Diffusion Notes

Eugene Francisco

March 2026

Sections 1 through 5 are heavily inspired by Turner in https://arxiv.org/pdf/2402.04384.

1 Setup:

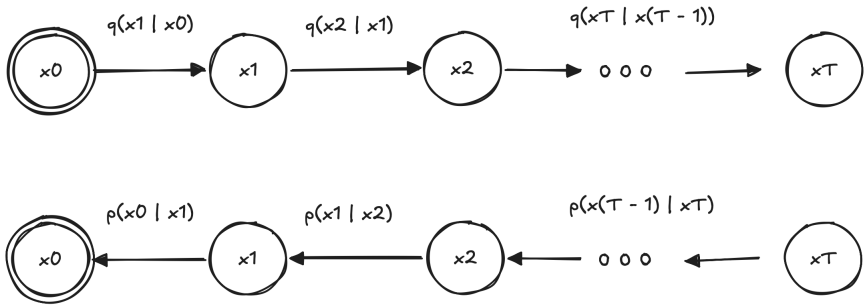
We are given a data set $\mathcal{X}$ of data sampled from $q(\cdot)$. We want to sample new data from q and in particular we want to sample data that hasn't been seen before from $\mathcal{X}$. Diffusion does this by denoising an augmented dataset

$$x^{(0)}, \ldots, x^{(T)}$$

where $x^{(0)}$ comes from the original dataset and $x^{(T)}$ is some much lower fidelity version of the original dataset. We fix in advance a noising algorithm which takes in a datapoint $x^{(t-1)}$ and outputs a lower fidelity datapoint $x^{(t)}$. We denote $q(x^{(t)}|x^{(t-1)})$ the density of $x^{(t)}$ conditioned on $x^{(t-1)}$. Further we fix $q(x^{(0)})$ to be just sampling a datapoint in the training set at random. To accomplish our goal, we will approximate q with a parametrized distribution p from which data will ultimately be generated.

Ideally, coming up with p will be easier as long as the following 2 things to be true:

1. The lowest fidelity data $x^{(T)}$ is easy to sample from directly. I.e., the distribution $q(x^{(T)})$ is a simple distribution.
2. Predicting a higher fidelity level from a lower fidelity level is easy to do. I.e., $q(x^{(t-1)} \mid x^{(t)})$ should be a distribution that is simple to learn/sample from.



A note on notation: we use qs to denote the *true* distribution while ps will denote the distributions that we are using to approximate q.

Choosing the Noisers:

For now, we will choose $q(x^{(t)} \mid x^{(t-1)})$ to be $\mathcal{N}(\lambda_t x^{(t-1)}\ ,\ \sigma_t^2)$ where λ_t is close to 1. To satisfy condition 1 above, it will help us to pick $\sigma_t^2 = 1 - \lambda_t^2$. If this choice is made, then we have the following property.

Lemma 1.1 *(Zero Mean and Unit Variance).* If we let $q(x^{(t)}|\ x^{(t-1)}) = \mathcal{N}(x^{(t)}; \lambda_t x^{(t-1)}, \sigma_t^2)$ and if we set $\sigma_t^2 = 1 - \lambda_t^2$, then

$$\mathbb{E}_{q(x^{(t)})}[x^{(t)}] = 0 \qquad \mathbb{E}_{q(x^{(t)})}\left[(x^{(t)})^2\right] = 1. \tag{1.2}$$

Furthermore

$$q(x^{(t)}|x^{(0)}) = \mathcal{N}\left(x^{(t)}, \left(\prod_{t'=1}^{t} \lambda_{t'}\right) x^{(0)}, 1 - \prod_{t'=1}^{t} \lambda_{t'}^2\right). \tag{1.3}$$

Proof:

Turner appendex 4.1.1 and 4.1.2.

Remark 1.3 If $\lambda_t < 1$ observe that as $T \to \infty$ we have $q(x^{(T)}|x^{(0)}) \to \mathcal{N}(x^{(T)}; 0, 1)$. This means that, as we pick larger and larger T, the distribution of $x^{(T)}$ approaches that of a standard normal, thus satisfying desire 1 above. Observe also that this did not require $q(x^{(0)})$ to be normal.

2 Denoising:

We now have a way of going from the original high fidelity $x^{(0)}$ to low fidelity $x^{(T)}$. In the first place we want to learn $q(x^{(0)})$ and we will do this by learning the parameters for a parametrized estimator p_θ. For now, we will imagine that we have different parameters for each level of the denoising process, so $p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)})$ is an approximation of $q(x^{(t-1)}|x^{(t)})$. To remove clutter, when the context is clear, we'll drop the subscript on θ and use $p_\theta(x^{(t-1)}|x^{(t)})$ instead.

To find θ, we can maximize the log-likelihood of the augmented dataset that was generated using q. You may think that this is

$$\begin{aligned}\theta^* &\stackrel{?}{=} \operatorname*{argmax}_\theta \log p_\theta(x^{(0)}) \\ &= \operatorname*{argmax}_\theta \log\left(p(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)}|x^{(t)})\right).\end{aligned} \tag{2.1}$$

Note that Remark 1.3 has allowed us to drop the parameters from $p(x^{(T)})$, which should be approximately $\mathcal{N}(0, 1)$. Equation (2.1) is almost right; however, recall that the augmented data were generated randomly with q. We don't just want to maximize the likelihood with respect to this particular augmented dataset; we really want to do it over the randomness that noising introduces as well. In other words, we want to maximize the *likelihood in expectation over* q:

$$\theta^* = \operatorname*{argmax}_\theta \mathbb{E}_q\left[\log\left(p(x^{(T)}) \prod_{t=1}^{T} p_\theta(x^{(t-1)}|x^{(t)})\right)\right]. \tag{2.2}$$

$$= \operatorname*{argmax}_\theta \mathbb{E}_q\left[\log\left(p(x^{(T)}) \prod_{t=1}^{T} p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)})\right)\right] \tag{2.3}$$

$$= \operatorname*{argmax}_\theta \left[\underbrace{\mathbb{E}_{q(x^T)}\left[\log(p(x^{(T)}))\right]}_{\star} + \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x^{(t-1)}, x^{(t)})}\left[\log\left(p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)})\right)\right]}_{\triangle}\right]. \tag{2.4}$$

A few things here. In (2.2), the expectation was over the joint $q(x^{(0:T)})$. But in $\star$, notice that $x^{(T)}$ has randomness w.r.t $q(x^{(T)})$ alone (the rest has been integrated out). Similarly, in $\triangle$, notice that the expectation is a joint only over $q(x^{(t-1)}, x^{(t)})$ alone.

Looking at (2.4), it becomes clear that all we need to maximize is

$$\mathbb{E}_{q(x^{(t-1)},x^{(t)})}\left[\log\left(p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)})\right)\right] \tag{2.5}$$

for each fidelity level $t \in \{T, \dots, 1\}$.

Choosing p_θ and amortizing

How should we maximize the expression in (2.4)? We first must choose what p_θ is. Since $p(x^{(T)}) = \mathcal{N}(x^{(T)}; 0, 1)$ and each noising step is Gaussian, a natural choice is for

$$p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)}) = \mathcal{N}(x^{(t-1)}|\mu_{\theta_{t-1}}(x^{(t)}), \sigma^2_{\theta_{t-1}}(x^{(t)})) \tag{2.5}$$

where $\mu_{\theta_{t-1}}(x^{(t)})$ and $\sigma^2_{\theta_{t-1}}(x^{(t)})$ are mean and variance estimates for $p_{\theta_{t-1}}$ parametrized by θ_{t-1}. In other words, our only task is to estimate the mean and variance of $q(x^{(t-1)}|x^{(t)})$. Suppose we were to do this and that θ_{t-1} has K parameters per fidelity jump. For example, we could use a DNN with $x^{(t)}$ as input, K parameters, and outputs the mean and variance of $p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)})$. The number of parameters would grow linearly with the fidelity depth, but can we make the number of parameters constant in the fidelity depth T?

One idea is to train a DNN that takes in as input $x^{(t)}$ *as well as the fidelity layer* $t-1$ that we are trying to produce; this DNN could then output for us μ and σ to use in (2.4). DNN or otherwise, this idea of using a shared set of parameters w to learn what θ_{t-1} should be is called amortizing, and crucially w does not scale with T. Assuming we have done this, and relabeling w as θ, the density in (2.5) becomes

$$p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)}) = \mathcal{N}(x^{(t-1)}|\mu_\theta(x^{(t)}, t-1), \sigma^2_\theta(x^{(t)}, t-1)). \tag{2.6}$$

to emphasize that the mean and variance estimates here come from one model that take in the datapoint and the fidelity level as input.

It is important to note here that the choice of $p_\theta(x^{(t-1)}|x^{(t)})$ as Gaussian does not mean that $q(x^{(t-1)}|x^{(t)})$ is Gaussian; remember that $q(x^{(0)})$ need not be Gaussian! The hope is that the mixture of Gaussians from the layers of p_θ will approximate q.

Maximizing our objective

Assuming the model in (2.6), equation (2.5) can be expanded to

$$-\frac{1}{2}\mathbb{E}_{q(x^{(t-1)},x^{(t)})}\left[\frac{\left[x^{(t-1)} - \mu_\theta(x^{(t)}, t-1)\right]^2}{\sigma^2_\theta(x^{(t)}, t-1)} + \log 2\pi\sigma^2_\theta(x^{(t)}, t-1)\right]. \tag{2.7}$$

A practical and naive way to sample an unbiased estimate for this quantity is to sample $x^{(0)}$ from our dataset (recall this is how we defined $q(x^{(0)})$) and then auto-regressively sample $q(x^{(\ell)}|x^{(\ell-1)})$. Once we reach t, we can estimate the quantity in (2.7) and take gradients w.r.t θ. Doing so requires on the order of t samples. However, doing so has the disadvantage of inheriting high estimator variance from the autoregressive sampling.

It turns out that by rewriting the objective in (2.7) slightly, we can get away with sampling only $x^{(t)} \sim q(x^{(t)})$ whose distribution we know from Lemma 1.1. In particular, using the tower law,

$$\begin{aligned}\mathbb{E}_{q(x^{(t-1)},x^{(t)})}\left[f(x^{(t-1)}, x^{(t)})\right] &= \mathbb{E}_{q(x^{(0)},x^{(t-1)},x^{(t)})}\left[f(x^{(t-1)}, x^{(t)})\right] \\ &= \mathbb{E}_{q(x^{(0)},x^{(t)})}\left[\mathbb{E}_{q(x^{(t-1)}|x^{(0)},x^{(t)})}\left(f(x^{(t-1)}, x^{(t)})\right)\right]\end{aligned} \tag{2.8}$$

This may look scarier than what we had before. But consider the inner expectation and $q(x^{(t-1)}|x^{(0)}, x^{(t)})$. By Bayes rule,

$$q(x^{(t-1)}|x^{(0)}, x^{(t)}) = \frac{q(x^{(0)}, x^{(t-1)}, x^{(t)})}{q(x^{(0)}, x^{(t)})}$$

and the RHS are all quantities that we can calculate since $x^{(\ell)}$ depends only on $x^{(\ell-1)}$ (first order Markov) and each of the $q(x^{(\ell)}|x^{(\ell-1)})$ are Gaussian. Rewritten, our objective in (2.7) becomes

$$\mathbb{E}_{q(x^{(0)},x^{(t)})}\Big[\mathbb{E}_{q(x^{(t-1)}|x^{(0)},x^{(t)})}\left(\frac{\left[x^{(t-1)} - \mu_\theta(x^{(t)}, t-1)\right]^2}{\sigma_\theta^2(x^{(t)}, t-1)} + \log 2\pi\sigma_\theta^2(x^{(t)}, t-1)\right)\Big].$$

You will recognize that, using linearity of expectation, the inner expectation here is an expectation of a Gaussian pdf w.r.t a Gaussian measure, which means it is Gaussian. In particular (skipping all the algebra) the inner expectation can be replaced with its closed form solution as

$$\mathbb{E}_{q(x^{(0)},x^{(t)})}\left[\frac{\left[\mu_{t-1|0,t} - \mu_\theta(x^{(t)}, t-1)\right]^2 + \sigma_{t-1|0,t}^2}{\sigma_\theta^2(x^{(t)}, t-1)} + \log 2\pi\sigma_\theta^2(x^{(t)}, t-1)\right] \tag{2.9}$$

where

$$q(x^{(t-1)}|x^{(0)}, x^{(t)}) = \mathcal{N}(x^{(t-1)}; \mu_{t-1|0,t}, \sigma_{t-1|0,t}^2)$$

and $\mu_{t-1|0,t}$ and $\sigma_{t-1|0,t}^2$ are known functions of $x^{(0)}, x^{(t)}$ that can be computed beforehand (more on this in the next section). Rewritten in the form of (2.9), we have two training advantages.

(i) The first is that we can sample $x^{(t)}$ directly from $x^{(0)}$, skipping the linear cost for fidelity sampling.

(ii) The second and more important is that **the inner expectation in (2.8) had a completely closed form solution once $x^{(0)}$ and $x^{(t)}$ were picked**

which reduces one layer of variance from our estimate here. In other words, the variance of our estimate came only from $x^{(0)}$ and $x^{(t)}$!

3 Closed form posteriors

Assume that we have chosen noise functions as in section 3. Recall from section 2 that to maximize the likelihood we calculated analytically $\mu_{t-1|0,t}$ and $\sigma_{t-1|0,t}^2$ and claimed that these had closed form solutions with a high level Bayes rule justification. The form of these closed form solutions will be helpful to us later, so here they are written out without proof and taken from Turner

$$\begin{aligned}
\mu_{t-1|0,t} &= a^{(t-1)}x^{(0)} + b^{(t-1)}x^{(t)} \\
a^{(t-1)} &= \frac{\left(\prod_{t'=1}^{t-1}\lambda_{t'}(1-\lambda_t^2)\right)}{1-\prod_{t'=1}^{t}\lambda_{t'}2} \\
b^{(t-1)} &= \frac{\left(1-\prod_{t'=1}^{t-1}\lambda_{t'}^2\right)\lambda_t}{1-\prod_{t'=1}^{t}\lambda_{t'}^2} \\
\sigma_{t-1|0,t}^2 &= \frac{\left(1-\prod_{t'=1}^{t-1}\lambda_{t'}^2\right)(1-\lambda_t^2)}{1-\prod_{t'=1}^{t}\lambda_{t'}^2}
\end{aligned} \tag{3.1}$$

4 Picking models to predict denoising mean and variance

Recall from equation (2.9) that we had estimates

$$\mu_\theta(x^{(t)}, t-1) \qquad \sigma^2_\theta(x^{(t)}, t-1) \tag{4.1}$$

which were estimates for the mean and variance of $p_\theta(x^{(t-1)}|x^{(t)})$ which we assumed was Gaussian. The estimates in (4.1) should be read as functions whose inputs are $x^{(t)}$ and $t-1$, whose parameters are θ, and whose output is the mean and variance estimate respectively. How should we pick the model for these estimators?

Mean estimate 1

The first option is taken from Turner's Mean Option 1 and takes inspiration from the closed form solutions from (4.1). Recall that Section 3 told us that the conditional expectation of $x^{(t-1)}$ given $x^{(t)}$ and $x^{(0)}$ was

$$\mu_{t-1|0,t} = a^{(t-1)}x^{(0)} + b^{(t-1)}x^{(t)}. \tag{4.2}$$

We have access to $a^{(t-1)}$ and $b^{(t-1)}$ via closed forms and $x^{(t)}$ is the data we are given. All that is missing to get the optimal guess is $x^{(0)}$. One idea which may seem silly at first is to have a model predict $\hat{x}^{(0)}_\theta(x^{(t)}, t-1)$ which we hope is approximately $x^{(0)}$ and then plug this into (4.2)! This model has two advantages: the first is that the model only has to predict one thing for all layers: $\hat{x}^{(0)}_\theta$. The second is that the model output can be composed as a simple set of residual connections from previous $x^{(t)}$.

Mean Estimate 2

Also taken from Turner. Recall from Lemma 3.2 that

$$x^{(t)} = \left(\prod_{t'=1}^{t} \lambda_{t'}\right) x^{(0)} + \left(\sqrt{1 - \prod_{t'=1}^{t} \lambda^2_{t'}}\right) \epsilon \qquad \epsilon \sim \mathcal{N}(0,1).$$

Let $c^{(t)} = \left(\prod_{t'=1}^{t} \lambda_{t'}\right)$ and $d^{(t)} = \left(\sqrt{1 - \prod_{t'=1}^{t} \lambda^2_{t'}}\right)$. Rearranging to isolate $x^{(0)}$ yields

$$x^{(0)} = \frac{x^{(t)} - d^{(t)}\epsilon}{c^{(t)}}.$$

Substituting this back into (4.2) yields

$$\mu_{t-1|0,t} = a^{(t-1)}\left(\frac{x^{(t)} - d^{(t)}\epsilon^{(t)}}{c^{(t)}}\right) + b^{(t-1)}x^{(t)}. \tag{4.3}$$

The only quantity here that we don't know is $\epsilon^{(t)}$ and so another idea is to train

$$\hat{\epsilon}^{(t)}_\theta(x^{(t)}, t-1) \approx \epsilon^{(t)} \tag{4.4}$$

and then substitute this into (4.3) to get our mean estimate.

Variance Option

Turner claims that variance is typically hard to estimate and it is not unusual to fix $\sigma^2_\theta(x^{(t)}, t-1)$ to something in advance. For the rest of this presentation, we will use v_t to denote that fixed variance at each denoising step. Some common choices described by Turner are

$$v_t = 1 - \lambda_t^2$$

which is the variance during the single step noising procedure. Another choice is

$$v_t = \frac{1-(c^{(t-1)})^2}{1-(c^{(t)})^2}(1-\lambda_t^2)$$

which is the variance of $q(x^{(t-1)}|x^{(t)}, x^{(0)})$.

5 DDPM Framework

Here is a description of the DDPM framework placed in terms of the formulation above. The DDPM framework is the style that learns an estimator $\hat{\epsilon}_\theta^{(t)}(x^{(t)}, t-1)$ like in equation (5.4). Remember that the first parameter of this estimator is the low fidelity data that we choose. In DDPM, we choose

$$x^{(t)} = \Lambda_t x^{(0)} + \sqrt{1-\Lambda_t^2}\epsilon^{(t)}$$

where $\epsilon^{(t)} \sim \mathcal{N}(0,1)$ and $\Lambda_t = \prod_{t'=1}^{t} \lambda_{t'}$. Since this formulation of estimating $\mu_{t-1|0,t}$ involves making $\hat{\epsilon}_\theta^{(t)}$ as close as possible to $\epsilon^{(t)}$, the training objective for DDPM is

$$\mathcal{L}(\theta) = -\frac{T}{2}\mathbb{E}_{\substack{t\sim\text{Uniform}(1,T)\\ x^{(0)}\sim\mathcal{X}\\ \epsilon\sim\mathcal{N}(0,1)}}\left[\epsilon^{(t)} - \hat{\epsilon}_\theta^{(t)}\left(\Lambda_t x^{(0)} + \sqrt{1-\Lambda_t^2}\epsilon^{(t)}, t-1\right)\right].$$

The expectation here is taken over $t \sim \text{Uniform}(1,T)$. In other words, we create $x^{(t)}$ using $\epsilon^{(t)}$ and then try to predict $\epsilon^{(t)}$.

6 Label Conditioning

Often we are less interested in generating data from $q(x)$ directly but instead more interested in sampling from $q(x|z)$ where z is a latent variable that directs what the generation should look like. For image diffusion for example z could be a label for the kind of image that should be generated. How does this change the setup of our diffusion?

First, the dataset changes to a set of pairs $\{x^{(0)}, z\} = \mathcal{X}$. Assuming that we perform the same noising process (with noise added independent of z), the new log-likelihood becomes

$$\theta^* = \underset{\theta}{\text{argmax}}\left[\mathbb{E}_{q(x^T, z)}\left[\log(p(x^{(T)}))\right] + \sum_{t=1}^{T}\mathbb{E}_{q(x^{(t-1)}, x^{(t)}, z)}\left[\log\left(p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)}, z)\right)\right]\right]. \quad (6.1)$$

Observe that the randomness is now over both x and z. As before, we model $p_{\theta_{t-1}}(x^{(t-1)}|x^{(t)}, z)$ so that we only have to predict the mean estimate

$$\mu_\theta(x^{(t)}, t-1, z)$$

and to do so we can either use the direct $\hat{x}^{(0)}|z$ estimate or the $\hat{\epsilon}^{(t)}|z$ estimate, now conditioning on z as well. For example, the updated DDPM training objective becomes

$$\mathcal{L}(\theta) = -\frac{T}{2}\mathbb{E}_{\substack{t\sim\text{Uniform}(1,T)\\ (x^{(0)},z)\sim\mathcal{X}\\ \epsilon\sim\mathcal{N}(0,1)}}\left[\epsilon^{(t)} - \hat{\epsilon}_\theta^{(t)}\left(\Lambda_t x^{(0)} + \sqrt{1-\Lambda_t^2}\epsilon^{(t)}, t-1, z\right)\right].$$

7 Inference

Once our model is trained, how do we actually go about generating data? After training, we are armed with distributions $p_{\hat{\theta}}(x^{(t-1)}|x^{(t)}, z)$ for $t \in \{1, \dots, T\}$. A naive approach to generate data would be the following:

(i) Sample $x^{(T)} \sim \mathcal{N}(0, 1)$.

(ii) For $t \in \{T, \dots, 1\}$:

  (a) Set $x^{(t-1)} \leftarrow \mu_{\hat{\theta}}(x^{(t)}, t-1, z)$.

(iii) Return $x^{(0)}$.

It turns out that this approach does not work well in practice. Although outputting the conditional expectation is optimal in the regression problem, it leads to "blurry" generations in practice. Intuitively, this can be thought of as predicting the "average" of an image which may not have that much detail. One solution is to add stochasticity to the generation process. This gives us the following.

Data Generation Algorithm 1 Given latent z:

(i) Sample $x^{(T)} \sim \mathcal{N}(0, 1)$.

(ii) For $t \in \{T, \dots, 1\}$:

  (a) Set $\mu_t \leftarrow \mu_{\hat{\theta}}(x^{(t)}, t-1, z)$.

  (b) Sample $\xi_t \sim \mathcal{N}(0, 1)$.

  (c) Set $x^{(t-1)} = \mu_t + \sqrt{v_t}\xi_t$.

(iii) Return $x^{(0)}$.