divergence between P and Q measures the "surprise" from sampling from distribution Q when the actual population distribution is P. Mathematically, it is defined as the following:

$$D_{KL}(P||Q)\ :=\ \sum_{x\in\chi} P(x)\ log\ \frac{P(x)}{Q(x)} \tag{6}$$

   (a) **First prove the following equation.** (Hint: Define $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ using conditional probability.)

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \tag{7}$$

   (b) **Using the above equation, show that (5) is equivalent to the following expression:** (Hint: use the properties of logarithms.)

$$\mathbb{E}_q[D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\ ||\ p(\mathbf{x}_T)) + \sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\ ||\ p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - log\ p_\theta(\mathbf{x}_0|\mathbf{x}_1)] \tag{8}$$

3 Reverse "Denoising" Process

Notice that the loss function, when written in the KL divergence form, involves a Gaussian posterior probability conditioned on $\mathbf{x}_t$ and $\mathbf{x}_0$. It can be shown that this distribution is actually tractable and has the following closed form:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = N(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \boldsymbol{I}) \tag{9}$$

where

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0\ +\ \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \qquad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{10}$$

and both α_t and $\bar{\alpha}_t$ are defined the same as in problem (1.4).

To establish the diffusion model, we need to choose the variances β_t for the forward process. To establish the denoising process, we need to choose a parameterization of the Gaussian distribution for the reverse process. We can then define a loss function using the parameterized distribution to construct a model.

To simplify implementation and improve computation efficiency, instead of directly computing the KL divergence loss on the target distribution q and predicted distribution p_θ, we choose to use the

Mean Squared Error of $\tilde{\mu}_t(x_t, x_0)$ (mean of target distribution) and $\mu_\theta(x_t, t)$ (mean of predicted distribution) as the training loss to obtain p_θ, which approximates q.

Revisiting $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := N(\mathbf{x}_{t-1};\ \mu_\theta(\mathbf{x}_t, t),\ \Sigma(\mathbf{x}_t, t))$ from (2), assume the variance is known from the variance scheduler and define $\Sigma_\theta(\mathbf{x}_t, t) := \sigma_t^2 \boldsymbol{I}$.

The objective is to find a reverse process distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := N(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \boldsymbol{I})$ such that μ_θ predicts the posterior mean $\tilde{\mu}_t$ of the forward process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = N(\mathbf{x}_{t-1};\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0),\ \tilde{\beta}_t \boldsymbol{I})$.

Define the loss at time $t-1$ as:

$$L_{t-1} := \mathbb{E}_q \left[\frac{1}{2{\sigma_t}^2} \left| \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t) \right|^2 \right] + C \tag{11}$$

where C is a constant independent on θ.

1. Using the following reparameterization of your answer to (1.3),

   $x_t(x_0, \epsilon) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ for $\epsilon \sim N(\mathbf{0}, \boldsymbol{I})$,

   Rewrite the loss (11) in terms of $x_t(x_0, \epsilon)$ (do not use $\tilde{\mu}_t$).

2. Instead of training the reverse process to predict the mean, we can train it to predict the noise ϵ. Specifically, using the following parameterization of μ_θ in terms of ϵ_θ :

   $$\mu_\theta(\mathbf{x}_t, t) = \tilde{\mu}_t(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_{\theta(}\mathbf{x}_t))) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)) \tag{12}$$

   rewrite your answer to problem (3.1) in terms of ϵ, ϵ_θ, and $\mathbf{x}_0$ only (no μ_θ or $\mathbf{x}_t$).

4 Predicting Mean vs. Predicting Noise

As shown in the prior questions, there are often multiple choices of loss functions and different parameterizations to choose from, whether to use for training or to gauge how effective one model might be over another.

Here, we'll look at two specific choices: one involving the mean of the distribution over the timesteps of the diffusion process, and the other involving the noise that was added to the input as a result of diffusion.

We'll consider just the loss functions themselves for this question, as the process of training denoising models will be covered in the last question of this homework.

1. **Implement both loss functions** in the `loss_func_comp.ipynb` Jupyter notebook, and compare their computation times.
2. **Comment on the relative speeds of these loss functions. How might this impact which function you'd want to use in practice?**

5 Code: Train and Observe

1. **Complete all parts of denoising.ipynb.**
2. Compare the states from the forward process and backward process. **What do you observe? Note down your observations in the written part of this homework.**

6 Homework Process and Study Group

1. **What sources did you use to work through this homework?**
2. **Did you work with anyone on this homework? If so, please list their names and Cal 1 ID's here.**
3. **About how many hours did this homework take you?**

References

[1] Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising Diffusion Probabilistic Models, *Advances in Neural Information Processing Systems 33 (NeurIPS)* (2020).

[2] The JAX Authors, Google, JAX reference documentation, *https://jax.readthedocs.io/en/latest/index.html* (2020).