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

Diffusion, Denoising, and Deep Networks: Commentary

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

0 Introduction

The motivating paper for our problem set is "Denoising Diffusion Probabilistic Models", which investigates diffusion probabilistic models and denoising with the goal of generating new high quality image samples.

Diffusion models are parameterized Markov chains trained to reverse a diffusion process in order to generate new samples from noise. The central advancement of the paper is parameterizing these diffusion models so that the denoising stages transform the input as if it were changing each stage to a lower noisiness score.

The structure of the diffusion models can be separated into three main components:

1. The forward process:

   Ideally, the original input from an interesting distribution is transformed into a latent space that approaches a Gaussian distribution. By adding Gaussian noise to the input over a series of time steps, we arrive at a state that looks approximately Gaussian.

   Though the process itself is non-deterministic due to the randomly sampling from Gaussian distribution and adding the Gaussian noise for each time step, the parameters that characterize the forward process (specifically, the variance schedules) are held to be fixed in the paper and in our problem set.

2. Predicting noise:

   Once the original input has been transformed to a diffused final state, we utilize the diffused state and the number of time steps for which it was diffused to predict the noise vector added during forward process. Here, the goal is that the predictor (some neural network being trained) will obtain an understanding of the underlying true distribution of the original inputs, and want to "push" the diffused states back onto the original distribution.

3. The reverse process:

   This is the process of generation, the ultimate goal of these diffusion models. Once the noise predictor has been trained, it can then be passed in a starting state sampled directly from a Gaussian to denoise one time step at a time. After enough time steps, the goal is for the resulting state to look like a new generated sample from the original interesting distribution.

In our problem set, we focus on teaching this generation via diffusion by introducing students to the necessary background and foundational equations to get some intuition and analytical understanding. Then, once we have introduced the students to diffusion probabilistic models analytically, we direct students to implement various parts in the set of three Jupyter notebooks using Jax: `diffusion.ipynb`, `loss_func_comp.ipynb`, and `denoising.ipynb`.

1 Forward “Noising" Process

This first problem introduces students to the forward process of progressively noising samples. The inputs to the process are noised by adding samples from a Gaussian and takes on the form below.

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

To build up intuition, we ask students to first consider the extremes of the noising parameters: what we expect to happen when the variances are almost 0 or when they are almost 1. Students then can build on this by engaging with these expressions through the `diffusion` notebook, which focuses on investigating and visualizing different choices of variance schedules.

In the notebook, we chose to split the diffusion process into the mean, variance, and randomness functions. This way, student-written code is deterministic, allowing us to test their implementation with an autograder that averages their results over a number of test cases.

After implementing, students can then see the visual results of running the diffusion process on a few given variance schedules. This is to get students thinking about what the process would look like with other variance schedules, which the next part of the notebook asks them to come up with to create their own visualizations.

(Optionally, we also include a small comparison of timed performance when diffusing without any speedups versus when diffusing using a function sped up by Jax's JIT functionality. This is to comment on how computation can take much longer for diffusion being used in training for larger models, and that even a small speedup for an inner function may help in the long run. However, since our homework deals with small, toy examples, the difference is not necessary but simply interesting.)

With this intuitive understanding in hand, the problem then prompts students to derive a closed-form expression for the recurrence as an alternative way of interpreting the diffusion process as a single combined step.

At the end of this question, students should have a good sense of the diffusion process and the effects of different choices of variance schedules.

2 Optimization Loss Function

The next written problem walks students through the optimization loss function used to train a neural net to approximate the forward diffusion process distribution. The reverse process also uses Gaussian transitions to achieve a predicted diffusion distribution. Intuitively, we would like to optimize our model by creating an expression for estimating the initial state for a given corrupted state. The most direct way to do this is to maximize the logarithmic likelihood of the denoised initial state at the end of the reverse process. However, this is computationally intractable, so the loss that is actually used in practice is one that is based on KL divergence. Each part of this problem is a building block toward deriving this KL divergence loss from our initial intuitive logarithmic likelihood loss.

The problem's introduction introduces the student to the variational bound, also known as the evidence lower bound, which is actually used as an upper bound in this case because the losses are negated. We give the student the bound without derivation (we assume that they have already encountered this during discussion). Part one of the problem builds upon this variational bound by expanding the expression into the sum of a logarithm of the probability of the completely diffused image and the sum of the logarithms of the probabilities associated with conditioning the image of the previous time step to the current time step.

The student must then convert this intermediate result into an expression that involves KL divergence. Since the derivation is relatively long, this part is split into two subparts. The first subpart is dedicated toward deriving an expression for $q(x_t|x_{t-1})$, which is an important intermediate expression for the KL divergence formula in the second subpart. The second subpart asks the student to use this intermediate to derive the final KL divergence loss, upon which the mean loss and noise loss functions are based in the later parts of this homework. After completing this problem, the student should have a much better sense of how and why we need to approximate the logarithmic likelihood loss with the KL surrogate loss and the loss we later derive from this.

3 Reverse “Denoising" Process

In practice, we don't directly compute the KL divergence loss; rather, the paper suggests two different approaches towards what loss the model should train on and therefore learn. One option is to train the model based on the MSE (mean squared error) between the mean of the forward process distribution and the predicted mean of the reverse process distribution. Alternatively, the other option is to train the model based on the MSE between target and predicted noise to obtain a model that learns the noise ϵ that was added to the original image during the forward diffusion process.

The purpose of this problem is to help students understand the equivalence between these two losses via parameterization and think about scenarios where we may want to use one over the other. We begin by presenting the squared mean loss as given in the paper:

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

where $\tilde{\mu}_t$ represents the mean of the distribution at time t that we are trying to predict, and μ_θ represents the predicted mean from the model that has parameters expressed as θ.

Given this mean loss, we first ask the student to reparameterize it in terms of x_0 and ϵ according to the following relation: $x_t(x_0, \epsilon) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ for $\epsilon \sim N(\mathbf{0}, \boldsymbol{I})$. The end result of this first part is a loss function that involves the noise variable ϵ.

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_t^2}{2{\sigma_t}^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right] + C \tag{3}$$

From this point, the second part of the question takes the student through a derivation of an expression that relies solely on ϵ, ϵ_θ, and x_0. This is the loss function the diffusion network uses to predict the noise for its reverse pass both in the paper and in our code implementations (Part 5).

4 Predicting Mean vs. Predicting Noise

Though the previous problem had the student walk through the steps of deriving the noise loss function from the mean loss, there's still the question of how these losses are implemented in practice or why we might choose one over another for training models since the parameterizations are equivalent. In this question, we explore one comparison between the two different loss functions: their computational efficiency and performance.

As pointed out in the paper, there are multiple options of loss functions and parameterizations available for this problem. Some, like the mean based loss function, focus on characterizing the distribution of the states through predicting the mean. Others, like the noise based loss function, focus on the diffused *noise* that gets added as part of the forward diffusion process.

Thus, for this question, students will turn to the `loss_func_comp` Jupyter notebook. The notebook guides the students through comparing the speed of the mean based loss and the noise based loss functions. In order to do so, the students must first implement both the mean and noise loss functions given the input parameters. We have provided autograded test cases with random inputs and expected outputs of dimension five to ensure the student's implementation is correct.

We then ask the student to run a speed comparison test on these two loss functions on a variety of random inputs with different vector dimensions ranging from 2 to 1000 with increments of 200. The student will find that, while the two loss functions are equivalent according to what was derived in problem 3, the mean loss function generally seems to be more expensive in the cases we are considering, compared to the noise loss function.

We mention in our solutions that the noise loss function also has the extra advantage of lending itself easily to further approximations or simplifications that can further speed up the training, but this is outside the scope of this homework, and we do not expect the student to fully understand this as this was just a footnote in the paper.

We also note in the solutions that the paper points out that their use of the noise loss function was motivated by better training performance compared to the mean loss function as well. Though students will likely not engage with this observation explicitly, the next question utilizes this noise loss for training, so students should be able to observe the results.

5 Code: Train and Observe

The last question of the homework puts together all the concepts introduced in the previous problems to create a working prototype of a denoising diffusion probabilistic model that can be used for generation. Here, students can note the following connections to the prior problems:

1. The closed-form expression for the forward process they derived in question 1.
2. As discussed in question 3, the model uses MSE loss rather than directly computing the KL divergence loss.
3. Specifically, the training loss we use in our implementation is based on comparing the actual noise added as part of the diffusion process with the predicted noise from the model. This idea of using the noise for the loss as opposed to the mean was introduced to students in question 4, and used here.

For the implementation aspect, we made a couple of design choices to better illustrate the key concepts that we wanted to highlight from the paper and for students to engage with through the `denoising` Jupyter notebook:

1. The paper uses CIFAR and other image datasets for training their large models, which they mention could take around 10 hours. Since we are interested in illustrating the possibilities and application of diffusion models in a way that students can more easily engage in, we opted to create a synthetic dataset and small dimensions of images (3 x 3) so training can run more quickly (in the range of a couple of minutes) and students can visualize results utilizing the `imshow` functionality from `matplotlib.pyplot`.

   Specifically, our training dataset consists of a single dot in the middle of the 3 x 3 image matrix. Even though we're passing in the same input each time to our model, we'll still end up with different and interesting results during each training loop due to the randomness that gets applied when both choosing t (the number of timesteps to run diffusion on in the input for) as well as when applying the Gaussian noise ϵ to each input in the diffusion process.

This also allows for the visualization to be more clear-cut, as students will be keeping an eye out for whether the generated images look like the single dot in the middle (a pattern that is hopefully easy to recognize).

2. We implement a U-Net style linear model to be used for predicting noise. This is inspired by the U-Net style model used in the paper, but with some key differences.

   For ease of implementation, we create this structure using fully connected layers instead of CNNs (Convolution Neural Networks). Additionally, we actually take on an "upside" down U-Net structure due to our inputs being very small (3 x 3 images) and in wanting to have enough expressiveness in our model to learn interesting behavior. As a result, rather than the classic U-Net structure that involves downsampling and then upsampling, here, we upsample our image to some larger dimensions before downsampling back to the dimensions of our original 3 x 3 images.

3. Finally, in our visualization section, we consider two main visualizations.

   First, generating from randomly sampled Gaussian noise using our model. This resembles the sampling process (Algorithm 2) as described in the paper.

   Second, we also have students visualize the single dot training image as x_0, then the states after a *single* diffusion timestep as x_1, and finally the reconstructed inputs from the model running the reverse denoising process as $\hat{x}_0$. The goal of this second visualization is to show how well the model can reconstruct from x_t to x_{t-1} (where $t = 1$ in this case), which is the goal of the training loop.

   We chose not to look at reconstruction over more timesteps, since asking the model to progressively denoise over multiple timesteps and recover the original image becomes more and more difficult due to the noisy denoiser and the noise of the forward process.

One of the takeaways from this question is that we'd like students to see that clearly, the denoising isn't perfect — the model and inputs are both relatively small compared to the ones mentioned in the paper and other parts of the field. Still, even with this small, locally-run, non-CNN network (or Colab-run if the student so wishes), we are able to generate some samples that look approximately like the initial interesting distribution. The network is still capable of learning some very important characteristics of denoising. We're able to generate some interesting images, ultimately!

6 Takeaways

We hope that from introducing the concepts of diffusion models, to exploring different parameterizations of loss functions, and to finally visualizing generated samples from a small, trained model, that students can walk away with a better understanding of all these ideas. As students ourselves, we certainly also learned a lot in the process of generating these new samples of diffusion model problems!