Chapter 18

Diffusion models

Chapter 15 described generative adversarial models, which generate plausible looking samples but do not define a probability distribution over the data. Chapter 16 discussed normalizing flows, which can assign a probability but which place architectural constraints on the network; each layer must be invertible and the determinant of its Jacobian must be easy to calculate. Chapter 17 introduced variational autoencoders, which also have a solid probabilistic foundation, but where computation of the likelihood is intractable and but can be approximated by a lower bound.

This chapter introduces diffusion models. Like normalizing flows, these are probabilistic models that define a nonlinear mapping from latent variables to the observed data where both quantities have the same dimension. Like variational autoencoders they approximate the data likelihood using a lower bound based on an encoder that maps *to* the latent variable. However, in diffusion models this encoder is predetermined; the goal is to learn a decoder that is the inverse of this process and can be used to produce samples. Diffusion models are easy to train and can produce very high quality samples, that exceed the realism of those produced by GANs. The reader should be familiar with variational autoencoders (chapter 17) before reading this chapter.

18.1 Overview

A diffusion model consists of an *encoder* and a *decoder*. The encoder takes a data sample $\mathbf{x}$ and maps it through a series of intermediate latent variables $\mathbf{z}_1 \ldots \mathbf{z}_T$. The decoder reverses this process; it starts with $\mathbf{z}_T$ and maps back through $\mathbf{z}_{T-1}, \ldots, \mathbf{z}_1$ until it finally (re-)creates a data point $\mathbf{x}$. In both encoder and decoder, the mappings are stochastic rather than deterministic.

The encoder is predetermined; it gradually blends the input with samples of white noise (figure 18.1). With enough steps, the conditional distribution $q(\mathbf{z}_T|\mathbf{x})$ and marginal distribution $q(\mathbf{z}_T)$ of the final latent variable both become the standard normal distribution. Since this process is pre-specified, all the learned parameters are in the decoder.

In the decoder, a series of networks are trained to map backwards between each

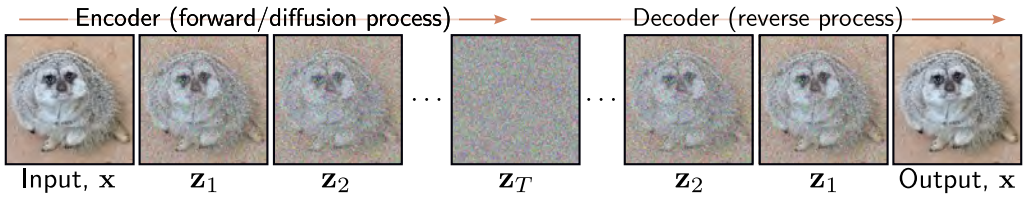

Figure 18.1 Diffusion models. The encoder (forward, or diffusion process) maps the input $\mathbf{x}$ through a series of latent variables $\mathbf{z}_1 \ldots \mathbf{z}_T$. This process is predetermined and gradually mixes the data with noise until only noise remains. The decoder (reverse process) is learned and passes the data back through the latent variables, removing noise at each stage. After training, new examples are generated by sampling noise vectors $\mathbf{z}_T$ and passing them through the decoder.

adjacent pair of latent variables $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$. The loss function encourages each network to invert the corresponding decoder step. The result is that noise is gradually removed from the representation until a realistic looking data example remains. To generate a new data example $\mathbf{x}$, we draw a sample from $q(\mathbf{z}_T)$ and pass it through the decoder.

In section 18.2, we consider the encoder in detail. Its properties are non-obvious, but are critical for the learning algorithm. In section 18.3 we discuss the decoder. Section 18.4 derives the training algorithm and section 18.5 reformulates it to be more practical. Section 18.6 discusses implementation details, including how to make the generation conditional on text prompts.

18.2 Encoder (forward process)

The *diffusion* or *forward* process[1] (figure 18.2) maps a data example $\mathbf{x}$ through a series of intermediate variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ with the same size as $\mathbf{x}$ according to:

$$\begin{aligned} \mathbf{z}_1 &= \sqrt{1-\beta_1}\mathbf{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \\ \mathbf{z}_t &= \sqrt{1-\beta_t}\mathbf{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t \qquad \forall\, t \in 2, \ldots, T, \end{aligned} \tag{18.1}$$

where $\boldsymbol{\epsilon}_t$ is noise drawn from a standard normal distribution. The first term attenuates the data plus any noise added so far, and the second adds more noise. The hyperparameters $\{\beta_t\} \in [0, 1]$ determine how quickly the noise is blended and are collectively known as the *noise schedule*. This can equivalently be written as:

[1]Note, this is the opposite nomenclature to normalizing flows, where the inverse mapping moves from the data to the latent variable, and the forward mapping moves back again.

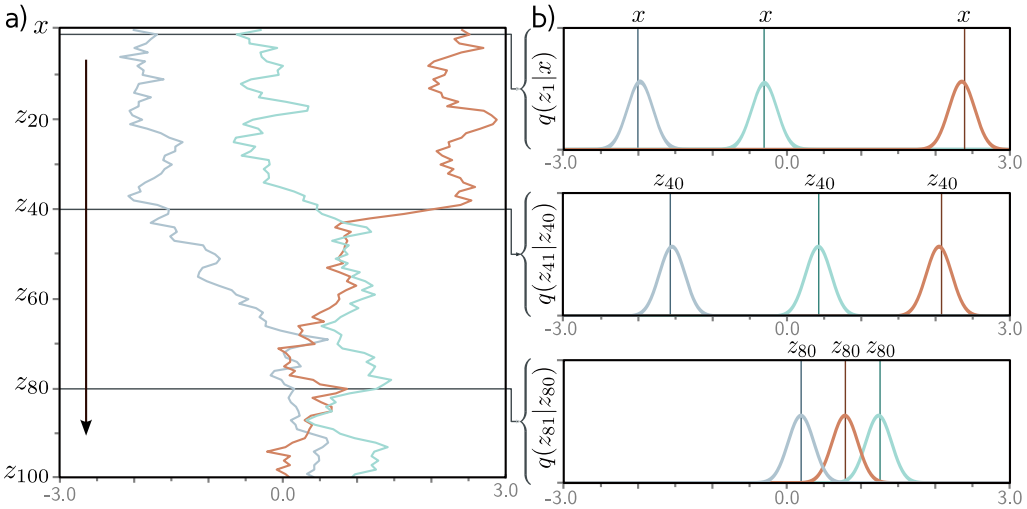

Figure 18.2 Forward process. a) We consider one-dimensional data x with $T = 100$ latent variables $z_1, \ldots, z_{100}$ and a $\beta = 0.03$ at all steps. Three values of x (orange, green, and gray) are initialized (top row). These are propagated through $z_1, \ldots, z_{100}$. At each step, the variable is updated by attenuating its value by $\sqrt{1-\beta}$ and adding noise with variance β (equation 18.1). Accordingly, the three examples noisily propagate through the variables with a tendency to move towards zero. b) The conditional probabilities $Pr(z_1|x)$ and $Pr(z_t|z_{t-1})$ are normal distributions with a mean that is slightly closer to zero than the current point and a fixed variance β_t (equation 18.2).

$$\begin{aligned} q(\mathbf{z}_1|\mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}\left[\sqrt{1-\beta_1}\mathbf{x}, \beta_1\mathbf{I}\right] \\ q(\mathbf{z}_t|\mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}\left[\sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}\right] \qquad \forall\, t \in 2, \ldots, T. \end{aligned} \tag{18.2}$$

With sufficient steps T, all traces of the original data are removed, and $q(\mathbf{z}_T|\mathbf{x}) = q(\mathbf{z}_T)$ becomes a standard normal distribution. Problem 18.1

The joint distribution of all of the latent variables $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$ given input $\mathbf{x}$ is:

$$q(\mathbf{z}_{1\ldots T}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}). \tag{18.3}$$

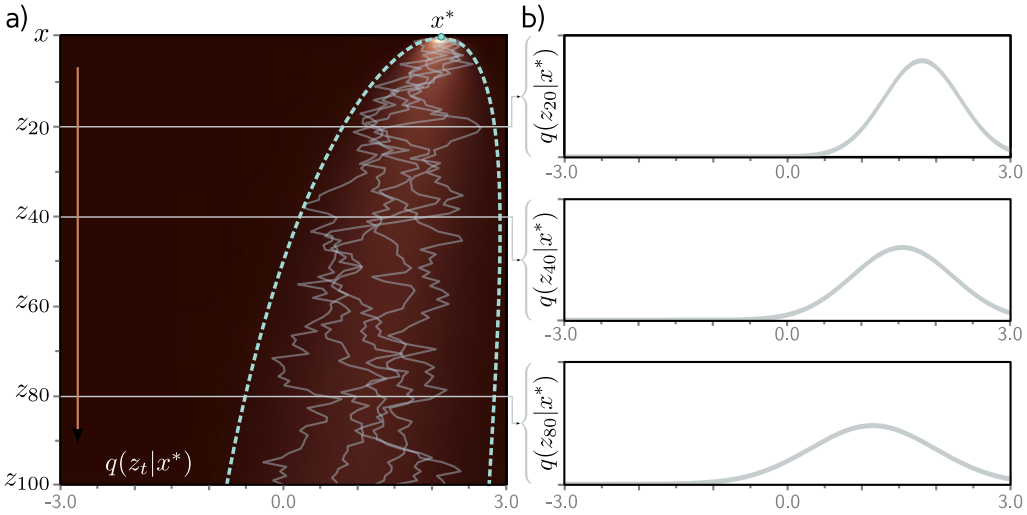

Figure 18.3 Diffusion kernel. a) The point $x^* = -2.0$ is propagated through the latent variables using equation 18.1 (five paths shown in gray). The diffusion kernel $q(z_t|x^*)$ is the probability distribution over variable z_t given that we started from x^*. It can be computed in closed-form and is a normal distribution whose mean moves towards zero and whose variance increases as t increases. Heatmap shows $q(z_t|x^*)$ for each variable. Cyan lines show ± 2 standard deviations of the normal. b) The diffusion kernel $q(z_t|x^*)$ is shown explicitly for $t = 20, 40, 80$. In practice, this means its easy to sample a latent variable z_t corresponding to a given x^* without computing the intermediate variables $z_1, \ldots, z_{t-1}$.

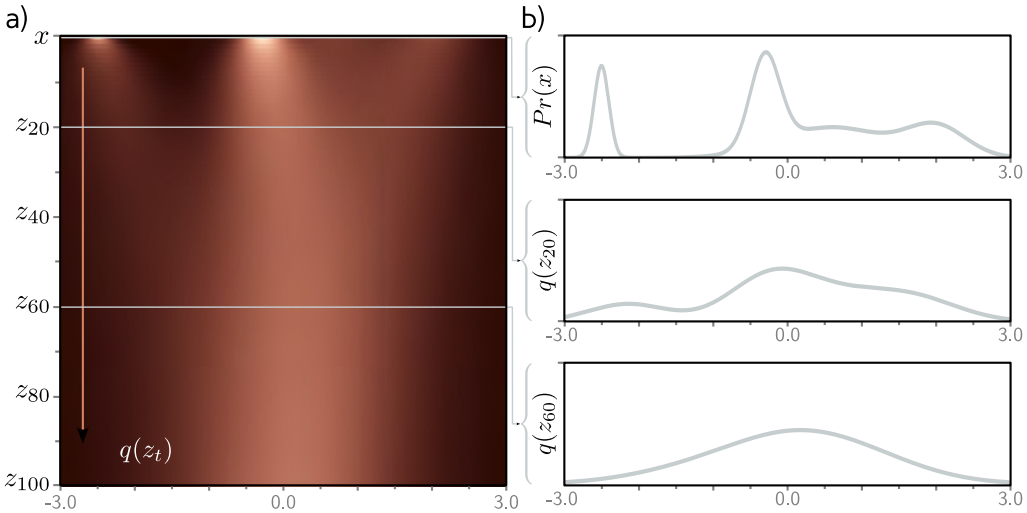

Figure 18.4 Marginal distributions. a) Given an initial density $Pr(x)$ with mean zero and standard deviation one, the diffusion process gradually blurs the distribution as it passes through the latent variables z_t and moves it towards a standard normal distribution. Each horizontal line of heatmap represents marginal distribution $Pr(x)$ (top row) or $q(z_t)$ (remaining rows). b) The top graph shows the initial distribution $Pr(x)$. The other two graphs show the marginal distributions $q(z_{20})$ and $q(z_{60})$, respectively.

18.2.1 Diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$

To train the decoder to invert this process, we will use multiple samples $\mathbf{z}_t$ for the same example $\mathbf{x}$. However, generating these sequentially using equation 18.1 is time-consuming when t is large. Fortunately, there is a closed-form expression for $q(\mathbf{z}_t|\mathbf{x})$, which allows us to directly draw samples $\mathbf{z}_t$ given initial data point $\mathbf{x}$ without computing the intermediate variables. This is known as the *diffusion kernel* (figure 18.3).

To derive an expression for $q(\mathbf{z}_t|\mathbf{x})$, consider the first two steps of the forward process:

$$\begin{aligned}\mathbf{z}_1 &= \sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1 \\ \mathbf{z}_2 &= \sqrt{1-\beta_2}\cdot\mathbf{z}_1 + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2.\end{aligned} \tag{18.4}$$

Substituting the first equation into the second, we get:

$$\begin{aligned}\mathbf{z}_2 &= \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{\beta_1}\cdot\boldsymbol{\epsilon}_1\right) + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2 \\ &= \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1}\cdot\mathbf{x} + \sqrt{1-(1-\beta_1)}\cdot\boldsymbol{\epsilon}_1\right) + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2 \\ &= \sqrt{(1-\beta_2)(1-\beta_1)}\cdot\mathbf{x} + \sqrt{1-\beta_2-(1-\beta_2)(1-\beta_1)}\cdot\boldsymbol{\epsilon}_1 + \sqrt{\beta_2}\cdot\boldsymbol{\epsilon}_2.\end{aligned} \tag{18.5}$$

The last two terms are independent samples from mean zero normal distributions with variances $1-\beta_2-(1-\beta_2)(1-\beta_1)$ and β_2, respectively. The mean of this sum is zero, and its variance is the sum of the component variances (see problem 18.2), so:

Problem 18.2

$$\mathbf{z}_2 = \sqrt{(1-\beta_2)(1-\beta_1)}\cdot\mathbf{x} + \sqrt{1-(1-\beta_2)(1-\beta_1)}\cdot\boldsymbol{\epsilon}, \tag{18.6}$$

where $\boldsymbol{\epsilon}$ is also a sample from a standard normal distribution.

If we continue this process by substituting this equation into the expression for $\mathbf{z}_3$ and so on, we can show that:

Problem 18.3

$$\mathbf{z}_t = \sqrt{\alpha_t}\cdot\mathbf{x} + \sqrt{1-\alpha_t}\cdot\boldsymbol{\epsilon}, \tag{18.7}$$

where $\alpha_t = \prod_{s=1}^{t} 1-\beta_s$. We can equivalently write this in probabilistic form:

$$q(\mathbf{z}_t|\mathbf{x}) = \text{Norm}_{\mathbf{z}_t}\left[\sqrt{\alpha_t}\cdot\mathbf{x}, (1-\alpha_t)\mathbf{I}\right]. \tag{18.8}$$

For any starting data point $\mathbf{x}$, variable $\mathbf{z}_t$ is normally distributed with a known mean and variance. Consequently, if we don't care about the history of the evolution through the intermediate variables, it is easy to generate samples from $q(\mathbf{z}_t|\mathbf{x})$.

18.2.2 Marginal distributions $q(\mathbf{z}_t)$

The marginal distribution $q(\mathbf{z}_t)$ is the probability of observing a value of $\mathbf{z}_t$ given the distribution of possible starting points $\mathbf{x}$ and the possible diffusion paths for each starting

point (figure 18.4). It can be computed by considering the joint distribution $q(\mathbf{x}, \mathbf{z}_{1\ldots t})$ and marginalizing over all the variables except $\mathbf{z}_t$:

$$\begin{aligned} q(\mathbf{z}_t) &= \int\int q(\mathbf{z}_{1\ldots t}, \mathbf{x}) d\mathbf{z}_{1\ldots t-1} d\mathbf{x} \\ &= \int\int q(\mathbf{z}_{1\ldots t}|\mathbf{x}) Pr(\mathbf{x}) d\mathbf{z}_{1\ldots t-1} d\mathbf{x}, \end{aligned} \tag{18.9}$$

where $q(\mathbf{z}_{1\ldots t}|\mathbf{x})$ was defined in equation 18.3.

However, since we now have an expression for the diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$ that "skips" the intervening variables, we can equivalently write:

$$q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{x}) Pr(\mathbf{x}) d\mathbf{x}. \tag{18.10}$$

Hence, if we sample repeatedly from the data distribution $Pr(\mathbf{x})$, and superimpose the diffusion kernel $q(\mathbf{z}_t|\mathbf{x})$ on each sample, the result is the marginal distribution $q(\mathbf{z}_t)$ (figure 18.4). However, the marginal distribution cannot be written in closed from because we don't know the original data distribution $Pr(\mathbf{x})$.

18.2.3 Conditional distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$

We defined the conditional probability $q(\mathbf{z}_t|\mathbf{z}_{t-1})$ as the mixing process (equation 18.2). To reverse this process we apply Bayes' rule:

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}) q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}. \tag{18.11}$$

This is intractable since we cannot compute the marginal distribution $q(\mathbf{z}_{t-1})$. This matches our intuition; we are mixing the original data example with noise at each stage, and there is no way to reverse this unless we knew the starting point.

For this simple 1D example, it's possible to evaluate these distributions numerically (figure 18.5). In general, their form is complex, but in many cases they are well-approximated by a normal distribution. This is important because when we build the decoder, we will approximate the reverse process using a normal distribution.

18.2.4 Conditional diffusion distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$

There is one final distribution related to the encoder to consider. We noted above that we could not find the conditional distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ because we do not know the marginal distribution $q(\mathbf{z}_{t-1})$. However, if we know the starting variable $\mathbf{x}$, then we *do* know the distribution $q(\mathbf{z}_{t-1}|\mathbf{x})$ at the time before. This is just the diffusion kernel (figure 18.3) and it is normally distributed.

Hence, it is possible to compute the conditional diffusion distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ in closed form (figure 18.6): This distribution is used to train the decoder. It is the

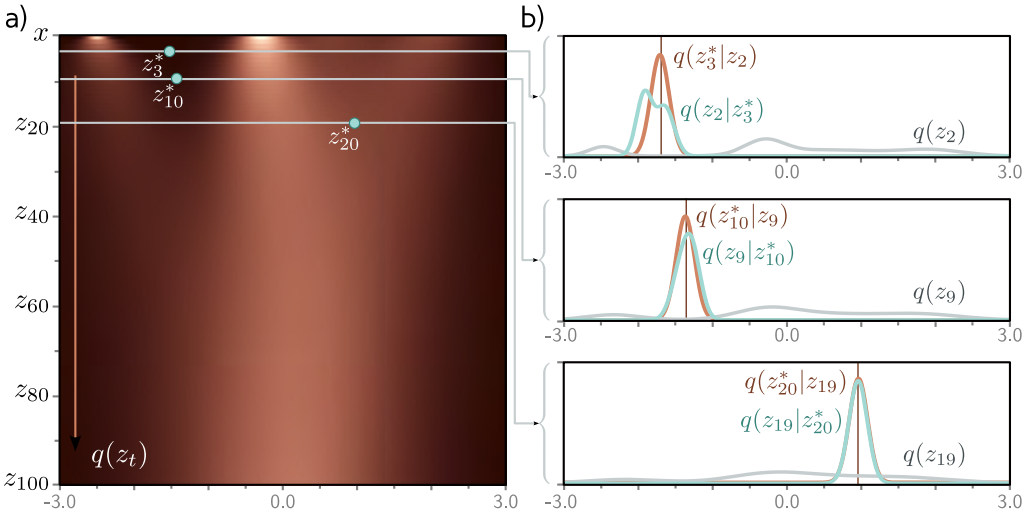

Figure 18.5 Conditional distribution $q(z_{t-1}|z_t)$. a) The marginal densities $q(z_t)$ with three points z_t^* highlighted. b) The probability $q(z_{t-1}|z_t^*)$ (cyan curves) is computed via Bayes' rule and is proportional to $q(z_t^*|z_{t-1})q(z_{t-1})$. In general, it is not normally distributed (top graph), although often the normal is a good approximation (bottom two graphs). The first contributing term $q(z_t^*|z_{t-1})$ is normal (equation 18.2) with a mean that is slightly further from zero than z_t^* (brown curves). The second term is the marginal density $q(z_{t-1})$ (gray curves).

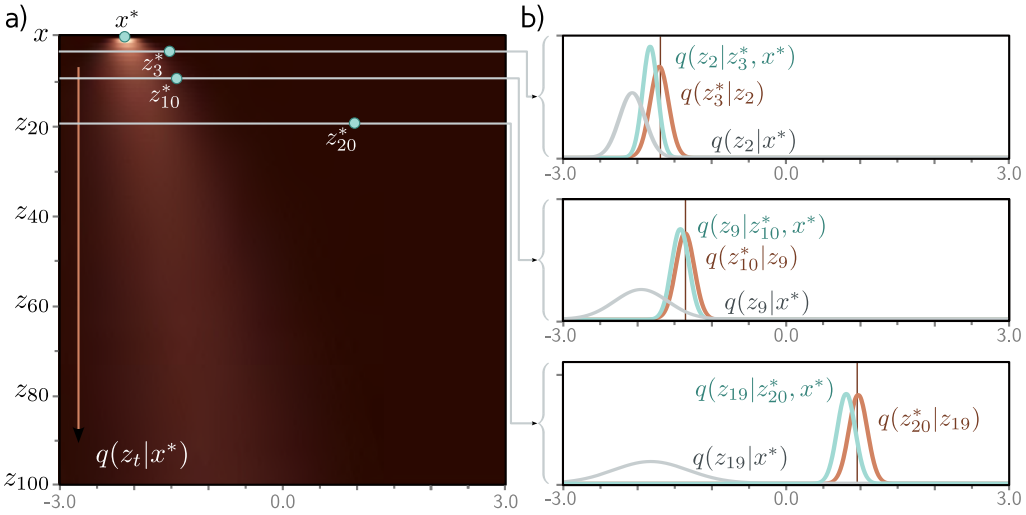

Figure 18.6 Conditional distribution $q(z_{t-1}|z_t, x)$. a) Diffusion kernel for $x^* = -2.1$ with three points z_t^* highlighted. b) The probability $q(z_{t-1}|z_t^*, x^*)$ is computed via Bayes' rule and is proportional to $q(z_t^*|z_{t-1})q(z_{t-1}|x^*)$. This *is* normally distributed and can be computed in closed form. The first term $q(z_t^*|z_{t-1})$ is normal with a mean that is slightly further from zero than z_t^* (brown curves). The second term is the diffusion kernel $q(z_{t-1}|x^*)$ (gray curves).

distribution over $\mathbf{z}_{t-1}$ when we know the current latent variable $\mathbf{z}_t$ and the training data example $\mathbf{x}$ (which, of course, we do when training). To compute an expression for $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ we start with Bayes' rule:

$$\begin{aligned} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \\ &\propto q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1}|\mathbf{x}) \\ &= \text{Norm}_{\mathbf{z}_t}\left[\sqrt{1-\beta_t}\cdot\mathbf{z}_{t-1}, \beta_t\mathbf{I}\right]\text{Norm}_{\mathbf{z}_{t-1}}\left[\sqrt{\alpha_{t-1}}\cdot\mathbf{x}, (1-\alpha_{t-1})\mathbf{I}\right] \\ &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t, \frac{\beta_t}{1-\beta_t}\mathbf{I}\right]\text{Norm}_{\mathbf{z}_{t-1}}\left[\sqrt{\alpha_{t-1}}\cdot\mathbf{x}, (1-\alpha_{t-1})\mathbf{I}\right] \end{aligned} \tag{18.12}$$

where between the first two lines we have used the fact that $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}) = q(\mathbf{z}_t|\mathbf{z}_{t-1})$ because the diffusion process is Markov and all information about $\mathbf{z}_t$ is captured by $\mathbf{z}_{t-1}$. Between lines three and four we use the Gaussian identity

Problem 18.4

$$\text{Norm}_{\mathbf{v}}\left[\mathbf{A}\mathbf{w}, \mathbf{B}\right] \propto \text{Norm}_{\mathbf{w}}\left[\left(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{B}^{-1}\mathbf{v}, \left(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A}\right)^{-1}\right], \tag{18.13}$$

Problem 18.5

to rewrite the first distribution in terms of $\mathbf{z}_{t-1}$. We then use a second Gaussian identity:

$$\begin{aligned} \text{Norm}_{\mathbf{w}}[\mathbf{a}, \mathbf{A}]\cdot\text{Norm}_{\mathbf{w}}[\mathbf{b}, \mathbf{B}] \;\propto\; & \\ \text{Norm}_{\mathbf{w}}\left[\left(\mathbf{A}^{-1}+\mathbf{B}^{-1}\right)^{-1}(\mathbf{A}^{-1}\mathbf{a}+\mathbf{B}^{-1}\mathbf{b}), \left(\mathbf{A}^{-1}+\mathbf{B}^{-1}\right)^{-1}\right], & \end{aligned} \tag{18.14}$$

to combine the two normal distributions in $\mathbf{z}_{t-1}$ which gives:

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}, \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{I}\right]. \tag{18.15}$$

Note that the constants of proportionality in equations 18.12, 18.13, and 18.14 must cancel out, since the final result is already a correctly normalized probability distribution.

18.3 Decoder model (reverse process)

When we learn a diffusion model, we learn the *reverse process*. In other words, we learn a series of probabilistic mappings back from latent variable $\mathbf{z}_T$ to $\mathbf{z}_{T-1}$, from $\mathbf{z}_{T-1}$ to $\mathbf{z}_{T-2}$ and so on until we reach the data $\mathbf{x}$. The true reverse distributions $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ of the diffusion process are complex multi-modal distributions (figure 18.5) that depend on the data distribution $Pr(\mathbf{x})$. We approximate these as normal distributions:

$$\begin{aligned} Pr(\mathbf{z}_T) &= \text{Norm}_{\mathbf{z}_T}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t], \sigma_t^2\mathbf{I}\right] \\ Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1) &= \text{Norm}_{\mathbf{x}}\left[\mathbf{f}_1[\mathbf{z}_1, \boldsymbol{\phi}_1], \sigma_t^1\mathbf{I}\right], \end{aligned} \tag{18.16}$$

where $\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_1]$ is a neural network that computes the mean of the normal distribution in the estimated mapping from $\mathbf{z}_{t-1}$ to the subsequent latent variable $\mathbf{z}_t$. The terms $\{\sigma_t^2\}$ are predetermined. If the hyperparameters β_t in the diffusion process are close to one (and the number of time steps T is large), then this normal approximation will be reasonable.

We generate new examples from $Pr(\mathbf{x})$ using ancestral sampling. We start by drawing $\mathbf{z}_T$ from $Pr(\mathbf{z}_T)$. Then we sample $\mathbf{z}_{T-1}$ from $Pr(\mathbf{z}_{T-1}|\mathbf{z}_T, \boldsymbol{\phi}_T)$, $\mathbf{z}_{T-2}$ from $Pr(\mathbf{z}_{T-2}|\mathbf{z}_{T-1}, \boldsymbol{\phi}_{T-1})$ and so on until we finally generate $\mathbf{x}$ from $Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)$.

18.4 Training

The joint distribution of the observed variable $\mathbf{x}$ and the latent variables $\{\mathbf{z}_t\}$ is:

$$Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T}) = Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1)\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) \cdot Pr(\mathbf{z}_T). \tag{18.17}$$

The likelihood of the parameters $Pr(\mathbf{x}|\boldsymbol{\phi}_{1\ldots T})$ is found by marginalizing over the latent variables:

Appendix B.1.2 Marginalization

$$Pr(\mathbf{x}|\boldsymbol{\phi}_{1\ldots T}) = \int Pr(\mathbf{x}, \mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T}) d\mathbf{z}_{1\ldots T}. \tag{18.18}$$

To train the model, we maximize the log likelihood of the training data $\{\mathbf{x}_i\}$ with respect to the parameters $\boldsymbol{\phi}$:

$$\hat{\boldsymbol{\phi}}_{1\ldots t} = \underset{\boldsymbol{\phi}_{1\ldots T}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\Big[\log\big[Pr(\mathbf{x}_i|\boldsymbol{\phi}_{1\ldots T})\big]\Big]\right]. \tag{18.19}$$

We can't maximize this directly because the marginalization in equation 18.18 is intractable. Hence, we use Jensen's inequality to define a lower bound on the likelihood and optimize the parameters $\{\boldsymbol{\phi}_t\}$ with respect to this bound exactly as we did for the VAE (see section 17.3.1).

18.4.1 Evidence lower bound (ELBO)

To derive the lower bound, we multiply and divide the log likelihood by the encoder distribution $q(\mathbf{z}_{1\ldots T}|\mathbf{x})$ and apply Jensen's inequality for the concave logarithm function:

$$\begin{aligned}\log\left[Pr(\mathbf{x},\boldsymbol{\phi}_{1\ldots T})\right] &= \log\left[\int Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})d\mathbf{z}_{1\ldots T}\right] \\ &= \log\left[\int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}d\mathbf{z}_{1\ldots T}\right] \\ &\geq \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right]d\mathbf{z}_{1\ldots T}. \quad (18.20)\end{aligned}$$

This gives us the evidence lower bound (ELBO):

$$\text{ELBO}[\boldsymbol{\phi}_{1\ldots T}] = \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right]d\mathbf{z}_{1\ldots T}. \quad (18.21)$$

In the VAE, the encoder $q(\boldsymbol{\phi})$ approximates the posterior distribution over the latent variables to make the bound tight, and the decoder maximizes this bound (figure 17.10). In diffusion models, the decoder must do all the work, since the encoder has no parameters. It makes the bound tighter by both (i) changing its parameters so that the static encoder does approximate the posterior $Pr(\mathbf{z}_{1\ldots T}|\mathbf{x},\boldsymbol{\phi}_{1\ldots T})$, and (ii) optimizing its parameters with respect to that bound (see figure 17.6).

18.4.2 Simplifying the ELBO

We now manipulate the log term from the ELBO into the final form that we will optimize. We first substitute in the definitions for the numerator and denominator, from equations 18.17 and 18.3, respectively:

$$\begin{aligned}\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] &= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\prod_{t=2}^{T}Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)\cdot Pr(\mathbf{z}_T)}{q(\mathbf{z}_1|\mathbf{x})\prod_{t=2}^{T}q(\mathbf{z}_t|\mathbf{z}_{t-1})}\right] \quad (18.22)\\ &= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)}{q(\mathbf{z}_1|\mathbf{x})}\right]+\log\left[\frac{\prod_{t=2}^{T}Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{\prod_{t=2}^{T}q(\mathbf{z}_t|\mathbf{z}_{t-1})}\right]+\log\Big[Pr(\mathbf{z}_T)\Big].\end{aligned}$$

Then we expand the denominator of the second term:

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}) = \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})q(\mathbf{z}_t|\mathbf{x})}{q(\mathbf{z}_{t-1}|\mathbf{x})}, \quad (18.23)$$

where the first equality follows because all of the information about variable $\mathbf{z}_t$ is encompassed in $\mathbf{z}_{t-1}$ and so the extra conditioning on the data $\mathbf{x}$ is irrelevant. The second equality is a straightforward application of Bayes' rule.

Appendix B.1.4 Bayes' rule

Substituting this result back in, we have:

$$
\begin{aligned}
&\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] \\
&\quad= \log\left[\frac{Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)}{q(\mathbf{z}_1|\mathbf{x})}\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\cdot\frac{q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})}\right] + \log\Big[Pr(\mathbf{z}_t)\Big] \\
&\quad= \log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right] + \log\left[\frac{\prod_{t=2}^{T} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{\prod_{t=2}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\right] + \log\left[\frac{Pr(\mathbf{z}_t)}{q(\mathbf{z}_T|\mathbf{x})}\right] \\
&\quad\approx \log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right] + \sum_{t=2}^{T}\log\left[\frac{Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\right], \qquad (18.24)
\end{aligned}
$$

where all but two of the terms in the sum of the ratio $q(\mathbf{z}_{t-1}|\mathbf{x})/q(\mathbf{z}_t|\mathbf{x})$ cancel out between lines two and three leaving only $q(\mathbf{z}_1|\mathbf{x})$ and $q(\mathbf{z}_T|\mathbf{x})$. The last term in the third line is approximately $\log[1] = 0$, since the result of the forward process $q(\mathbf{z}_T|\mathbf{x})$ should be a standard normal distribution and so is the prior $Pr(\mathbf{z}_T)$.

The simplified ELBO is hence:

$$
\begin{aligned}
\text{ELBO}[\boldsymbol{\phi}_{1\ldots T}] &= \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}_{1\ldots T}|\boldsymbol{\phi}_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right] d\mathbf{z}_{1\ldots T} \qquad (18.25)\\
&\approx \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\left(\log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right] + \sum_{t=2}^{T}\log\left[\frac{Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)}{q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})}\right]\right) d\mathbf{z}_{1\ldots T}\\
&= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}\Big[\log\left[Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1)\right]\Big] - \sum_{t=2}^{T}\mathrm{D}_{KL}\Big[Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)||q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})\Big].
\end{aligned}
$$

where we have substituted in the definition of $q(\mathbf{z}_{1\ldots T}|\mathbf{x})$ from equation 18.3 between lines two and three and integrated out the irrelevant terms. The second term results from the definition of KL-divergence.

Appendix B.5.1 KL divergence

18.4.3 Analyzing the ELBO

The first probability term in the ELBO was defined in equation 18.16:

$$
Pr(\mathbf{x}|\mathbf{z}_1,\boldsymbol{\phi}_1) = \mathrm{Norm}_{\mathbf{x}}\left[\mathbf{f}_1[\mathbf{z}_1,\boldsymbol{\phi}_1],\sigma_t^2\mathbf{I}\right], \qquad (18.26)
$$

and is equivalent to the reconstruction term in the VAE. The ELBO will be larger if the model prediction matches the observed data. As for the VAE, we will approximate the expectation over the log of this quantity using a Monte-Carlo estimate (see equations 17.22–17.23), in which we estimate the expectation with a sample from $q(\mathbf{z}_1|\mathbf{x})$.

The KL-divergence terms in the ELBO measure the distance between $Pr(\mathbf{z}_{t-1}|\mathbf{z}_t,\boldsymbol{\phi}_t)$ and $q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})$ which were defined in equations 18.16 and 18.15, respectively:

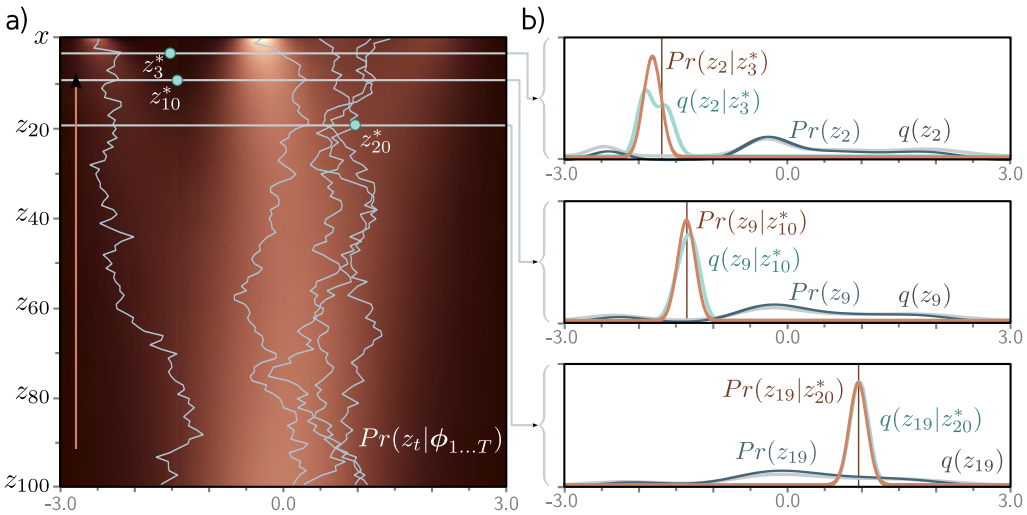

Figure 18.7 Fitted Model. a) Individual samples can be generated by sampling from the standard normal distribution $Pr(z_T)$ and then sampling z_{T-1} from $Pr(z_{T-1}|z_T) = \text{Norm}_{\mathbf{z}_{T-1}}[\mathbf{f}_T[z_T, \boldsymbol{\phi}_T], \sigma_T^2\mathbf{I}]$ and so on until we reach x (five paths shown). The estimated marginal densities (heatmap) are the aggregation of these samples, and are similar to the true marginal densities (figure 18.4). b) The estimated distribution $Pr(z_{t-1}|z_t)$ (brown curve) is a reasonable approximation to the true posterior of the diffusion model $q(z_{t-1}|z_t)$ (cyan curve) from figure 18.5. The marginal distributions $Pr(z_t)$ and $q(z_t)$ of the estimated and true models (dark blue and gray curves, respectively) are also similar.

$$\begin{aligned} Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t], \sigma_t^2\mathbf{I}\right] \\ q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \text{Norm}_{\mathbf{z}_{t-1}}\left[\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}, \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}\mathbf{I}\right]. \end{aligned} \tag{18.27}$$

Appendix B.5.4 KL divergence between normal distributions

Problem 18.6

The KL-divergence between two normal distributions has a closed form expression. Moreover, many of the terms in this expression do not depend on $\boldsymbol{\phi}$ (see problem 18.6), and the expression simplifies to the squared difference between the means:

$$\begin{aligned} D_{KL}\Big[Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t)||q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})\Big] &= \\ \frac{1}{2\sigma_t^2}\left\|\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} - \mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]\right\|^2 + C, \end{aligned} \tag{18.28}$$

where C is a constant.

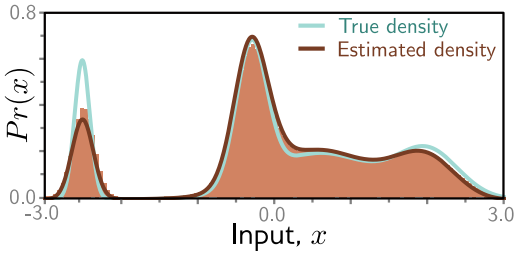

Figure 18.8 Fitted model results. Cyan and brown curves are original and estimated densities and correspond to the top rows of figures 18.5 and 18.7, respectively. Vertical bars are binned samples from the model, generated by propagating back samples $Pr(\mathbf{z}_T)$ as shown for the five paths in figure 18.7.

18.4.4 Diffusion loss function

To fit the model, we maximize the ELBO with respect to the parameters $\boldsymbol{\phi}_{1\ldots T}$. We recast this as a minimization by multiplying with minus one to give the loss function:

$$\begin{aligned} L[\boldsymbol{\phi}] &= \sum_{i=1}^{I}\Bigg(\overbrace{-\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big]}^{\text{reconstruction term}} \\ &\quad + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\Bigg\| \underbrace{\left(\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_{it} + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_i\right)}_{\text{mean of } q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})} - \underbrace{\mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]}_{\text{predicted } \mathbf{z}_{t-1}} \Bigg\|^2\Bigg), \end{aligned} \tag{18.29}$$

where $\mathbf{x}_i$ is the i^{th} data point and $\mathbf{z}_{it}$ is the associated latent variable at diffusion step t.

18.4.5 Training procedure

This loss function can be used to train a network for each diffusion time step. It minimizes the difference between the estimate $\mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$ of the hidden variable at the previous time step and the most likely value that it took given the ground truth de-noised data $\mathbf{x}$.

Figures 18.7 and 18.8 show the fitted reverse process for the simple 1D example. This model was trained by (i) taking a large dataset of examples $\mathbf{x}$ from the original density, (ii) using the diffusion kernel to predict many corresponding values for the latent variable at time $\mathbf{z}_t$, and then (iii) training a series of models $\mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$ to minimize the loss function in equation 18.29. These models were nonparametric (i.e., lookup tables relating 1D input to 1D output), but more typically they would be deep neural networks.

18.5 Reparameterization of loss function

Although the loss function in equation 18.29 can be used, diffusion models have been found to work better with a different parameterization; the loss function is modified so

that the model aims to predict the noise that was mixed with the original data example to create the current variable. Section 18.5.1 discusses reparameterizing the target (first two terms in second line of equation 18.29) and section 18.5.2 discusses reparameterizing the network (last term in second line of equation 18.29).

18.5.1 Reparameterization of target

The original diffusion update was given by:

$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \tag{18.30}$$

and so the data term $\mathbf{x}$ in equation 18.28, can be expressed as the diffused image minus the noise that was added to it:

$$\mathbf{x} = \frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}. \tag{18.31}$$

Substituting this into the target terms from equation 18.29 gives:

$$\begin{aligned}\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} &= \\ \frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\left(\frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}\right) \\ \frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\beta_t}{1-\alpha_t}\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\right),\end{aligned} \tag{18.32}$$

Problem 18.7

where we have used the fact that $\sqrt{\alpha_t}/\sqrt{\alpha_{t-1}} = \sqrt{1-\beta_t}$ between the second and third lines. Simplifying further, we get:

$$\begin{aligned}\frac{(1-\alpha_{t-1})}{1-\alpha_t}\sqrt{1-\beta_t}\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} &= \\ \left(\frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}}{1-\alpha_t} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\right)\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon} \\ \left(\frac{(1-\alpha_{t-1})(1-\beta_t)}{(1-\alpha_t)\sqrt{1-\beta_t}} + \frac{\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\right)\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon} \\ \frac{(1-\alpha_{t-1})(1-\beta_t)+\beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon} \\ \frac{1-\alpha_t}{(1-\alpha_t)\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon} \\ \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon},\end{aligned} \tag{18.33}$$

where we have multiplied the numerator and denominator of the first term by $\sqrt{1-\beta_t}$ between lines two and three, and multiplied out the terms and simplified the numerator in the first term between lines three and four.

Problem 18.8

Substituting this back into the loss function we have:

$$\begin{aligned} L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} -\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] \\ &\quad + \sum_{t=2}^{T}\frac{1}{2\sigma_t^2}\left\|\left(\frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_{it} - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}\right) - \mathbf{f}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t]\right\|^2. \end{aligned} \tag{18.34}$$

18.5.2 Reparameterization of network

Now we replace the model $\hat{\mathbf{z}}_{t-1} = \mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$, with a new model $\hat{\boldsymbol{\epsilon}} = \mathbf{g}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$, which predicts the noise $\boldsymbol{\epsilon}$ that was mixed with $\mathbf{x}$ to create $\mathbf{z}_t$:

$$\mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t] = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]. \tag{18.35}$$

Substituting the new model into equation 18.34 produces the criterion:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} -\log\Big[\text{Norm}_{\mathbf{x}_i}\left[\mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1],\sigma_1^2\mathbf{I}\right]\Big] + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2. \tag{18.36}$$

The log normal can equivalently be written as a least squares loss plus a constant C_i:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I}\frac{1}{2\sigma_1^2}\left\|\mathbf{x}_i - \mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1]\right\|^2 + \sum_{t=2}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2 + C_i.$$

Substituting in the definitions of $\mathbf{x}$ and $\mathbf{f}_t[\mathbf{z}_t,\boldsymbol{\phi}_t]$ from equations 18.31 and 18.35, respectively, the first term simplifies to:

Problem 18.9

$$\frac{1}{2\sigma_1^2}\left\|\mathbf{x}_i - \mathbf{f}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1]\right\|^2 = \frac{1}{2\sigma_1^2}\left\|\frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_{i1},\boldsymbol{\phi}_1] - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\boldsymbol{\epsilon}_{i1}\right\|^2. \tag{18.37}$$

Adding this back to the final loss function yields:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I}\sum_{t=1}^{T}\frac{\beta_t^2}{(1-\alpha_t)(1-\beta_t)2\sigma_t^2}\left\|\mathbf{g}_t[\mathbf{z}_{it},\boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it}\right\|^2, \tag{18.38}$$

where we have disregarded the additive constants C_i.

Algorithm 18.1: Diffusion model training

Input: Training data $\mathbf{x}$
Output: Model parameters $\boldsymbol{\phi}_t$
repeat
  for $i \in \mathcal{B}$ **do** // For every training example index in batch
    $t \sim \text{Uniform}[1, \ldots T]$ // Sample random timestep
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ // Sample noise
    $\ell_i = \left\| \mathbf{g}_t\left[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\phi}_t\right] - \boldsymbol{\epsilon} \right\|^2$ // Compute individual loss
  Accumulate losses for batch and take gradient step
until converged

where the log normal has been replaced by a least squares term plus an additive constant C (see section 5.3.1). In practice, the scaling factors (which might be different at each time step) are ignored giving an even simpler formulation:

$$\begin{aligned} L[\boldsymbol{\phi}] &= \sum_{i=1}^{I}\sum_{t=1}^{T} \left\| \mathbf{g}_t[\mathbf{z}_{it}, \boldsymbol{\phi}_t] - \boldsymbol{\epsilon}_{it} \right\|^2 \\ &= \sum_{i=1}^{I}\sum_{t=1}^{T} \left\| \mathbf{g}_t\left[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{it}, \boldsymbol{\phi}_t\right] - \boldsymbol{\epsilon}_{it} \right\|^2, \end{aligned} \tag{18.39}$$

where we have rewritten $\mathbf{z}_t$ using the diffusion kernel (equation 18.30) in the second line.

18.6 Implementation

This leads to straightforward algorithms for both training the model (algorithm 18.1) and sampling (algorithm 18.2). The training algorithm has the advantages that it is (i) simple to implement and (ii) naturally augments the dataset; we can reuse every original data point $\mathbf{x}_i$ as many times as we want at each time step with different noise instantiations $\boldsymbol{\epsilon}$. The sampling algorithm has the disadvantage that it requires serial processing of many neural networks $\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$, and is hence time consuming.

18.6.1 Application to images

The early successes of diffusion models were in modeling image data. Here, we need to construct models that can take a noisy image and predict the noise that was added to it at each stage. The obvious architectural choice for this image-to-image mapping is an encoder-decoder model similar to the U-Net.

Algorithm 18.2: Sampling

Input: Model, $\mathbf{g}_t[\bullet, \boldsymbol{\phi}_t]$
Output: Sample, $\mathbf{x}$
$\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ // Sample last latent variable
for $t = T \ldots 2$ **do**
  $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ // Predict previous latent variable
  $\boldsymbol{\epsilon} \sim \text{Norm}_{\boldsymbol{\epsilon}}[\mathbf{0}, \mathbf{I}]$ // Draw new noise vector
  $\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t\boldsymbol{\epsilon}$ // Add noise to previous latent variable
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_1, \boldsymbol{\phi}_1]$ // Generate sample from $\mathbf{z}_1$ without noise

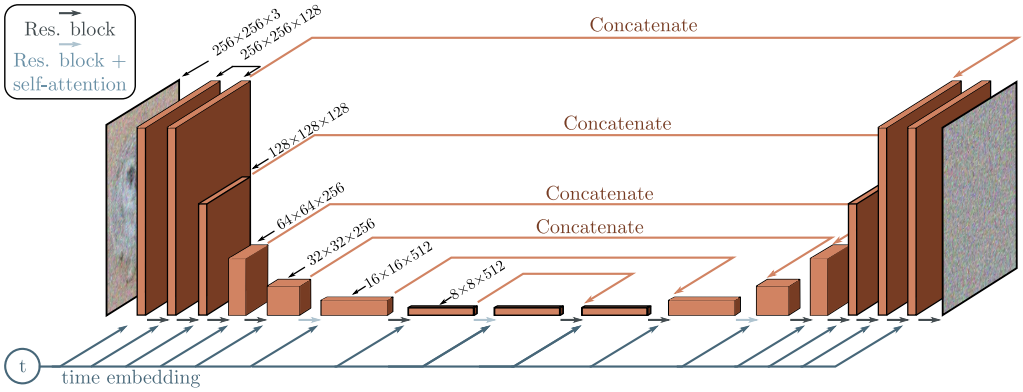

Figure 18.9 U-Net as used in diffusion models for images. The network aims to predict the noise that was added to the image. It consists of an encoder which reduces the scale and increases the number of channels and a decoder which increases the scale and reduces the number of channels. The encoder representations are concatenated to their partner in the decoder. Connections between adjacent representations consist of residual blocks, and periodic global self-attention in which every spatial position interacts with every other spatial position. A single network is used for all time steps, by passing a sinusoidal time embedding (figure 12.5) through a shallow neural network and adding the result to the channels at every spatial position at every stage of the U-Net.

However, there may be a very large number of diffusion steps, and training and storing this many U-Nets is inefficient. The solution is to train a single U-Net that also takes a predetermined vector representing the time step as input (figure 18.9). In practice, this is resized to match the number of channels at each stage of the U-Net and used to offset and/or scale the representation at each spatial position.

The logic for having so many time steps is that the conditional probabilities $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ become closer to normally distributed when the hyperparameters β_t are close to zero and so the variational approximation is closer. However, in practice, this makes sampling very

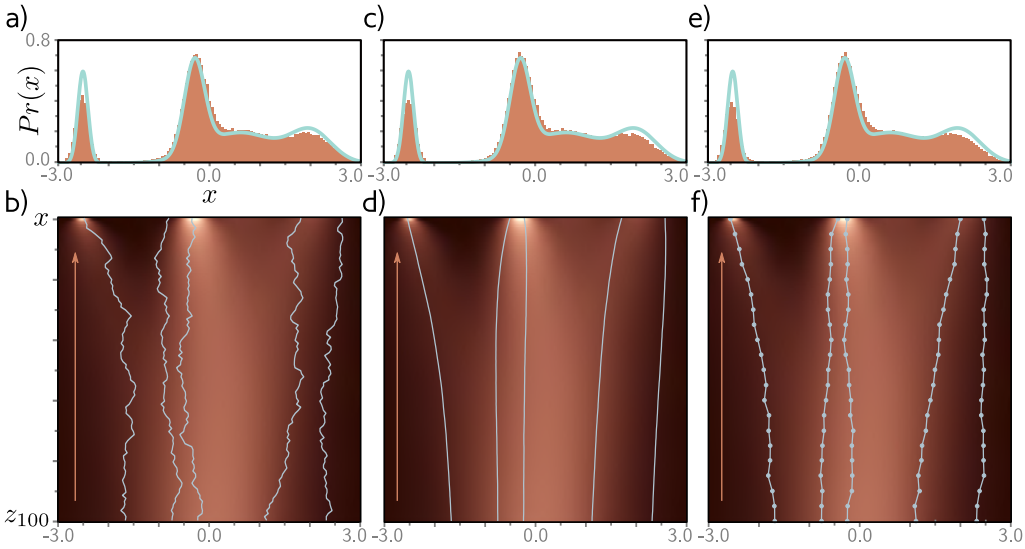

Figure 18.10 Different diffusion processes that are all compatible with the same model. a) Histogram of samples generated from re-parameterized model plotted alongside ground truth density curve. b) Five sampled trajectories superimposed on ground truth marginal distributions (figure 18.5) The same trained model is compatible with a family of diffusion models (and corresponding updates in the opposite direction), including the denoising diffusion implicit (DDIM) model, which is deterministic and does not add noise at each step. c) Histogram of samples from DDIM model. d) Five trajectories from DDIM model. The same model is also compatible with accelerated diffusion models that skip inference steps for increased sampling speed. e) Histogram of samples from accelerated model. f) Five trajectories from accelerated model.

slow. We might have to run the U-Net model through $T = 1000$ steps to generate good images.

18.6.2 Improving generation speed

The loss function (equation 18.39) requires the diffusion kernel to have the form $q(\mathbf{z}_t|\mathbf{x}) = \text{Norm}[\sqrt{\alpha_t}\mathbf{x}, \sqrt{1-\alpha_t}\cdot\mathbf{I}]$. The same loss function will be valid for *any* forward process with this relation and there are a family of such compatible processes. These are all optimized by the same loss function, but have different rules for the forward process, and different corresponding rules for how to use the estimated noise $\mathbf{g}[\mathbf{z}_t, \boldsymbol{\phi}_t]$ to estimate $\mathbf{z}_{t-1}$ from $\mathbf{z}_t$ in the reverse process (figure 18.10).

Among this family are *denoising diffusion implicit models*, which are no longer stochastic after the first step from $\mathbf{x}$ to $\mathbf{z}_1$ and *accelerated sampling* models where the

forward process is defined only on a sub-sequence of time steps. This allows a reverse process that skips time steps and hence makes sampling much more efficient; good samples can be created with 50 time steps when the forward process is no longer stochastic. This is much faster than before but still slower than most other generative models.

18.6.3 Conditional generation

If the data has associated labels c, then these can be exploited to control the generation. Sometimes this can improve generation results in GANs and we might expect this to be the case in diffusion models as well; it's easier to denoise an image if you have some information about what that image contains. One approach to conditional synthesis in GANs is *classifier guidance*. This modifies the denoising update from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$ to take into account additional information c. In practice, this means adding an extra term into the final update step in algorithm 18.2 to:

$$\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t^2 \frac{\partial \log\big[\Pr(c|\mathbf{z}_t)\big]}{\partial \mathbf{z}_t} + \sigma_t \boldsymbol{\epsilon}. \tag{18.40}$$

The new term depends on the gradient of a classifier $Pr(c|\mathbf{z}_t)$ that is based on the latent variable $\mathbf{z}_t$. This maps features from the downsampling half of the U-Net to the class c. Like the U-Net , it is usually shared across all time-steps and takes time as an input. The update from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$ now makes the class c more likely.

Classifier-free guidance avoids learning a separate classifier $Pr(c|\mathbf{z}_t)$, but instead incorporates class information into the main model $\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t, c]$. In practice, this usually takes the form of adding an embedding based on c to the layers of the U-Net in a similar way to how the time step is added (see figure 18.9). This model is jointly trained on conditional and unconditional objectives by randomly dropping the class information during training. Hence, it can both generate unconditional or conditional data examples at test time or any weighted combination of the two. This brings a surprising advantage; if the conditioning information is over-weighted, then the model tends to produce very high quality, but slightly stereotypical examples. This is somewhat analogous to the use of truncation in GANs (figure 15.10).

18.6.4 Improving generation quality

As for other generative models, the highest quality results result from applying a combination of tricks and extensions to the basic model. First, it's been noted that it also helps to estimate the variances σ_t^2 of the reverse process as well as the mean (i.e., the widths of the brown normal distributions in figure 18.7). This particularly improves the results when sampling with fewer steps. Second, it's possible to modify the noise schedule in the forward process so that β_t varies at each step, and this can improve results.

Third, to generate high resolution images, a cascade of diffusion models is used. The first creates a low resolution image (possibly guided by class information). The subsequent diffusion models generate progressively higher resolution images. They condition

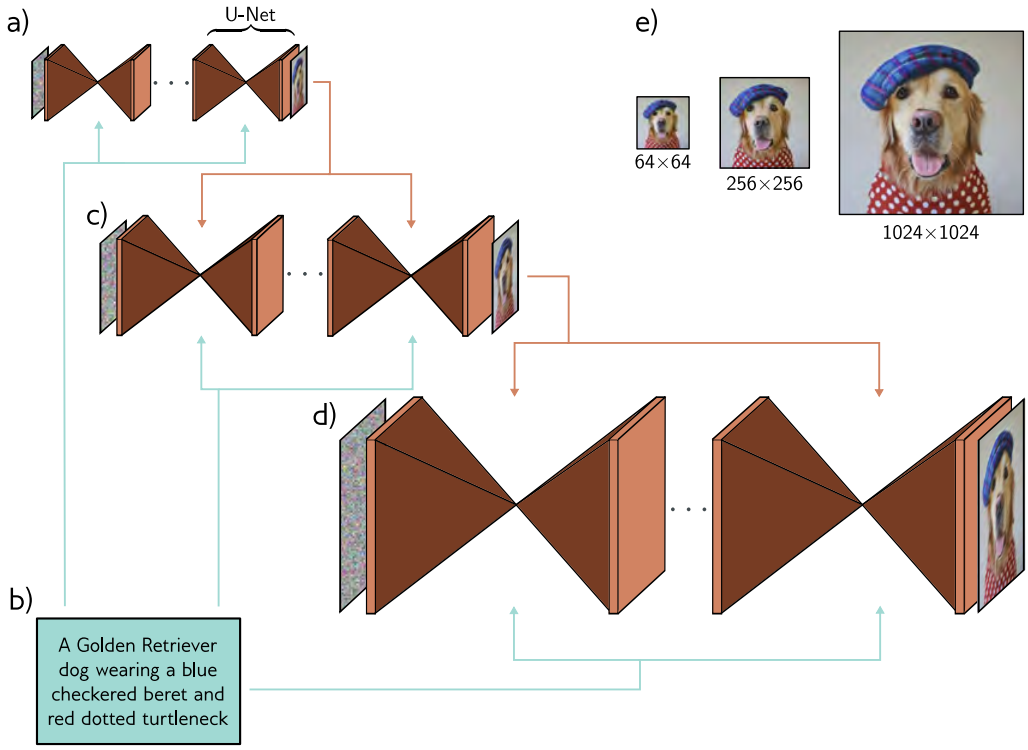

Figure 18.11 Cascaded conditional generation based on a text prompt. a) A diffusion model consisting of a series of U-Nets is used to generate a 64×64 image. b) This generation is conditioned on a sentence embedding computed by a language model. c) A higher resolution 256×256 image is generated and conditioned on the smaller image *and* the text encoding. d) This is repeated to create a 1024×1024 image. e) Final image sequence. Adapted from Saharia et al. (2022b).

on the lower resolution image by resizing this and appending it to the layers of the constituent U-Net, as well as any other class information (figure 18.11).

Combining all of these techniques allows generation of very high quality images. Figure 18.12 shows examples of images generated from a model conditioned on the ImageNet class. It is particularly impressive that the same model can learn to generate such diverse classes. Figure 18.13 shows images generated from a model that is trained to condition on text captions encoded by a language model like BERT, which are inserted into the model in the same way as the time step and lower resolution images (figures 18.9 and 18.11). This results in very realistic images that agree with the caption. Since the diffusion model is by its nature stochastic, it's possible to generate multiple images that are conditioned on the same caption.

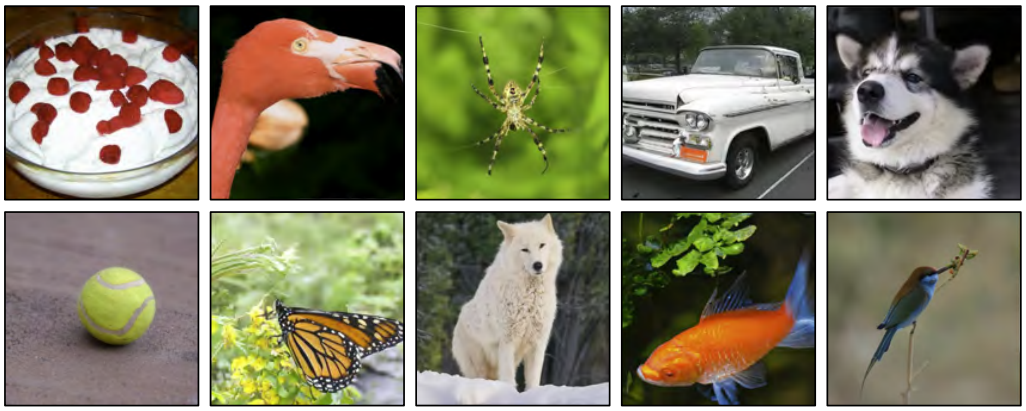

Figure 18.12 Conditional generation using classifier guidance. Image samples conditioned on different ImageNet classes. The same model produces high quality samples of highly varied image classes. Adapted from Dhariwal & Nichol (2021).

Figure 18.13 Conditional generation using text prompts. Synthesized images from a cascaded generation framework, conditioned on a text prompt encoded by a large language model. The stochastic model can produce many different images compatible with the prompt. The model can count objects, and incorporate text into images. Adapted from Saharia et al. (2022b).

18.7 Summary

Diffusion models map the data examples through a series of latent variables by repeatedly blending the current representation with random noise. After sufficient steps, the representation becomes indistinguishable from white noise. Since these steps are small, the reverse denoising process at each step can be approximated with a normal distribution, and predicted by a deep learning model. The loss function is based on the evidence lower bound (ELBO), and ultimately results in a simple least-squares formulation.

For image generation, each denoising step is implemented using a U-Net, so sampling is slow compared to other generative models. To improve generation speed, it's possible to change the diffusion model to a deterministic formulation and here sampling with fewer steps works well. Several methods have been proposed to condition generation on class information, images, and text information. Combining these methods produces very impressive text-to-image synthesis.

Notes

Denoising diffusion models were introduced by Sohl-Dickstein et al. (2015), and early related work based on score-matching was carried out by Song & Ermon (2019). Ho et al. (2020) produced image samples that were competitive with GANs and kick-started a wave of interest in this area. Most of the exposition in this chapter including the original formulation and the reparameterization is derived from this paper. Dhariwal & Nichol (2021) improved on the quality of these results and showed for the first time that images from diffusion models were quantitatively superior to GAN models in terms of Fréchet Inception Distance. At the time of writing, the state-of-the-art results for conditional image synthesis have been achieved by Karras et al. (2022). Surveys of denoising diffusion models can be found in Croitoru et al. (2022), Cao et al. (2022), Luo (2022), and Yang et al. (2022).

Applications for images: Applications of diffusion models include text-to-image generation (Nichol et al., 2022; Ramesh et al., 2022; Saharia et al., 2022b), image-to-image tasks such as colorization, inpainting, uncropping and restoration (Saharia et al., 2022a), super-resolution (Saharia et al., 2022c), image editing (Hertz et al., 2022; Meng et al., 2021), removing adversarial perturbations (Nie et al., 2022), semantic segmentation (Baranchuk et al., 2022), and medical imaging (Song et al., 2021b; Chung & Ye, 2022; Chung et al., 2022; Peng et al., 2022; Xie & Li, 2022; Luo et al., 2022) where the diffusion model is sometimes used as a prior.

Different data types: Diffusion models have also been applied to video data (Ho et al., 2022b; Harvey et al., 2022; Yang et al., 2022; Höppe et al., 2022; Voleti et al., 2022) for generation, past and future frame prediction, and interpolation. They have been used for 3D shape generation (Zhou et al., 2021; Luo & Hu, 2021), and recently a technique has been introduced to generate 3D models using only a 2D text-to-image diffusion model (Poole et al., 2023). Austin et al. (2021) and Hoogeboom et al. (2021) investigated diffusion models for discrete data. Kong et al. (2021) and Chen et al. (2021d) applied diffusion models to audio data.

Alternatives to denoising: The diffusion models in this chapter mix noise with the data and build a model to gradually denoise the result. However, degrading the image using noise is not necessary. Rissanen et al. (2022) devised a method that progressively blurred the image and

Bansal et al. (2022) show that the same ideas work with a large family of degradations which do not have to be stochastic. These include masking, morphing, blurring, and pixelating.

Comparison to other generative models: Diffusion models synthesize higher quality images than other generative models and are simple to train. They can be thought of as a special case of a hierarchical VAE (Vahdat & Kautz, 2020; Sønderby et al., 2016b) where the encoder is fixed and the latent space is the same size as the data. They are probabilistic, but like the VAE can only compute a lower bound on the likelihood of a data point. However, Kingma et al. (2021) show that this lower bound improves on the exact log likelihoods for test data from normalizing flows and autoregressive models. The main disadvantages of diffusion models is that they are slow and that the latent space has no semantic interpretation.

Improving quality: Many techniques have been proposed to improve image quality. These include the reparameterization of the network described in section 18.5 and the equal-weighting of the subsequent terms (Ho et al., 2020). Choi et al. (2022) subsequently investigated different weightings of terms in the loss function.

Kingma et al. (2021) improved the test log-likelihood of the model by learning the denoising weights β_t. Conversely, Nichol & Dhariwal (2021) improved performance by learning separate variances σ^2 of the denoising estimate at each time step in addition to the mean. Bao et al. (2022) show how to learn the variances *after* training the model.

Ho et al. (2022a) developed the cascaded method for producing very high resolution images (figure 18.11). To prevent artifacts in lower resolution images being propagated the higher resolutions, they introduced *noise conditioning augmentation*; here, the lower resolution image is degraded by adding noise to the conditioning image at each training step. This reduces the reliance on the exact details of the lower resolution image during training. It is done during inference, and here where the best noise level is chosen by sweeping over different values.

Improving speed: One of the major drawbacks of diffusion models is that they take a long time to train and sample from. Stable diffusion (Rombach et al., 2022) projects the original data to a smaller latent space using a conventional VAE and then runs the diffusion process in this smaller space. This has the advantages of reducing the dimensionality of the training data for the diffusion process, and allowing other data types (text, graphs, etc.) to be described by diffusion models. Vahdat et al. (2021) applied a similar approach.

Song et al. (2021a) showed that an entire family of diffusion processes are compatible with the training objective. Most of these processs are non-Markovian (i.e., the diffusion step does not only depend on the results of the previous step). One of these models is the denoising diffusion implicit model (DDIM) in which the updates are not stochastic (figure 18.10b). This model is amenable to taking larger steps (figure 18.10b) without inducing large errors. It effectively converts the model into an ordinary differential equation (ODE) in which the trajectories have low curvature, and allows efficient numerical methods for solving ODEs to be applied.

Song et al. (2021c) propose converting the underlying stochastic differential equations into a *probability flow* ODE which has the same marginal distributions as the original process. Vahdat et al. (2021), Xiao et al. (2022b), and Karras et al. (2022) all exploit techniques for solving ODEs to speed up synthesis. Karras et al. (2022) identified the best-performing time discretization for sampling, and evaluated different sampler schedules. The results of these and other improvements has been a significant drop in steps required during synthesis.

Sampling is slow because many small diffusion steps are required to ensure that the posterior distribution $Pr(\mathbf{z}_{t-1}|\mathbf{z}_t)$ is close to Gaussian (figure 18.5), and so the variational Gaussian distribution is appropriate. If we use a model that describes a more complex distribution at each denoising step, then we can use fewer diffusion steps in the first place. To this end, Xiao et al. (2022b) have investigated using conditional GAN models and Gao et al. (2021) investigated

using conditional energy based models. Although these models cannot describe the original data distribution, they suffice to predict the (much simpler) reverse diffusion step.

Salimans & Ho (2022) distilled adjacent steps of the denoising process into a single step to speed up synthesis. Dockhorn et al. (2022) introduced momentum into the diffusion process. This makes the trajectories smoother and so more amenable to coarse sampling.

Conditional generation: Dhariwal & Nichol (2021) introduced classifier guidance, in which a classifier learned to identify the category of object being synthesized at each step, and this is used to bias the denoising update towards that class. This works well, but training a separate classifier is expensive. *Classifier free guidance* (Ho & Salimans, 2022) concurrently trains conditional and unconditional denoising models by dropping the class information some proportion of the time in a process akin to dropout. This technique allows control of the relative contributions of the conditional and unconditional components. By over-weighting the conditional component, the model produces more typical and realistic samples.

The standard technique for conditioning on images is to append the (resized) image to the different layers of the U-Net. For example, this was used in the cascaded generation process for super-resolution (Ho et al., 2022a). Choi et al. (2021) provide a method for conditioning on images in an unconditional diffusion model by matching the latent variables with those of a conditioning image. The standard technique for conditioning on text is to linearly transform the text to the same size as the U-Net layer and then add it to the representation in the same way that the time embedding is introduced (figure 18.9).

Text-to-image: Before diffusion models, state-of-the-art text-to-image systems were based on transformers (Ramesh et al., 2021). GLIDE (Nichol et al., 2022) and Dall·E 2 (Ramesh et al., 2022) both conditioned on embeddings from the CLIP model (Radford et al., 2021) which generates joint embeddings for text and image data. Imagen (Saharia et al., 2022b) showed that text embeddings from a large language model could produce even better results (see figure 18.13). The same authors introduced a benchmark (DrawBench) which is designed to evaluate the ability of a model to render colors, numbers of objects, spatial relations and other characteristics. Feng et al. (2022) have developed a Chinese text-to-image model.

Connections to other models: This chapter described diffusion models as hierarchical variational autoencoders because this approach connects most closely with the other parts of this book. However, diffusion models also have close connections with stochastic differential equations (consider the paths in figure 18.5) and with score matching (Song & Ermon, 2019, 2020). Song et al. (2021c) presented a framework based on stochastic differential equations that encompasses both the denoising and score matching interpretations. Diffusion models also have close connections to normalizing flows (Zhang & Chen, 2021). Yang et al. (2022) present an overview of the relation between diffusion models and other generative approaches.

Problems

Problem 18.1 Show that if $\text{Var}[\mathbf{x}_{t-1}] = \mathbf{I}$ and we use the update:

$$\mathbf{x}_t = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t, \tag{18.41}$$

then $\text{Var}[\mathbf{x}_t] = \mathbf{I}$ and so the variance stays the same.

Problem 18.2 Consider the variable:

$$z = a \cdot \epsilon_1 + b \cdot \epsilon_2, \tag{18.42}$$

where both ϵ_1 and ϵ_2 are drawn from independent standard normal distributions with mean zero and unit variance. Show that:

$$\begin{aligned} \mathbb{E}[z] &= 0 \\ \text{Var}[z] &= a^2 + b^2, \end{aligned} \tag{18.43}$$

and so we could equivalently compute $z = \sqrt{a^2 + b^2} \cdot \epsilon$ where ϵ is also drawn from a standard normal distribution.

Problem 18.3 Continue the process in equation 18.5 to show that:

$$\mathbf{x}_3 = \prod_{s=1}^{3} \sqrt{1-\beta_s} \cdot \mathbf{x} + \sqrt{1 - \prod_{s=1}^{3}(1-\beta_s)} \cdot \boldsymbol{\epsilon}. \tag{18.44}$$

Problem 18.4 Prove the relation:

$$\text{Norm}_{\mathbf{v}}\left[\mathbf{A}\mathbf{w}, \mathbf{B}\right] \propto \text{Norm}_{\mathbf{w}}\Big[(\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}^{-1}\mathbf{v}, (\mathbf{A}^T\mathbf{B}^{-1}\mathbf{A})^{-1}\Big]. \tag{18.45}$$

Problem 18.5 Prove the relation:

$$\begin{aligned} \text{Norm}_{\mathbf{x}}[\mathbf{a}, \mathbf{A}]\text{Norm}_{\mathbf{x}}[\mathbf{b}, \mathbf{B}] \quad &\propto \\ &\text{Norm}_{\mathbf{x}}\Big[(\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b})(\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}\Big]. \end{aligned} \tag{18.46}$$

Problem 18.6
The KL-divergence between two normal distributions is given by:

$$\begin{aligned} D_{KL}\Big[\text{Norm}_{\mathbf{w}}[\mathbf{a}, \mathbf{A}]\Big|\Big|\text{Norm}_{\mathbf{w}}[\mathbf{b}, \mathbf{B}]\Big] \quad &= \\ &\frac{1}{2}\left(\text{tr}\left[\mathbf{B}^{-1}\mathbf{A}\right] - d + (\mathbf{a}-\mathbf{b})^T\mathbf{B}^{-1}(\mathbf{a}-\mathbf{b}) + \log\left[\frac{|\mathbf{B}|}{|\mathbf{A}|}\right]\right). \end{aligned} \tag{18.47}$$

Substitute the definitions from equation 18.27 into this expression and show that the only term that depends on the parameters $\boldsymbol{\phi}$ is the first term from equation 18.28.

Problem 18.7 If $\alpha_t = \prod_{s=1}^{t} 1 - \beta_s$, then show that:

$$\sqrt{\frac{\alpha_{t-1}}{\alpha_t}} = \sqrt{1-\beta_t}. \tag{18.48}$$

Problem 18.8 If $\alpha_t = \prod_{s=1}^{t} 1 - \beta_s$, then show that:

$$\frac{(1-\alpha_{t-1})(1-\beta_t) + \beta_t}{(1-\alpha_t)\sqrt{1-\beta_t}} = \frac{1-\alpha_t}{(1-\alpha_t)\sqrt{1-\beta_t}}. \tag{18.49}$$

Problem 18.9 Prove equation 18.37.