Chapter 7

Inference with Low-Dimensional Distributions

"*Mathematics is the art of giving the same name to different things.*"

— Henri Poincaré

In the previous Chapters of this book, we have studied how to effectively and efficiently learn a representation for a variable $\boldsymbol{x}$ in the world with a distribution $p(\boldsymbol{x})$ that has a low-dimensional support in a high-dimensional space. So far, we have mainly developed the methodology for learning representation and autoencoding in a general, distribution or task-agnostic fashion. With such a learned representation, one can already use it to perform some generic and basic tasks such as classification (if the encoding is supervised with the class) and generation of random samples that have the same distribution as the given data (say natural images or natural languages).

More generally, however, the universality and scalability of the theoretical and computational framework presented in this book has enabled us to learn the distribution of a variety of important real-world high-dimensional data such as natural languages, human poses, natural images, videos, and even 3D scenes. Once the intrinsically rich and low-dimensional structures of these real data can be learned and represented correctly, they start to enable a broad family of powerful, often seemingly miraculous, tasks. Hence, from here onwards, we will start to show how to connect and tailor the general methods presented in previous Chapters to learn useful representations for specific structured data distributions and for many popular tasks in modern practice of machine intelligence.

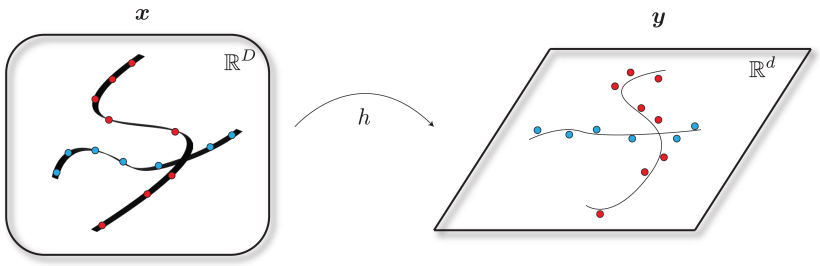


Figure 7.1: **Inference with low-dimensional distributions.** This is the generic picture for this chapter: we have a low-dimensional distribution for $\boldsymbol{x} \in \mathbb{R}^D$ (here depicted as a union of two 2-dimensional manifolds in $\mathbb{R}^3$) and a measurement model $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w} \in \mathbb{R}^d$. We want to infer various things about this model, including the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$, or the conditional expectation $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{y}]$, given various information about the model and (potentially finite) samples of either $\boldsymbol{x}$ or $\boldsymbol{y}$.

7.1 Bayesian Inference and Constrained Optimization

7.1.1 Bayesian Inference with Low-Dimensional Distributions

Leveraging low-dimensionality for stable and robust inference. Generally speaking, a good representation or autoencoding should enable us to utilize the learned low-dimensional distribution of the data $\boldsymbol{x}$ and its representation $\boldsymbol{z}$ for various subsequent classification, estimation, and generation tasks under different conditions. As we have alluded to earlier in Chapter 1 Section 1.2.2, the importance of the *low-dimensionality* of the distribution is the key for us to conduct stable and robust inference related to the data $\boldsymbol{x}$, as illustrated by the few simple examples in Figure 1.11, from incomplete, noisy, and even corrupted observations. As it turns out, the very same concept carries over to real-world high-dimensional data whose distributions have a low-dimensional support, such as natural images and languages.

Despite a dazzling variety of applications in the practice of machine learning with data such as languages, images, videos and many other modalities, almost all practical applications can be viewed as a special case of the following inference problem: given an observation $\boldsymbol{y}$ that depends on $\boldsymbol{x}$, say

$$\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}, \tag{7.1.1}$$

where $h(\cdot)$ represents measurements of a part of $\boldsymbol{x}$ or certain observed attributes and $\boldsymbol{w}$ represents some measurement noise and even (sparse) corruptions, solve the "inverse problem" of obtaining a most likely estimate $\hat{\boldsymbol{x}}(\boldsymbol{y})$ of $\boldsymbol{x}$ or generating a sample $\hat{\boldsymbol{x}}$ that is at least consistent with the observation $\boldsymbol{y} \approx h(\hat{\boldsymbol{x}})$. Figure 7.1 illustrates the general relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$.

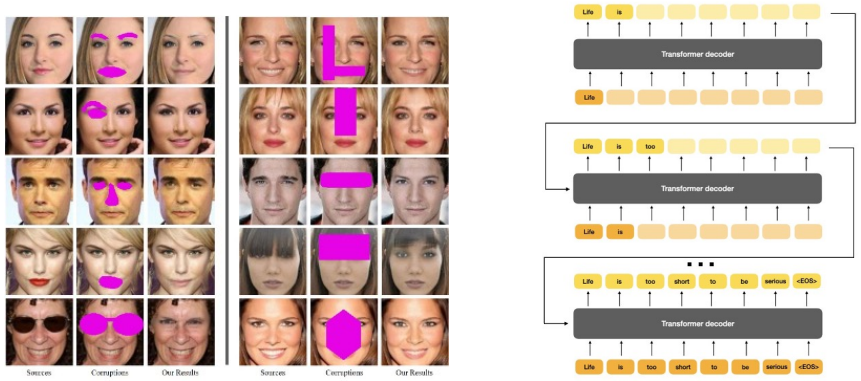


Figure 7.2: Left: image completion. Right: text prediction. In particular, text prediction is the inspiration for the popular Generative Pre-trained Transformer (GPT).

Example 7.1 (Image Completion and Text Prediction). The popular natural image completion and natural language prediction are two typical tasks that require us to recover a full data $\boldsymbol{x}$ from its partial observations $\boldsymbol{y}$, with parts of $\boldsymbol{x}$ masked out and to be completed based on the rest. Figure 7.2 shows some examples of such tasks. In fact, it is precisely these tasks which have inspired how to train modern large models for text generation (such as GPT) and image completion (such as the masked autoencoder) that we will study in greater detail later. ■

Statistical interpretation via Bayes' rule. Generally speaking, to accomplish such tasks well, we need to get ahold of the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$. If we had this, then we would be able to find the maximal likelihood estimate (prediction):

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x} \mid \boldsymbol{y}); \tag{7.1.2}$$

or compute the conditional expectation estimate:

$$\hat{\boldsymbol{x}} = \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{y}] = \int \boldsymbol{x} p(\boldsymbol{x} \mid \boldsymbol{y}) \mathrm{d}\boldsymbol{x}; \tag{7.1.3}$$

or sample from the conditional distribution:

$$\hat{\boldsymbol{x}} \sim p(\boldsymbol{x} \mid \boldsymbol{y}). \tag{7.1.4}$$

Notice that if the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ has a low-dimensional support that is nonlinear, these three different estimates can be rather different, as illustrated in Figure 7.3. Conceptually, the maximum a posteriori (MAP) estimate is the most desired one—it is the sample of the highest probability

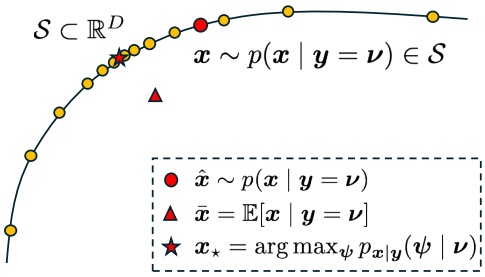


Figure 7.3: Comparison of three different surrogates for an estimate of $\boldsymbol{x}$ given $\boldsymbol{y}$.

to have produced the observation $\boldsymbol{y}$, but it is typically the most expensive to compute.

Notice that from Bayes' rule, we have

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})}. \tag{7.1.5}$$

For instance, the maximal likelihood estimate can be computed by solving the following (maximal log likelihood) program:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} [\log p(\boldsymbol{y} \mid \boldsymbol{x}) + \log p(\boldsymbol{x})], \tag{7.1.6}$$

say via gradient ascent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \cdot \big(\nabla_{\boldsymbol{x}} \log p(\boldsymbol{y} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})\big). \tag{7.1.7}$$

Efficiently computing the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ naturally depends on how we learn and exploit the low-dimensional distribution $p(\boldsymbol{x})$ of the data $\boldsymbol{x}$ and the observation model $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$ that determines the conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$.

Remark 7.1 (End-to-End versus Bayesian Inference). In the modern practice of data-driven machine learning, for certain popular tasks people often directly learn the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ or a (probabilistic) mapping or a regressor. Such a mapping is often modeled by some deep networks and trained end-to-end with sufficient paired samples $(\boldsymbol{x}, \boldsymbol{y})$. Such an approach is very different from the above Bayesian approach in which both the distribution of $\boldsymbol{x} \sim p(\boldsymbol{x})$ and the (observation) mapping are needed. The benefit of the Bayesian approach is that the learned distribution $p(\boldsymbol{x})$ can facilitate many different tasks with varied observation models and conditions.

7.1.2 Constrained Optimization with Submanifolds

Geometric interpretation as constrained optimization. As the support $\mathcal{S}_{\boldsymbol{x}}$ of the distribution of $\boldsymbol{x}$ is low-dimensional, we may assume that there exists

a function F such that

$$F(\boldsymbol{x}) = 0 \qquad \Longleftrightarrow \qquad \boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x}} \tag{7.1.8}$$

such that $\mathcal{S}_{\boldsymbol{x}} = F^{-1}(\{0\})$ is the low-dimensional support of the distribution $p(\boldsymbol{x})$. Geometrically, one natural choice of $F(\boldsymbol{x})$ is the "distance function" to the support $\mathcal{S}_{\boldsymbol{x}}$:

$$F(\boldsymbol{x}) = \min_{\boldsymbol{x}_p \in \mathcal{S}_{\boldsymbol{x}}} \|\boldsymbol{x} - \boldsymbol{x}_p\|_2. \tag{7.1.9}$$

Notice that, in reality, we only have discrete samples on the support of the distribution. In the same spirit of continuation, through diffusion or lossy coding studied in Chapters 3 and 4, we may approximate the distance function as $F(\boldsymbol{x}) \approx \min_{\boldsymbol{x}_p \in \mathcal{C}_{\boldsymbol{x}}^{\epsilon}} \|\boldsymbol{x} - \boldsymbol{x}_p\|_2$ where $\mathcal{S}_{\boldsymbol{x}}$ is replaced by a covering $\mathcal{C}_{\boldsymbol{x}}^{\epsilon}$ of the samples with ϵ-balls. But if the analytical form of a simple distribution is given, sometimes $F(\boldsymbol{x})$ can be computed explicitly.

Example 7.2 (Distance to a Line in $\mathbb{R}^3$). Let us assume that the support of a low-dimensional distribution in $\mathbb{R}^3$ is the x_3-axis. Then the distance function $F(\boldsymbol{x})$ is given by:

$$F(\boldsymbol{x}) = \sqrt{x_1^2 + x_2^2}. \tag{7.1.10}$$

■

To see how such a function plays an important role in exploiting the distribution, for simplicity, we will assume that, for the rest of the subsection, the distance function is already given.

Now given $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$, to solve for $\boldsymbol{x}$, we can solve the following constrained optimization problem:

$$\max_{\boldsymbol{x}} -\frac{1}{2}\|h(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 \quad \text{s.t.} \quad F(\boldsymbol{x}) = 0. \tag{7.1.11}$$

Using the method of augmented Lagrange multipliers, we can solve the following unconstrained program:

$$\max_{\boldsymbol{x}} \left[-\frac{1}{2}\|h(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + \lambda F(\boldsymbol{x}) - \frac{\mu}{2} F(\boldsymbol{x})^2 \right] \tag{7.1.12}$$

for some constant Lagrange multiplier λ. This is equivalent to the following program:

$$\max_{\boldsymbol{x}} \left[\log \exp\Big(-\frac{1}{2}\|h(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 \Big) + \log \exp\Big(-\frac{\mu}{2}\big(F(\boldsymbol{x}) - \lambda/\mu\big)^2 \Big) \right], \tag{7.1.13}$$

where $c \doteq \lambda/\mu$ can be viewed as a "mean" for the constraint function. As μ becomes large when enforcing the constraint via continuation[1], c becomes

[1] In the same spirit of continuation in Chapter 3 where we obtained better approximations of our distribution by sending $\varepsilon \to 0$, here we send $\mu \to \infty$. Larger values of μ will constrain F to take smaller and smaller values at the optimum, meaning that the optimum lies within a smaller and smaller neighborhood of the support $\mathcal{S}_{\boldsymbol{x}}$. Interestingly, the theory of Lagrange multipliers hints that, under certain benign conditions on F and other terms in the objective, we only need to make μ large enough in order to ensure $F(\boldsymbol{x}) = 0$ at the optimum, meaning that at *finite* penalty we get *perfect* approximation of the support. In general, we should have the intuition that μ plays the same role as ε^{-1}.

increasingly small. The above program may be interpreted in two different ways.

Firstly, one may view the first term as the conditional probability of $\boldsymbol{y}$ given $\boldsymbol{x}$, and the second term as a probability density for $\boldsymbol{x}$:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) \propto \exp\Big(-\frac{1}{2}\|h(\boldsymbol{x}) - \boldsymbol{y}\|_2^2\Big), \quad p(\boldsymbol{x}) \propto \exp\Big(-\frac{\mu}{2}(F(\boldsymbol{x}) - c)^2\Big). \tag{7.1.14}$$

Hence, solving the constrained optimization for the inverse problem is equivalent to conducting Bayes inference with the above probability densities. Hence solving the above program (7.1.13) via gradient ascent is equivalent to the above maximal likelihood estimate (7.1.7), in which the gradient takes the form:

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{y} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \tag{7.1.15}$$

$$= -(h(\boldsymbol{x}) - \boldsymbol{y})^\top \frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}) - \mu(F(\boldsymbol{x}) - c)\frac{\partial F}{\partial \boldsymbol{x}}(\boldsymbol{x}), \tag{7.1.16}$$

where $\frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x})$ and $\frac{\partial F}{\partial \boldsymbol{x}}(\boldsymbol{x})$ are the Jacobian of $h(\boldsymbol{x})$ and $F(\boldsymbol{x})$, respectively.

Example 7.3 (Gradient of $F(\boldsymbol{x})$). For the distance function defined in Example 7.2, its gradient is given by

$$\frac{\partial F}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \frac{1}{\sqrt{x_1^2 + x_2^2}} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}. \tag{7.1.17}$$

Notice that unlike regular smooth functions whose gradient is typically unique at a fixed point, the gradients of the distance function at a point in the support, say $(0, 0, x_3)$, can span an entire subspace complementary to the support. ■

Notice that the above (gradient)

$$-\mu(F(\boldsymbol{x}) - c)\frac{\partial F}{\partial \boldsymbol{x}}(\boldsymbol{x})$$

always points towards the low-dimensional support of the distribution. Hence the descent process can be viewed as a "denoising" process, studied in Chapter 3, that gradually enforces the data to be closer to the correct support. Notice that the above derivation suggests that the "step size" of the score $\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$ of the denoising process is

$$\mu\big|F(\boldsymbol{x}) - \lambda/\mu\big|$$

which for any fixed μ is nearly proportional to the distance of the point $\boldsymbol{x}$ to the support.

Secondly, notice that solving the above program (7.1.13) with μ increasing is equivalent to:

$$\boldsymbol{x}_\star = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|h(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + \frac{\mu}{2}\big(F(\boldsymbol{x}) - \lambda/\mu\big)^2 \quad \text{as} \quad \mu \uparrow \infty. \tag{7.1.18}$$

Due to the conspicuous quadratic form of the two terms in the above equation, they can also be interpreted as certain "energy" functions. Such a formulation

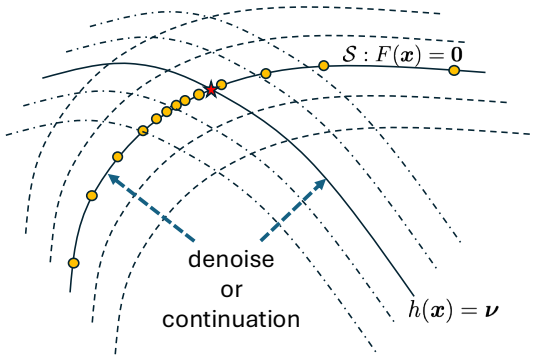


Figure 7.4: Illustration of the continuation process of enforcing the constraints $F(\boldsymbol{x}) = 0$ and $\boldsymbol{y} = h(\boldsymbol{x})$.

is often referred to as "Energy Minimization" in the machine learning literature, advocated by people like Yann LeCun [LCH+06]. Here the energy of a point $\boldsymbol{x}$ is a quadratic function of its distance to the support of the desired distribution. Notice that in Chapter 3 and Chapter 4, we have argued that the notion of entropy is to measure the "volume" or uncertainty of the data distribution whereas here the energy depends on the "distance" of $\boldsymbol{x}$ to the support of the distribution. As we minimize the above energy functions, the entropy (or uncertainty) of the feasible solutions reduces until it reaches (a feasible solution of) the optimal MAP estimate, as illustrated in Figure 7.4.

7.1.3 Representative Practical Settings for Inference

Notice that the above discussion and derivation are based on the assumption that we have perfect knowledge about the function $F(\boldsymbol{x})$ and the observation model $h(\boldsymbol{x})$. In practice, however, they may not be available at all and need to be "learned" from the data given. Hence to make the above conceptual solution truly computable, we need to deal with various situations in which information about the distributions of $\boldsymbol{x}$ and the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ are given or accessible in different ways and forms.

In general, they can mostly be categorized into four cases, which are, conceptually, increasingly more challenging:

- *Case 1:* Both a model for the distribution of $\boldsymbol{x}$ and the observation model $\boldsymbol{y} = h(\boldsymbol{x})$ (possibly with added noise $\boldsymbol{w}$) are known, even with an analytical form. This is typically the case for many classic signal processing problems, such as signal denoising, the sparse vector recovery problem we saw in Chapter 2 and the low-rank matrix recovery problem to be introduced below.

- *Case 2:* We do not have a model for the distribution but only samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of $\boldsymbol{x}$, and the observation model $\boldsymbol{y} = h(\boldsymbol{x})$ (possibly

with added noise $\boldsymbol{w}$) is known.[2] A model for the distribution $p(\boldsymbol{x})$ of $\boldsymbol{x}$ needs to be learned, and subsequently the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$. Natural image completion or natural language completion (e.g., BERT and GPT) are typical examples of this class of problems.

- *Case 3:* We only have some paired samples: $(\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ of the two variables $(\boldsymbol{x}, \boldsymbol{y})$. The distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$ and their relationship $h(\cdot)$ need to be learned from these paired sample data. For example, given many images and their captions, learning to conduct text-conditioned image generation is one such problem.

- *Case 4:* We only have the samples $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ of the observations $\boldsymbol{y}$, and the observation model $h(\cdot)$ needs to be known, at least in some parametric family $h(\cdot, \boldsymbol{\theta})$. The distribution $p(\boldsymbol{x})$ and $p(\boldsymbol{x} \mid \boldsymbol{y})$ need to be learned from $\hat{\boldsymbol{x}}$, estimated from $\boldsymbol{Y}$. For example, learning to render a new view from a sequence of calibrated or uncalibrated views is one such problem.

In this chapter, we will discuss general approaches to learn the desired distributions and solve the associated conditional estimation or generation for these cases, typically with a representative practical problem. Throughout the chapter, the reader should keep Figure 7.1 in mind.

7.2 Conditional Inference with a Known Data Distribution

Notice that in the setting we have discussed in previous Chapters, the autoencoding network is trained to reconstruct a set of samples of the random vector $\boldsymbol{x}$. This would allow us to regenerate samples from the learned (low-dimensional) distribution. In practice, the low-dimensionality of the distribution, once given or learned, can be exploited for stable and robust recovery, completion, or prediction tasks. That is, under rather mild conditions, one can recover $\boldsymbol{x}$ from highly compressive, partial, noisy or even corrupted measures of $\boldsymbol{x}$ of the kind:

$$\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}, \tag{7.2.1}$$

where $\boldsymbol{y}$ is typically an observation of $\boldsymbol{x}$ that is of much lower dimension than $\boldsymbol{x}$ and $\boldsymbol{w}$ can be random noise or even sparse gross corruptions. This is a class of problems that have been extensively studied in the classical signal processing literature, for low-dimensional structures such as sparse vectors, low-rank matrices, and beyond. Interested readers may see [WM22] for a complete exposition of this topic.

Here to put the classic work in a more general modern setting, we illustrate the basic idea and facts through the arguably simplest task of data (and particularly image) completion. That is, we consider the problem of recovering a

[2] In the literature, this setting is sometimes referred to as the *empirical Bayesian inference*.

sample $\boldsymbol{x}$ when parts of it are missing (or even corrupted). We want to recover or predict the rest of $\boldsymbol{x}$ from observing only a fraction of it:

$$f : \mathcal{P}_\Omega(\boldsymbol{x}) \mapsto \hat{\boldsymbol{x}}, \tag{7.2.2}$$

where $\mathcal{P}_\Omega(\,\cdot\,)$ represents a masking operation (see Figure 7.5 for an example).

In this section and the next, we will study the completion task under two different scenarios: One is when the distribution of the data $\boldsymbol{x}$ of interest is already given *a priori*, even in a certain analytical form. This is the case that prevails in classic signal processing where the structures of the signals are assumed to be known, for example, band-limited, sparse or low-rank. The other is when only raw samples of $\boldsymbol{x}$ are available and we need to learn the low-dimensional distribution from the samples in order to solve the completion task well. This is the case for the tasks of natural image completion or video frame prediction. As a precursor to the rest of the chapter, we start with the simplest case of image completion: when the image to be completed can be well modeled as a low-rank matrix. We will move on to increasingly more general cases and more challenging settings later.

Low-rank matrix completion. The low-rank *matrix completion* problem is a classical problem for data completion when its distribution is low-dimensional and known. Consider a random sample of a matrix $\boldsymbol{X}_o = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m \times n}$ from the space of all matrices of rank r. In general, we assume the rank of the matrix is

$$\operatorname{rank}(\boldsymbol{X}_o) = r < \min\{m, n\}. \tag{7.2.3}$$

So it is clear that locally the intrinsic dimension of the space of all matrices of rank r is much lower than the ambient space mn.

Now, let Ω indicate a set of indices of observed entries of the matrix $\boldsymbol{X}_o$. Let the observed entries be:

$$\boldsymbol{Y} = \mathcal{P}_\Omega(\boldsymbol{X}_o). \tag{7.2.4}$$

The remaining entries supported on Ω^c are unobserved or missing. The problem is whether we can recover from $\boldsymbol{Y}$ the missing entries of $\boldsymbol{X}$ correctly and efficiently. Figure 7.5 shows one example of completing such a matrix.

Notice that the fundamental reason why such a matrix can be completed is that columns and rows of the matrix are highly correlated and they all lie on a low-dimensional subspace. For the example shown in Figure 7.5, the dimension or the rank of the matrix completed is only two. Hence the fundamental idea to recover such a matrix is to seek a matrix that has the lowest rank among all matrices that have entries agreeing with the observed ones:

$$\min_{\boldsymbol{X}} \operatorname{rank}(\boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{Y} = \mathcal{P}_\Omega(\boldsymbol{X}). \tag{7.2.5}$$

This is known as the *low-rank matrix completion* problem. See [WM22] for a full characterization of the space of all low-rank matrices. As the rank function is discontinuous and rank minimization is in general an NP-hard problem, we would like to relax it with something easier to optimize.

Figure 7.5: Illustration of completing an image as low-rank matrix with some entries masked or corrupted. Left: the masked/corrupted image $\boldsymbol{Y}$; middle: the mask Ω; right: the completed image $\hat{\boldsymbol{X}}$.

Based on our knowledge about compression from Chapter 3, we could promote the low-rankness of the recovered matrix $\boldsymbol{X}$ by enforcing the lossy coding rate (or the volume spanned by $\boldsymbol{X}$) of the data in $\boldsymbol{X}$ to be small:

$$\min R_\epsilon(\boldsymbol{X}) = \frac{1}{2}\log\det\left(\boldsymbol{I} + \alpha \boldsymbol{X}\boldsymbol{X}^\top\right) \quad \text{subject to} \quad \boldsymbol{Y} = \mathcal{P}_\Omega(\boldsymbol{X}). \tag{7.2.6}$$

The problem can be viewed as a continuous relaxation of the above low-rank matrix completion problem (7.2.5) and it can be solved via gradient descent. One can show that the gradient descent operator for the $\log\det$ objective is precisely minimizing a close surrogate of the rank of the matrix $\boldsymbol{X}\boldsymbol{X}^\top$.

The rate distortion function is a nonconvex function, and its gradient descent does not always guarantee finding the globally optimal solution. Nevertheless, since the underlying structure sought for $\boldsymbol{X}$ is piecewise linear, the rank function admits a rather effective convex relaxation: the nuclear norm—the sum of all singular values of the matrix $\boldsymbol{X}$. As shown in the compressive sensing literature, under fairly broad conditions,[3] the matrix completion problem (7.2.5) can be effectively solved by the following convex program:

$$\min \|\boldsymbol{X}\|_* \quad \text{subject to} \quad \boldsymbol{Y} = \mathcal{P}_\Omega(\boldsymbol{X}), \tag{7.2.7}$$

where the nuclear norm $\|\boldsymbol{X}\|_*$ is the sum of singular values of $\boldsymbol{X}$. In practice, we often convert the above constrained convex optimization program to an unconstrained one:

$$\min \|\boldsymbol{X}\|_* + \lambda\|\boldsymbol{Y} - \mathcal{P}_\Omega(\boldsymbol{X})\|_F^2, \tag{7.2.8}$$

for some properly chosen $\lambda > 0$. Interested readers may refer to [WM22] for how to develop algorithms that can solve the above programs efficiently and effectively. Figure 7.5 shows a real example in which the matrix $\hat{\boldsymbol{X}}$ is actually recovered by solving the above program.

[3] Typically, such conditions specify the necessary and sufficient amount of entries needed for the completion to be computationally feasible. These conditions have been systematically characterized in [WM22].

Further extensions. It has been shown that images (or more accurately textures) and 3D scenes with low-rank structures can be very effectively completed via solving optimization programs of the above kind, even if there is additional corruption and distortion [LRZ+12; YZB+23; ZLG+10]:

$$\boldsymbol{Y} \circ \tau = \boldsymbol{X}_o + \boldsymbol{E}, \tag{7.2.9}$$

where τ is some unknown nonlinear distortion of the image and $\boldsymbol{E}$ is an unknown matrix that models some (sparse) occlusion and corruption. Again, interested readers may refer to [WM22] for a more detailed account.

7.3 Conditional Inference with a Learned Data Representation

In the previous subsection, the reason we can infer $\boldsymbol{x}$ from the partial observation $\boldsymbol{y}$ is because (the support of) the distribution of $\boldsymbol{X}$ is known or specified *a priori*, say as the set of all low-rank matrices. For many practical datasets, we do not have their distribution in an analytical form like the low-rank matrices, say the set of all natural images. Nevertheless, if we have sufficient samples of the data $\boldsymbol{x}$, we should be able to learn its low-dimensional distribution first and leverage it for future inference tasks based on an observation $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$. In this section, we assume the observation model $h(\cdot)$ is given and known. We will study the case when $h(\cdot)$ is not explicitly given in the next section.

7.3.1 Image Completion with Masked Auto-Encoding

For a general image $\boldsymbol{X}$ such as the one shown on the left of Figure 7.6, we can no longer view it as a low-rank matrix. However, humans still demonstrate remarkable ability to complete a scene and recognize familiar objects despite severe occlusion. This suggests that our brain has learned the low-dimensional distribution of natural images and can use it for completion, and hence recognition. However, the distribution of all natural images is not as simple as a low-dimensional linear subspace. Hence a natural question is whether we can learn the more sophisticated distribution of natural images and use it to perform image completion?

One empirical approach to the image completion task is to find an encoding and decoding scheme by solving the following *masked autoencoding* (MAE) program that minimizes the reconstruction loss:

$$\min_{f,g} L_{\text{MAE}}(f, g) \doteq \mathbb{E}\big[\|(g \circ f)(\mathcal{P}_\Omega(\boldsymbol{X})) - \boldsymbol{X}\|_2^2\big]. \tag{7.3.1}$$

Unlike the matrix completion problem which has a simple underlying structure, we should no longer expect that the encoding and decoding mappings admit simple closed forms or the program can be solved by explicit algorithms.

For a general natural image, we can no longer assume that its columns or rows are sampled from a low-dimensional subspace or a low-rank Gaussian.

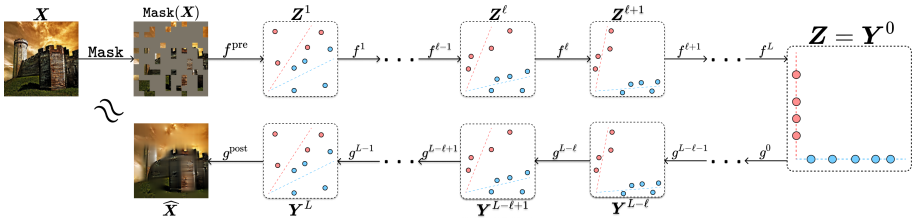


Figure 7.6: **Diagram of the overall (masked) autoencoding process.** The (image) token representations are transformed iteratively towards a parsimonious (e.g., compressed and sparse) representation by each encoder layer f^ℓ. Furthermore, such representations are transformed back to the original image by the decoder layers g^ℓ. Each encoder layer f^ℓ is meant to be (partially) inverted by a corresponding decoder layer $g^{L-\ell}$.

However, it is reasonable to assume that the image consists of multiple regions. Image patches in each region are similar and can be modeled as one (low-rank) Gaussian or subspace. Hence, to exploit the low-dimensionality of the distribution, the objective *of the encoder* f is to transform $\boldsymbol{X}$ to a representation $\boldsymbol{Z}$:

$$f : \boldsymbol{X} \mapsto \boldsymbol{Z} \tag{7.3.2}$$

such that the distribution of $\boldsymbol{Z}$ can be well modeled as a mixture of subspaces, say $\{\boldsymbol{U}_{[K]}\}$, such that the rate reduction is maximized while the sparsity is minimized:

$$\mathbb{E}_{\boldsymbol{Z}=f(\boldsymbol{X})}[\Delta R_\epsilon(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) - \lambda\|\boldsymbol{Z}\|_0] = \mathbb{E}_{\boldsymbol{Z}=f(\boldsymbol{X})}[R_\epsilon(\boldsymbol{Z}) - R^c_\epsilon(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) - \lambda\|\boldsymbol{Z}\|_0], \tag{7.3.3}$$

where the functions $R_\epsilon(\cdot)$ and $R^c_\epsilon(\cdot)$ are defined in (5.2.2) and (5.2.3), respectively.

As we have shown in the previous Chapter 5, the encoder f that minimizes the above objective can be constructed as a sequence of transformer-like operators. As shown in the work of [PBW+24], the decoder g can be viewed and hence constructed explicitly as the inverse process of the encoder f. Figure 7.7 illustrates the overall architectures of both the encoder and the corresponding decoder at each layer. The parameters of the encoder f and decoder g can be learned by optimizing the reconstruction loss (7.3.1) via gradient descent.

Figure 7.8 shows some representative results of the thus-designed masked auto-encoder. More implementation details and results of the masked autoencoder for natural image completion can be found in Chapter 8 Section 8.5.

7.3.2 Conditional Sampling with Measurement Matching

The above (masked) autoencoding problem aims to generate a sample image that is consistent with certain observations or conditions. But let us examine the approach more closely: given the visual part of an image $\boldsymbol{X}_v = \mathcal{P}_\Omega(\boldsymbol{X})$, we

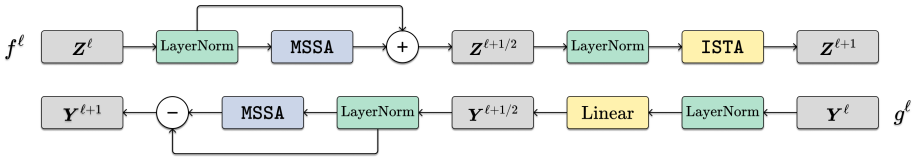


Figure 7.7: **Diagram of each encoder layer (*top*) and decoder layer (*bottom*).** Notice that the two layers are highly anti-parallel — each is constructed to do the operations of the other in reverse order. That is, in the decoder layer g^ℓ, the ISTA block of $f^{L-\ell}$ is partially inverted first using a linear layer, then the MSSA block of $f^{L-\ell}$ is reversed; this order unravels the transformation done in $f^{L-\ell}$.

Figure 7.8: **Autoencoding visualizations of CRATE-Base and ViT-MAE-Base [HCX+22] with 75% patches masked.** We observe that the reconstructions from CRATE-Base are on par with the reconstructions from ViT-MAE-Base, despite using $< 1/3$ of the parameters.

try to estimate the masked part $\boldsymbol{X}_m = \mathcal{P}_{\Omega^c}(\boldsymbol{X})$. For realizations $(\boldsymbol{\Xi}_v, \boldsymbol{\Xi}_m)$ of the random variable $\boldsymbol{X} = (\boldsymbol{X}_v, \boldsymbol{X}_m)$, let

$$p_{\boldsymbol{X}_m|\boldsymbol{X}_v}(\boldsymbol{\Xi}_m \mid \boldsymbol{\Xi}_v)$$

be the conditional distribution of $\boldsymbol{X}_m$ given $\boldsymbol{X}_v$. It is easy to show that the optimal solution to the MAE formulation (7.3.1) is given by the conditional expectation:

$$\arg\min_{h=g\circ f} L_{\text{MAE}}(h) = \boldsymbol{\Xi}_v \mapsto \boldsymbol{\Xi}_v + \mathbb{E}[\boldsymbol{X}_m \mid \boldsymbol{X}_v = \boldsymbol{\Xi}_v]. \tag{7.3.4}$$

In general, however, this expectation may not even lie on the low-dimensional distribution of natural images! This partially explains why some of the recovered patches in Figure 7.8 are a little blurry.

For many practical purposes, we would like to learn (a representation of) the conditional distribution $p_{\boldsymbol{X}_m|\boldsymbol{X}_v}$, or equivalently $p_{\boldsymbol{X}|\boldsymbol{X}_v}$, and then get a clear (most likely) sample from this distribution directly. Notice that, when the distribution of $\boldsymbol{X}$ is low-dimensional, it is possible that if a sufficient part of $\boldsymbol{X}$, $\boldsymbol{X}_v$, is observed, it fully determines $\boldsymbol{X}$ and hence the missing part $\boldsymbol{X}_m$. In other words, the distribution $p_{\boldsymbol{X}|\boldsymbol{X}_v}$ is a generalized function (analogous to a delta function).

Figure 7.9: **Sampling visualizations from models trained via ambient diffusion [DSD+23b] with 80% of the pixels masked.** Using a similar ratio of masked pixels as in Figure 7.8, the ambient diffusion sampling algorithm recovers a much sharper image than the blurry image recovered by the MAE-based method. The former method samples from the distribution of natural images, while the latter approximates the conditional expectation (i.e., average) of this distribution given the observation; this averaging causes the blurriness.

Hence, instead of solving the completion task as a conditional estimation problem, we should address it as a conditional sampling problem. To that end, we should first learn the (low-dimensional) distribution of all natural images $\boldsymbol{X}$. If we have sufficient samples of natural images, we can learn the distribution via a denoising process $\boldsymbol{X}_t$ described in Chapter 3. Then the problem of recovering $\boldsymbol{X}$ from its partial observation $\boldsymbol{Y} = \mathcal{P}_\Omega(\boldsymbol{x}) + \boldsymbol{w}$ becomes a conditional generation problem—to sample the distribution conditioned on the observation.

General linear measurements. In fact, we may even consider recovering $\boldsymbol{X}$ from a more general linear observation model:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}_0, \quad \boldsymbol{X}_t = \boldsymbol{X}_0 + \sigma_t \boldsymbol{G}, \tag{7.3.5}$$

where $\boldsymbol{A}$ is a linear operator on matrix space[4] and $\boldsymbol{G} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The masking operator $\mathcal{P}_\Omega(\cdot)$ in the image completion task is one example of such a linear model. Then it has been shown by [DSD+23a] that

$$\hat{\boldsymbol{X}}_* = \arg\min_{\hat{\boldsymbol{X}}} \mathbb{E}[\|\boldsymbol{A}(\hat{\boldsymbol{X}}(\boldsymbol{A}\boldsymbol{X}_t, \boldsymbol{A}) - \boldsymbol{X}_0)\|^2] \tag{7.3.6}$$

satisfies the condition that:

$$\boldsymbol{A}\hat{\boldsymbol{X}}_*(\boldsymbol{A}(\boldsymbol{X}_t), \boldsymbol{A}) = \boldsymbol{A}\mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{A}\boldsymbol{X}_t, \boldsymbol{A}]. \tag{7.3.7}$$

Notice that in the special case when $\boldsymbol{A}$ is of full column rank, we have $\mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{A}\boldsymbol{X}_t, \boldsymbol{A}] = \mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{X}_t]$. Hence, in the more general case, it has been suggested by [DSD+23a] that one could still use the so obtained $\mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{A}(\boldsymbol{X}_t), \boldsymbol{A}]$ to replace the $\mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{X}_t]$ in the normal denoising process for $\boldsymbol{X}_t$:

$$\boldsymbol{X}_{t-s} = \gamma_t \boldsymbol{X}_t + (1 - \gamma_t)\mathbb{E}[\boldsymbol{X}_0 \mid \boldsymbol{A}\boldsymbol{X}_t, \boldsymbol{A}]. \tag{7.3.8}$$

[4] i.e., if we imagine unrolling $\boldsymbol{X}$ into a long vector then $\boldsymbol{A}$ takes the role of a matrix on $\boldsymbol{X}$-space

This usually works very well in practice, say for many image restoration tasks, as shown in [DSD+23a]. Compared to the blurry images recovered from MAE, the images recovered by the above method are much sharper as it leverages a learned distribution of natural images and samples a (sharp) image from the distribution that is consistent with the measurement, as shown in Figure 7.9 (cf. Figure 7.8).

General nonlinear measurements. To generalize the above (image) completion problems and make things more rigorous, we may consider that a random vector $\boldsymbol{x} \sim p$ is partially observed through a more general observation function:

$$\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}, \tag{7.3.9}$$

where $\boldsymbol{w}$ usually stands for some random measurement noise, say of a Gaussian distribution $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$. It is easy to see that, for $\boldsymbol{x}$ and $\boldsymbol{y}$ so related, their joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ is naturally nearly degenerate if the noise $\boldsymbol{w}$ is small. To a large extent, we may view $p(\boldsymbol{x}, \boldsymbol{y})$ as a noisy version of a hypersurface defined by the function $\boldsymbol{y} = h(\boldsymbol{x})$ in the joint space $(\boldsymbol{x}, \boldsymbol{y})$. Practically speaking, we will consider a setting more akin to masked autoencoding than to pure matrix completion, where we always have access to a corresponding clean sample $\boldsymbol{x}$ for every observation $\boldsymbol{y}$ we receive.[5]

Like image/matrix completion, we are often faced with a setting where $\boldsymbol{y}$ denotes a degraded or otherwise "lossy" observation of the input $\boldsymbol{x}$. This can manifest in quite different forms. For example, in various scientific or medical imaging problems, the measured data $\boldsymbol{y}$ may be a compressed and corrupted observation of the underlying data $\boldsymbol{x}$; whereas in 3D vision tasks, $\boldsymbol{y}$ may represent an image captured by a camera of a physical object with an unknown (low-dimensional) pose $\boldsymbol{x}$. Generally, by virtue of mathematical modeling (and, in some cases, co-design of the measurement system), we know h and can evaluate it on any input, and we can exploit this knowledge to help reconstruct and sample $\boldsymbol{x}$.

At a technical level, we want the learned representation of the data to facilitate us to sample the conditional distribution $p_{\boldsymbol{x}|\boldsymbol{y}}$, also known as the posterior, effectively and efficiently. More precisely, write $\boldsymbol{\nu}$ to denote a realization of the random variable $\boldsymbol{y}$. We want to generate samples $\hat{\boldsymbol{x}}$ such that:

$$\hat{\boldsymbol{x}} \sim p_{\boldsymbol{x}|\boldsymbol{y}}(\,\cdot \mid \boldsymbol{y} = \boldsymbol{\nu}). \tag{7.3.10}$$

Recall that in Section 3.2, we have developed a natural and effective way to produce *unconditional* samples of the data distribution p. The ingredients are the denoisers $\bar{\boldsymbol{x}}^*(t, \boldsymbol{\xi}) = \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$, or their learned approximations $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})$, for different levels of noisy observations $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}$ (and $\boldsymbol{\xi}$ for their realizations) under Gaussian noise $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $t \in [0, T]$ with a choice of times $0 = t_1 <$

[5] In some more specialized applications, in particular in scientific imaging, it is of interest to be able to learn to generate samples from the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$ without access to any clean/ground-truth samples of $\boldsymbol{x}$. We give a brief overview of methods for this setting in the end-of-chapter notes.

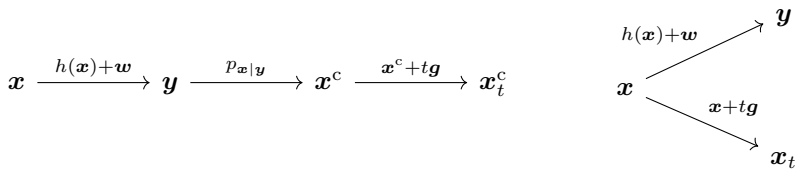


Figure 7.10: Statistical dependency diagrams for the conditional sampling process. **Left:** In a direct (conceptual) application of the diffusion-denoising scheme we have developed in Chapter 3 to conditional sampling, we use samples from the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$ to train denoisers directly on the posterior at different noise levels, then use them to generate new samples. In practice, however, we do not normally have direct samples from the posterior, but rather paired samples $(\boldsymbol{x}, \boldsymbol{y})$ from the joint. **Right:** It turns out that it suffices to have only noisy observations of $\boldsymbol{x}$ to realize the denoisers corresponding to $p_{\boldsymbol{x}_t^{\mathrm{c}}|\boldsymbol{x}^{\mathrm{c}}}$: this follows from conditional independence of $\boldsymbol{x}_t$ and $\boldsymbol{y}$ given $\boldsymbol{x}$. It implies that $p_{\boldsymbol{x}_t^{\mathrm{c}}|\boldsymbol{y}} = p_{\boldsymbol{x}_t|\boldsymbol{y}}$, which gives a score function for denoising that consists of the unconditional score function, plus a correction term that enforces measurement consistency.

$\cdots < t_L = T$ at which to perform the iterative denoising, starting from $\hat{\boldsymbol{x}}_{t_L} \sim \mathcal{N}(\boldsymbol{0}, T^2\boldsymbol{I})$ (recall Equation (3.2.82)).[6] We could directly apply this scheme to generate samples from the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$ *if* we had access to a dataset of samples $\boldsymbol{x}^{\mathrm{c}} \sim p_{\boldsymbol{x}|\boldsymbol{y}}(\,\cdot \mid \boldsymbol{\nu})$ for each realization $\boldsymbol{\nu}$ of $\boldsymbol{y}$, by generating noisy observations $\boldsymbol{x}_t^{\mathrm{c}}$ and training denoisers to approximate $\mathbb{E}[\boldsymbol{x}^{\mathrm{c}} \mid \boldsymbol{x}_t^{\mathrm{c}} = \cdot\,, \boldsymbol{y} = \boldsymbol{\nu}]$, the mean of the posterior under the noisy observation (see Figure 7.10(a)). However, performing this resampling given only paired samples $(\boldsymbol{x}, \boldsymbol{y})$ from the joint distribution (say by binning the samples over values of $\boldsymbol{y}$) requires prohibitively many samples for high-dimensional data, and alternate approaches explicitly or implicitly rely on density estimation, which similarly suffers from the curse of dimensionality.

Fortunately, it turns out that this is not necessary. Consider the alternate statistical dependency diagram in Figure 7.10(b), which corresponds to the random variables in the usual denoising-diffusion process, together with the measurement $\boldsymbol{y}$. Because our assumed observation model (7.3.9) implies that

[6] Recall from our discussion in Section 3.2.2 that a few small improvements to this basic iterative denoising scheme are sufficient to bring competitive practical performance. For clarity as we develop conditional sampling, we will focus here on the simplest instantiation.

$\boldsymbol{x}_t$ and $\boldsymbol{y}$ are independent conditioned on $\boldsymbol{x}$, we have for any realization $\boldsymbol{\nu}$ of $\boldsymbol{y}$

$$\begin{aligned} p_{\boldsymbol{x}_t^{\mathrm{c}}|\boldsymbol{y}}(\cdot \mid \boldsymbol{\nu}) &= \int \underbrace{p_{\boldsymbol{x}_t^{\mathrm{c}}|\boldsymbol{x}^{\mathrm{c}}}(\cdot \mid \boldsymbol{\xi})}_{=\mathcal{N}(\boldsymbol{\xi}, t^2\boldsymbol{I})} \cdot \underbrace{p_{\boldsymbol{x}^{\mathrm{c}}|\boldsymbol{y}}}_{=p_{\boldsymbol{x}|\boldsymbol{y}}}(\boldsymbol{\xi} \mid \boldsymbol{\nu}) \mathrm{d}\boldsymbol{\xi} \\ &= \int p_{\boldsymbol{x}_t|\boldsymbol{x},\boldsymbol{y}}(\cdot \mid \boldsymbol{\xi}, \boldsymbol{\nu}) \cdot p_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{\xi} \mid \boldsymbol{\nu}) \mathrm{d}\boldsymbol{\xi} \\ &= \int p_{\boldsymbol{x}_t,\boldsymbol{x}|\boldsymbol{y}}(\cdot, \boldsymbol{\xi} \mid \boldsymbol{\nu}) \mathrm{d}\boldsymbol{\xi} \\ &= p_{\boldsymbol{x}_t|\boldsymbol{y}}(\cdot \mid \boldsymbol{\nu}). \end{aligned} \tag{7.3.11}$$

Above, the first line recognizes an equivalence between the distributions arising in Figure 7.10 (a,b); the second line applies this together with conditional independence of $\boldsymbol{x}_t$ and $\boldsymbol{y}$ given $\boldsymbol{x}$; the third line uses the definition of conditional probability; and the final line marginalizes over $\boldsymbol{x}$. Thus, the denoisers from the conceptual posterior sampling process are equal to $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \cdot, \boldsymbol{y} = \boldsymbol{\nu}]$, which we can learn solely from paired samples $(\boldsymbol{x}, \boldsymbol{y})$, and by Tweedie's formula (Theorem 3.2), we can express these denoisers in terms of the score function of $p_{\boldsymbol{x}_t|\boldsymbol{y}}$, which, by Bayes' rule, satisfies

$$p_{\boldsymbol{x}_t|\boldsymbol{y}}(\boldsymbol{\xi} \mid \boldsymbol{\nu}) = \frac{p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) p_{\boldsymbol{x}_t}(\boldsymbol{\xi})}{p_{\boldsymbol{y}}(\boldsymbol{\nu})}. \tag{7.3.12}$$

Recall that the density of $\boldsymbol{x}_t$ is given by $p_t = \varphi_t * p$, where φ_t denotes the standard Gaussian density with zero mean and covariance $t^2\boldsymbol{I}$ and $*$ denotes convolution. This is nothing but the *unconditional score function* obtained from the standard diffusion training that we developed in Section 3.2! The conditional score function then satisfies, for any realization $(\boldsymbol{\xi}, \boldsymbol{\nu})$ of $(\boldsymbol{x}_t, \boldsymbol{y})$,

$$\nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{x}_t|\boldsymbol{y}}(\boldsymbol{\xi} \mid \boldsymbol{\nu}) = \underbrace{\nabla_{\boldsymbol{\xi}} \log p_t(\boldsymbol{\xi})}_{\text{score matching}} + \underbrace{\nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi})}_{\text{measurement matching}}, \tag{7.3.13}$$

giving (by Tweedie's formula) our proposed denoisers as

$$\begin{aligned} \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, \boldsymbol{y} = \boldsymbol{\nu}] &= \boldsymbol{\xi} + t^2 \nabla_{\boldsymbol{\xi}} \log p_t(\boldsymbol{\xi}) + t^2 \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) \\ &= \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] + t^2 \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}). \end{aligned} \tag{7.3.14}$$

The resulting operators are interpretable as a *corrected* version of the unconditional denoiser for the noisy observation, where the correction term (the so-called "measurement matching" term) enforces consistency with the observations $\boldsymbol{y}$. This decomposition is the basis of *classifier guidance*, which we will develop in detail in Section 7.4.1. The reader should take care to note to which argument the gradient operators are applying in the above score functions in order to fully grasp the meaning of this operator.

The key remaining issue in making this procedure computational is to prescribe how to compute the measurement matching correction, since in general

we do not have a closed-form expression for the likelihood $p_{\boldsymbol{y}|\boldsymbol{x}_t}$ except for when $t = 0$. Before taking up this problem, we discuss an illustrative concrete example of the entire process, continuing from those we have developed in Section 3.2.

Example 7.4. Consider the case where the data distribution is Gaussian with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$, i.e., $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Assume that $\boldsymbol{\Sigma} \succeq \mathbf{0}$ is nonzero. Moreover, in the measurement model (7.3.9), suppose we obtain linear measurements of $\boldsymbol{x}$ with independent Gaussian noise, where $\boldsymbol{A} \in \mathbb{R}^{d\times D}$ and $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \sigma\boldsymbol{w}$ with $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ independent of $\boldsymbol{x}$. Below, we are going to work out the following consequences of this model for the general framework we have derived above. Specifically:

1. We will work out the precise forms of the posterior distribution $p_{\boldsymbol{x}|\boldsymbol{y}}$ that we are aiming to sample from.
2. We will work out the form of the measurement matching term from (7.3.13), which gives us the additive correction to the unconditional denoiser for $p_{\boldsymbol{x}|\boldsymbol{x}_t}$ that yields the denoiser for $p_{\boldsymbol{x}|\boldsymbol{x}_t,\boldsymbol{y}}$ (recall (7.3.14)).
3. We will assess the specific form of the measurement matching term we derived in the previous step, and posit a natural approximation that makes it far less sample and compute intensive to learn the measurement matching term from data. This approximation will be applied in greater generality, after the conclusion of this example.

First, we note that $\boldsymbol{x} \overset{\mathrm{d}}{=} \boldsymbol{\Sigma}^{1/2}\boldsymbol{g} + \boldsymbol{\mu}$, where $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is independent of $\boldsymbol{w}$ and $\boldsymbol{\Sigma}^{1/2}$ is the unique positive square root of the covariance matrix $\boldsymbol{\Sigma}$, and after some algebra, we can then write

$$\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} \overset{\mathrm{d}}{=} \begin{bmatrix}\boldsymbol{\Sigma}^{1/2} & \mathbf{0}\\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \sigma\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{g}\\ \boldsymbol{w}\end{bmatrix} + \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{A}\boldsymbol{\mu}\end{bmatrix}.$$

By independence, we have that $(\boldsymbol{g}, \boldsymbol{w})$ is jointly Gaussian, which means that $(\boldsymbol{x}, \boldsymbol{y})$ is also jointly Gaussian, as the affine image of a jointly Gaussian vector. Its covariance matrix is given by

$$\begin{bmatrix}\boldsymbol{\Sigma}^{1/2} & \mathbf{0}\\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \sigma\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}^{1/2} & \mathbf{0}\\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \sigma\boldsymbol{I}\end{bmatrix}^\top = \begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{A}^\top\\ \boldsymbol{A}\boldsymbol{\Sigma} & \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top + \sigma^2\boldsymbol{I}\end{bmatrix}.$$

Now, we apply the fact that conditioning a random vector with joint Gaussian distribution on a subset of coordinates is again a Gaussian distribution (Exercise 3.2). By this, we obtain that

$$p_{\boldsymbol{x}|\boldsymbol{y}}(\,\cdot\mid\boldsymbol{\nu}) = \mathcal{N}\Bigg(\underbrace{\boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{A}^\top\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top + \sigma^2\boldsymbol{I}\right)^{-1}(\boldsymbol{\nu} - \boldsymbol{A}\boldsymbol{\mu})}_{\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{\nu})}, \underbrace{\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{A}^\top\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}\boldsymbol{\Sigma}}_{\boldsymbol{\Sigma}_{\boldsymbol{x}|\boldsymbol{y}}}\Bigg). \tag{7.3.15}$$

Next, since $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}'$, where $\boldsymbol{g}' \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is independent of all other random vectors, and since $\boldsymbol{x} \mid \boldsymbol{y}$ is Gaussian, by the calculation directly above, it follows by another application of Exercise 3.2 that $\boldsymbol{x} \mid \boldsymbol{x}_t, \boldsymbol{y}$ is Gaussian. Its conditional expectation function is given by reading off the mean of the Gaussian distribution, which in this case is

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, \boldsymbol{y} = \boldsymbol{\nu}] = \boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{\nu}) + \boldsymbol{\Sigma}_{\boldsymbol{x}|\boldsymbol{y}} \left(\boldsymbol{\Sigma}_{\boldsymbol{x}|\boldsymbol{y}} + t^2\boldsymbol{I}\right)^{-1} \left(\boldsymbol{\xi} - \boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{\nu})\right). \quad (7.3.16)$$

The functional form of this denoiser is quite simple, but it carries an unwieldy dependence on the problem data $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{A}$, and σ^2. We can gain further insight into its behavior by comparing it with Equation (7.3.14). We have as usual

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] = \boldsymbol{\mu} + \boldsymbol{\Sigma} \left(\boldsymbol{\Sigma} + t^2\boldsymbol{I}\right)^{-1} (\boldsymbol{\xi} - \boldsymbol{\mu}), \quad (7.3.17)$$

which is rather simple—suggesting that the measurement matching term is rather complicated. To confirm this, we can calculate the likelihood $p_{\boldsymbol{y}|\boldsymbol{x}_t}$ directly using the following expression for the joint distribution of $(\boldsymbol{x}, \boldsymbol{x}_t, \boldsymbol{y})$:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x}_t \\ \boldsymbol{y} \end{bmatrix} \overset{\mathrm{d}}{=} \begin{bmatrix} \boldsymbol{\Sigma}^{1/2} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\Sigma}^{1/2} & t\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \boldsymbol{0} & \sigma\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{g}' \\ \boldsymbol{w} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \boldsymbol{A}\boldsymbol{\mu} \end{bmatrix}, \quad (7.3.18)$$

where $\boldsymbol{g}' \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ independent of the other Gaussians. This is again a jointly Gaussian distribution; restricting to only the final two rows, we have the covariance

$$\begin{bmatrix} \boldsymbol{\Sigma}^{1/2} & t\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \boldsymbol{0} & \sigma\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}^{1/2} & t\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{\Sigma}^{1/2} & \boldsymbol{0} & \sigma\boldsymbol{I} \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{\Sigma} + t^2\boldsymbol{I} & \boldsymbol{\Sigma}\boldsymbol{A}^\top \\ \boldsymbol{A}\boldsymbol{\Sigma} & \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top + \sigma^2\boldsymbol{I} \end{bmatrix}.$$

Another application of Exercise 3.2 then gives us

$$\begin{aligned} p_{\boldsymbol{y}|\boldsymbol{x}_t}(\,\cdot \mid \boldsymbol{\xi}) = \mathcal{N}\Bigg(& \underbrace{\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{\Sigma} \left(\boldsymbol{\Sigma} + t^2\boldsymbol{I}\right)^{-1} (\boldsymbol{\xi} - \boldsymbol{\mu})}_{\boldsymbol{\mu}_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\xi})}, \\ & \underbrace{\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top + \sigma^2\boldsymbol{I} - \boldsymbol{A}\boldsymbol{\Sigma} \left(\boldsymbol{\Sigma} + t^2\boldsymbol{I}\right)^{-1} \boldsymbol{\Sigma}\boldsymbol{A}^\top}_{\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}_t}} \Bigg). \end{aligned} \quad (7.3.19)$$

Now notice that $\boldsymbol{\mu}_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\xi}) = \boldsymbol{A}\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$. So, by the chain rule,

$$\begin{aligned} & t^2 \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) \\ = \; & t^2 \nabla_{\boldsymbol{\xi}} \left[-\frac{1}{2} (\boldsymbol{\nu} - \boldsymbol{A}\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}])^\top \boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}_t}^{-1} (\boldsymbol{\nu} - \boldsymbol{A}\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]) \right] \\ = \; & t^2 (\boldsymbol{\Sigma} + t^2\boldsymbol{I})^{-1} \boldsymbol{\Sigma}\boldsymbol{A}^\top \boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}_t}^{-1} \left(\boldsymbol{\nu} - \boldsymbol{A}\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]\right). \end{aligned} \quad (7.3.20)$$

This gives us a more interpretable decomposition of the conditional posterior denoiser (7.3.16): following Equation (7.3.14), it is the sum of the unconditional posterior denoiser (7.3.17) and the measurement matching term (7.3.20).

We can further analyze the measurement matching term to understand cases under which it can be approximated. Notice that

$$\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}_t} = \sigma^2 \boldsymbol{I} + \boldsymbol{A}\boldsymbol{\Sigma}^{1/2}\left(\boldsymbol{I} - \boldsymbol{\Sigma}^{1/2}\left(\boldsymbol{\Sigma} + t^2\boldsymbol{I}\right)^{-1}\boldsymbol{\Sigma}^{1/2}\right)\boldsymbol{\Sigma}^{1/2}\boldsymbol{A}^\top. \tag{7.3.21}$$

If we let $\boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ denote an eigenvalue decomposition of $\boldsymbol{\Sigma}$, where $(\boldsymbol{v}_i)$ are the columns of $\boldsymbol{V}$, we can further write

$$\boldsymbol{\Sigma}^{1/2}\left(\boldsymbol{I} - \boldsymbol{\Sigma}^{1/2}\left(\boldsymbol{\Sigma} + t^2\boldsymbol{I}\right)^{-1}\boldsymbol{\Sigma}^{1/2}\right)\boldsymbol{\Sigma}^{1/2} = t^2\boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\left(\boldsymbol{\Lambda} + t^2\boldsymbol{I}\right)^{-1}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^\top \tag{7.3.22}$$

$$= t^2\sum_{i=1}^{D}\frac{\lambda_i}{\lambda_i + t^2}\boldsymbol{v}_i\boldsymbol{v}_i^*. \tag{7.3.23}$$

Then for any eigenvalue of $\boldsymbol{\Sigma}$ equal to zero, the corresponding summand is zero; and writing $\lambda_{\min}(\boldsymbol{\Sigma})$ for the smallest positive eigenvalue of $\boldsymbol{\Sigma}$, we have that whenever $t \ll \sqrt{\lambda_{\min}(\boldsymbol{\Sigma})}$, it holds

$$\frac{\lambda_i t^2}{\lambda_i + t^2} \approx 0. \tag{7.3.24}$$

So, when $t \ll \sqrt{\lambda_{\min}(\boldsymbol{\Sigma})}$, we have the approximation

$$\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}_t} \approx \sigma^2\boldsymbol{I}. \tag{7.3.25}$$

The right-hand side of this approximation is equal to $\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}}$. So we have in turn

$$\nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) \approx \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{\nu} \mid \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]). \tag{7.3.26}$$

■

Let us take some time to interpret the results of Example 7.4 before we generalize it. The approximation (7.3.26) is, of course, a direct consequence of the specific modeling assumptions we have made in Example 7.4. However, notice that if we directly interpret this approximation, it is *ab initio* tractable: the likelihood $p_{\boldsymbol{y}|\boldsymbol{x}} = \mathcal{N}(\boldsymbol{A}\boldsymbol{x}, \sigma^2\boldsymbol{I})$ is a simple Gaussian distribution centered at the observation, and the approximation to the measurement matching term that we arrive at can be interpreted as simply evaluating the log-likelihood at the conditional expectation $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$, then taking gradients with respect to $\boldsymbol{\xi}$ (which involves backpropagating through the conditional expectation).

To gain insight into the effect of the convenient approximation (7.3.26), we implement and simulate a simple numerical experiment in the Gaussian setting in Figure 7.11. The sampler we implement is a direct implementation of the simple scheme (3.2.82) we have developed in Chapter 3 and recalled above, using the true conditional posterior denoiser, i.e. Equation (7.3.16) (samples are marked with blue circles), and the convenient approximation to this denoiser made through the measurement matching approximation (7.3.26) (samples are

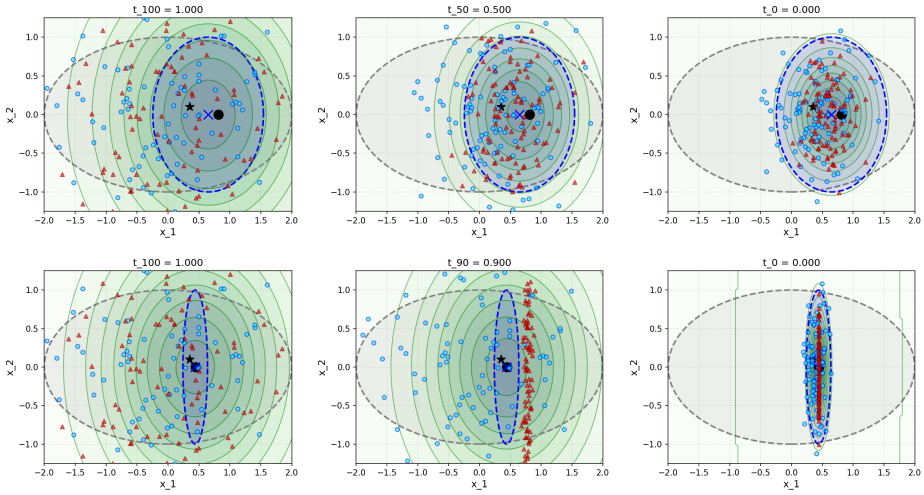


Figure 7.11: Numerical simulation of the conditional sampling setup (7.3.9), with Gaussian data, linear measurements, and Gaussian noise. We simulate $D = 2$ and $d = 1$, with $\boldsymbol{\Sigma} = \boldsymbol{e}_1\boldsymbol{e}_1^\top + \frac{1}{4}\boldsymbol{e}_2\boldsymbol{e}_2^\top$, $\boldsymbol{\mu} = \mathbf{0}$, and $\boldsymbol{A} = \boldsymbol{e}_1^\top$. The underlying signal $\boldsymbol{x}$ is marked with a black star, and the measurement $\boldsymbol{y}$ is marked with a black circle. Each individual plot corresponds to a different value of sampler time t_ℓ, with different rows corresponding to different observation noise levels σ^2. In each plot, the covariance matrix of $\boldsymbol{x}$ is plotted in gray, the posterior covariance matrix and posterior mean of $p_{\boldsymbol{x}|\boldsymbol{y}}$ are plotted in blue (with the posterior mean marked by a blue "x"), and contours for $p_{\boldsymbol{x}_{t_\ell}|\boldsymbol{y}}$ are drawn in green. The sampler hyperparameters are $T = 1$, $L = 100$, and we draw 100 independent samples to initialize the samplers. Samplers are implemented with the closed-form denoisers derived in Example 7.4, with those using the approximation (7.3.26) marked with red triangles, and those using the exact conditional posterior denoiser marked with blue circles. **Top:** For large observation noise $\sigma = 0.5$, both the exact conditional posterior denoiser and the approximate one do a good job of converging to the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$. Sampling time (corresponding to time in the "forward process", so larger times mean larger noise) decreases from left to right. The convergence dynamics for the exact and approximate measurement matching term are similar. **Bottom:** For smaller observation noise $\sigma = 0.1$, the approximate measurement matching term leads to extreme bias in the sampler (red triangles): samples rapidly converge to an affine subspace of points that are consistent, modulo some shrinkage from the posterior mean denoiser, with the measured ground truth, and later sampling iterations are unable to recover the lost posterior variance along this dimension. Note that different times t_ℓ are plotted in the bottom row, compared to the top row, to show the rapid collapse of the approximation to the posterior along the measurement dimension.

marked with red triangles). The top row of Figure 7.11 shows a setting with large measurement noise σ^2, and the bottom with small measurement noise. The measurement matching approximation works very well in the large-noise setting, with the caveat that in the small-noise setting, it suffers from rapid collapse of the variance of the sampling distribution along directions that are parallel to the rows of the linear measurement operator $\boldsymbol{A}$, which cannot be corrected by later iterations of sampling. Our analysis in Example 7.4 precisely characterizes the level at which the noise becomes too "small" in terms of the data covariance matrix $\boldsymbol{\Sigma}$.

Generalizing the approximation: Diffusion Posterior Sampling. Example 7.4 suggests a convenient approximation for the measurement matching term, i.e. (7.3.26), which can be made beyond the Gaussian setting of the example. To motivate this approximation in greater generality, notice that by conditional independence of $\boldsymbol{y}$ and $\boldsymbol{x}_t$ given $\boldsymbol{x}$, we can write

$$p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) = \int p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{\nu} \mid \boldsymbol{\xi}') p_{\boldsymbol{x}|\boldsymbol{x}_t}(\boldsymbol{\xi}' \mid \boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}'. \tag{7.3.27}$$

Formally, when the posterior $p_{\boldsymbol{x}|\boldsymbol{x}_t}$ is a delta function centered at its mean $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$, the approximation (7.3.26) is exact. More generally, when the posterior $p_{\boldsymbol{x}|\boldsymbol{x}_t}$ is highly concentrated around its mean, the approximation (7.3.26) is accurate. This holds, for example, for sufficiently small t, which we saw explicitly in the Gaussian setting of Example 7.4. Although the numerical simulation in Figure 7.11 suggests that this approximation is not without its caveats in certain regimes, it has proved to be a reliable baseline in practice, after being proposed by Chung et al. as "Diffusion Posterior Sampling" (DPS) [CKM+23]. In addition, there are even principled and generalizable approaches to improve it by incorporating better estimates of the posterior variance (which turn out to be exact in the Gaussian setting of Example 7.4), which we discuss further in the end-of-chapter summary.

Thus, with the DPS approximation, we arrive at the following approximation for the conditional posterior denoisers $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{x}_t]$, via Equation (7.3.14):

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, \boldsymbol{y} = \boldsymbol{\nu}] \approx \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] + t^2 \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{\nu} \mid \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]). \tag{7.3.28}$$

And, for a neural network or other model $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})$ trained as in Section 3.2 to approximate the denoisers $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$ for each $t \in [0, T]$, we arrive at the learned conditional posterior denoisers

$$\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi}, \boldsymbol{\nu}) = \bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi}) + t^2 \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{\nu} \mid \bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})). \tag{7.3.29}$$

Note that the approximation (7.3.28) is valid for arbitrary forward models h in the observation model (7.3.9), including nonlinear h, and even to arbitrary noise models for which a clean expression for the likelihood $p_{\boldsymbol{y}|\boldsymbol{x}}$ is known. Indeed, in the case of Gaussian noise, we have

$$p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) \propto \exp\left(-\frac{1}{2\sigma^2} \left\| h(\boldsymbol{\xi}) - \boldsymbol{\nu} \right\|_2^2\right). \tag{7.3.30}$$

Hence, evaluating the right-hand side of (7.3.29) requires only

1. A pretrained denoiser $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})$ for the data distribution p (of $\boldsymbol{x}$), learned as in Section 3.2 via Algorithm 3.2;
2. Forward and backward pass access to the forward model h for the measurements (7.3.9);
3. A forward and backward pass through $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})$, which can be evaluated efficiently using (say) backpropagation.

Algorithm 7.1 Conditional sampling under measurements (7.3.9), with an unconditional denoiser and DPS.

Input: An ordered list of timesteps $0 \le t_0 < \cdots < t_L \le T$ to use for sampling.
Input: An unconditional denoiser $\bar{\boldsymbol{x}}_\theta \colon \{t_\ell\}_{\ell=1}^L \times \mathbb{R}^D \to \mathbb{R}^D$ for $p_{\boldsymbol{x}}$.
Input: Measurement realization $\boldsymbol{\nu}$ of $\boldsymbol{y}$ (Equation (7.3.9)) to condition on.
Input: Forward model $h : \mathbb{R}^D \to \mathbb{R}^d$ and measurement noise variance $\sigma^2 > 0$.
Input: Scale and noise level functions $\alpha, \sigma \colon \{t_\ell\}_{\ell=0}^L \to \mathbb{R}_{\ge 0}$.
Output: A sample $\hat{\boldsymbol{x}}$, approximately from $p_{\boldsymbol{x}|\boldsymbol{y}}$.

function DDIMSamplerConditionalDPS($\bar{\boldsymbol{x}}_\theta, \boldsymbol{\nu}, h, \sigma^2, (t_\ell)_{\ell=0}^L$)
  Initialize $\hat{\boldsymbol{x}}_{t_L} \sim$ approximate distribution of $\boldsymbol{x}_{t_L}$ (VP $\implies \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, VE $\implies \mathcal{N}(\mathbf{0}, t_L^2 \boldsymbol{I})$.)
  for $\ell = L, L-1, \ldots, 1$ **do**
    Compute

$$w_{t_{\ell-1}} \doteq \alpha_{t_{\ell-1}} - \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \alpha_{t_\ell};$$

$$\hat{\boldsymbol{x}}_{t_{\ell-1}} \doteq \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \hat{\boldsymbol{x}}_{t_\ell} + \left(\bar{\boldsymbol{x}}_\theta(t_\ell, \hat{\boldsymbol{x}}_{t_\ell}) - \frac{\sigma_{t_\ell}^2}{2 \alpha_{t_\ell} \sigma^2} \nabla_{\boldsymbol{\xi}} \left[\| h(\bar{\boldsymbol{x}}_\theta(t_\ell, \boldsymbol{\xi})) - \boldsymbol{\nu} \|_2^2 \right] \Big|_{\boldsymbol{\xi} = \hat{\boldsymbol{x}}_{t_\ell}} \right)$$

  end for
  return $\hat{\boldsymbol{x}}_{t_0}$
end function

Combining this scheme with the basic implementation of unconditional sampling we developed in Section 3.2, we obtain a practical algorithm for conditional sampling of the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$ given measurements following (7.3.9). Algorithm 7.1 records this scheme for the case of Gaussian observation noise with known standard deviation σ, with minor modifications to extend to a general noising process, as in Equation (3.2.85) and the surrounding discussion in Chapter 3 (our discussion above made the simplifying choices $\alpha_t = 1$, $\sigma_t = t$, and $t_\ell = T\ell/L$, as for Equation (3.2.82) in Section 3.2).

Application to medical image reconstruction. The framework that we have developed above is already powerful enough to be applied to solve various *scientific inverse problems*, where the measurements $\boldsymbol{y} = h(\boldsymbol{x}_o) + \boldsymbol{w}$ are the

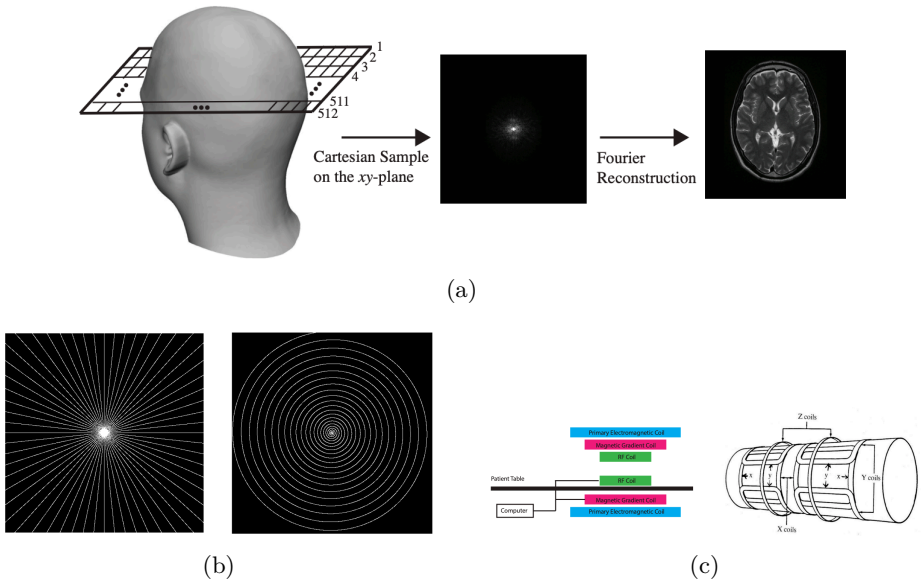


Figure 7.12: Simplified overview of magnetic resonance imaging (MRI) reconstruction. **(a)** An overview of the imaging process. A slice of the object being imaged (say, a human brain) is measured in the Fourier domain via the MRI machine. The spatial image can be reconstructed via direct inversion of the Fourier domain measurements. **(b)** In practice, the MRI machine *subsamples* the Fourier domain via a structured sampling pattern for the sake of efficiency. This creates a linear inverse problem, where priors about the structure of the underlying image must be exploited to enable unique reconstruction. **(c)** The actual measurement apparatus uses a controlled magnetic field to obtain the Fourier domain measurements. Images reproduced with permission from [WM22].

outputs of a known measurement process h on a specific signal of interest $\boldsymbol{x}_o$, and where there are infinitely many possible signals $\boldsymbol{x}$ that satisfy $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$. This means that to find the *true* $\boldsymbol{x}_o$ that generated $\boldsymbol{y}$, we need to exploit prior information about the structure of $\boldsymbol{x}_o$. We will consider the probabilistic setting where $\boldsymbol{x}_o \sim p_{\boldsymbol{x}}$, and where we have access to a prior on $p_{\boldsymbol{x}}$, say in the form of a diffusion model.

An instructive example of a scientific inverse problem is magnetic resonance imaging (MRI) (Figure 7.12), wherein the data distribution $p_{\boldsymbol{x}}$ is over images corresponding to an underlying object of interest (such as a 2D slice of a human subject's brain). In MRI applications, the measurement operator h corresponds to the machine generating and modulating the magnetic field and recording the patient's response to it (Figure 7.12(c)). It can be modeled as implementing the *Fourier transform* of the underlying spatial signal (Figure 7.12(a))—see [WM22, Chapter 10] for a detailed mathematical derivation. In addition, for efficiency's

sake, the MRI machine typically only measures a subset of the measured frequencies, in a structured measurement pattern that can be modeled as either a radial or a spiral pattern (Figure 7.12(b)). Writing $\mathcal{F}$ to denote the (discrete) Fourier transform operator and $\mathcal{S}$ to denote the machine's (known) subsampling pattern, the measured signal is then

$$\boldsymbol{y} = \mathcal{S} \circ \mathcal{F}(\boldsymbol{x}_o) + \boldsymbol{w}. \tag{7.3.31}$$

The measurement operator $\mathcal{S} \circ \mathcal{F}$ is *linear*, implying that it can be represented by a $m \times n$ matrix after appropriate reshaping, but it is also *compressive*, which means that $m \ll n$ and that we need to exploit prior knowledge about $p_{\boldsymbol{x}}$ to reconstruct $\boldsymbol{x}_o$ (or more generally to sample from the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$).

The distribution $p_{\boldsymbol{x}}$ of MRI images is simpler than some image distributions (e.g., natural images), but it is still complex enough to benefit from using a learned model for $p_{\boldsymbol{x}}$ rather than an analytical one (cf. [WM22]). One direct approach is to apply the diffusion posterior sampling approximation we developed in the previous section, which led to Algorithm 7.1, with the specific MRI measurement operator $\boldsymbol{A} = \mathcal{S} \circ \mathcal{F}$ and a diffusion model pretrained on MRI data. This precise combination of approaches has not appeared in the literature. Instead, we highlight results from a different but related approach, which enforces measurement consistency on each diffusion step. Following the pioneering work of Song et al. [SSX+22], numerous such measurement consistency algorithms for medical image reconstruction have been developed, including the improved approaches of Song et al. [SKZ+24] and Rout et al. [RRD+23] (which utilize latent diffusion models, as we have developed for representation autoencoders in Chapter 6). Figure 7.13 shows visual comparisons of reconstructed MRI images across three different samples, comparing FISTA-TV (a classical, accelerated version of the ISTA algorithm we studied in Chapter 2), the measurement consistency diffusion modeling approach of Song et al. [SSX+22], and the ground truth. It can be seen that superior reconstruction quality is achieved by the diffusion-based method, as well as superior faithfulness to fine detail in the ground truth image. Table 7.1 shows MRI reconstruction accuracy results (measured in terms of PSNR) for the approach developed by Song et al. [SSX+22]. It compares favorably to supervised baselines (trained on many independent paired samples $(\boldsymbol{y}, \boldsymbol{x}_o)$, which is an additional requirement) and unsupervised baselines (which correspond to measurement matching approximations that came before diffusion posterior sampling).

7.4 Conditional Inference with Paired Data and Measurements

In many practical applications, we do not know either the distribution of the data $\boldsymbol{x}$ of interest or the explicit relationship between the data and certain observed attributes $\boldsymbol{y}$ of the data. We only have a (large) set of paired samples $(\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ from which we need to infer the data

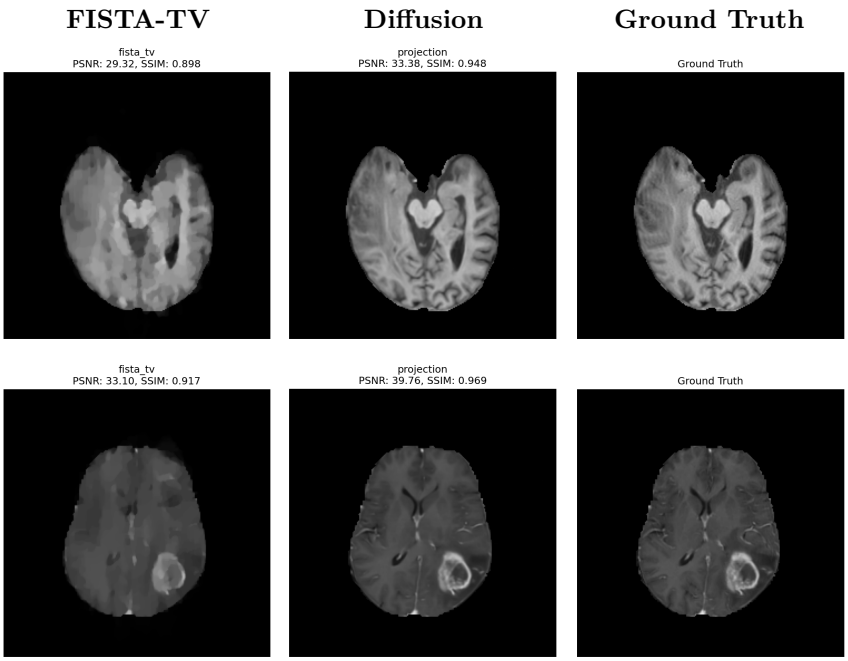


Figure 7.13: **Visual comparison of MRI reconstruction methods.** Three example brain MRI reconstructions showing FISTA-TV, the measurement consistency diffusion approach of Song et al. [SSX+22], and ground truth images on the BraTS MRI reconstruction dataset at 8x undersampling. The diffusion-based approach produces reconstructions that more closely match the ground truth structure compared to FISTA-TV, which can be seen to over-smooth the fine details present in the ground truth image. At the same time, the degree of faithfulness to the ground truth image in the diffusion-based reconstruction (arising from the proper enforcement of measurement consistency) is remarkable.

distribution and a mapping that models their relationship:

$$h : \boldsymbol{x} \mapsto \boldsymbol{y}. \tag{7.4.1}$$

The problem of image classification can be viewed as one such example. In a sense, the classification problem is to learn an (extremely lossy) compressive encoder for natural images. Say, given a random sample of an image $\boldsymbol{x}$, we would like to predict its class label $\boldsymbol{y}$ that best correlates the content in $\boldsymbol{x}$. We know the distribution of natural images of objects is low-dimensional compared to the dimension of the pixel space. From the previous chapters, we have learned that given sufficient samples, in principle, we can learn a structured low-dimensional representation $\boldsymbol{z}$ for $\boldsymbol{x}$ through a learned compressive encoding:

$$f : \boldsymbol{x} \mapsto \boldsymbol{z}. \tag{7.4.2}$$

Method	24× Accel.	8× Accel.	4× Accel.
Cascade DenseNet	$23.39_{\pm 2.17}$	$28.35_{\pm 2.30}$	$30.97_{\pm 2.33}$
DuDoRNet	$18.46_{\pm 3.05}$	$\mathbf{37.88_{\pm 3.03}}$	$30.53_{\pm 4.13}$
Song et al. [SSK+21]	$27.83_{\pm 2.73}$	$35.04_{\pm 2.11}$	$37.55_{\pm 2.08}$
Jalal et al. [JAD+21]	$28.80_{\pm 3.21}$	$36.44_{\pm 2.28}$	$38.76_{\pm 2.32}$
Song et al. [SSX+22]	$\mathbf{29.42_{\pm 3.03}}$	$37.63_{\pm 2.70}$	$\mathbf{39.91_{\pm 2.67}}$

Table 7.1: **MRI reconstruction results** comparing measurement matching approaches across different acceleration factors (the degree of Fourier space subsampling in $\mathcal{S}$) on the BraTS MRI reconstruction dataset. PSNR metrics (higher is better) with standard deviations are reported. Cascade DenseNet [ZFZ19] and DuDoRNet [ZZ20] are supervised baselines trained on paired samples at an 8x acceleration level. The last three methods are diffusion-based approaches using measurement consistency. Reproduced with permission from Song et al. [SSX+22].

The representation $\boldsymbol{z}$ can also be viewed as a learned (lossy but structured) code for $\boldsymbol{x}$. It is rather reasonable to assume that if the class assignment $\boldsymbol{y}$ truly depends on the low-dimensional structures of $\boldsymbol{x}$ and the learned code $\boldsymbol{z}$ truly reflects such structures, $\boldsymbol{y}$ and $\boldsymbol{z}$ can be made highly correlated and hence their joint distribution $p(\boldsymbol{z}, \boldsymbol{y})$ should be extremely low-dimensional. Therefore, we may combine the two desired codes $\boldsymbol{y}$ and $\boldsymbol{z}$ together and try to learn a combined encoder:

$$f : \boldsymbol{x} \mapsto (\boldsymbol{z}, \boldsymbol{y}) \tag{7.4.3}$$

where the joint distribution of $(\boldsymbol{z}, \boldsymbol{y})$ is highly low-dimensional.

From our study in previous chapters, the mapping f is usually learned as a sequence of compression or denoising operators in the same space. Hence to leverage such a family of operations, we may introduce an auxiliary vector $\boldsymbol{w}$ that can be viewed as an initial random guess of the class label $\boldsymbol{y}$. In this way, we can learn a compression or denoising mapping:

$$f : (\boldsymbol{x}, \boldsymbol{w}) \mapsto (\boldsymbol{z}, \boldsymbol{y}) \tag{7.4.4}$$

within a common space. In fact, the common practice of introducing an auxiliary "class token" in the training of a transformer for classification tasks, such as in ViT, can be viewed as learning such a representation by compressing (the coding rate of) given (noisy) samples of $(\boldsymbol{x}, \boldsymbol{w})$. If the distribution of the data $\boldsymbol{x}$ is already a mixture of (low-dimensional) Gaussians, the work [WTL+08] has shown that classification can be done effectively by directly minimizing *the (lossy) coding length* associated with the given samples.

7.4.1 Class-Conditioned Image Generation

While a learned classifier allows us to classify a given image $\boldsymbol{x}$ to its corresponding class, we often would like to generate an image of a given class, by sampling

the learned distribution of natural images. To some extent, this can be viewed as the "inverse" problem to image classification. Let $p_{\boldsymbol{x}}$ denote the distribution of natural images, say modeled by a diffusion-denoising process. Given a class label random variable $y \in [K]$ with realization ν, say an "Apple", we would like to sample the conditional distribution $p_{\boldsymbol{x}|y}(\cdot \mid \nu)$ to generate an image of an apple:

$$\hat{\boldsymbol{x}} \sim p_{\boldsymbol{x}|\boldsymbol{y}}(\cdot \mid \boldsymbol{\nu}). \tag{7.4.5}$$

We call this *class-conditioned image generation*.

In Section 7.3.2, we have seen how to use the denoising-diffusion paradigm for conditional sampling from the posterior $p_{\boldsymbol{x}|\boldsymbol{y}}$ given *model-based* measurements $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$ (Equation (7.3.9)), culminating in the DPS algorithm (Algorithm 7.1). This is a powerful framework, but it does not apply to the class (or text) conditioned image generation problem here, where an explicit generative model h for the observations/attributes y is not available due to the intractability of analytical modeling. In this section, we will present techniques for extending conditional sampling to this setting.

Thus, we now assume only that we have access to samples from the joint distribution of $(\boldsymbol{x}, y)$:

$$(\boldsymbol{x}, y) \sim p_{\boldsymbol{x},y}. \tag{7.4.6}$$

As in the previous section, we define $\boldsymbol{x}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{g}$ with $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ independent of $(\boldsymbol{x}, \boldsymbol{y})$, as in Equation (3.2.85) in Chapter 3, and we will repeatedly use the notation $\boldsymbol{\xi}$ to denote realizations of $\boldsymbol{x}$ and $\boldsymbol{x}_t$.

This section focuses on the conceptual and algorithmic underpinnings for conditional sampling in modern generative models. For a treatment focused on implementation issues in various applications, see Chapter 8, Section 8.7 and subsequent sections.

Classifier Guidance: Consistency via a Pretrained Classifier

Recall from Section 7.3.2 that the following decomposition of the optimal conditional posterior denoiser holds, by virtue of Bayes' rule and conditional independence of $\boldsymbol{y}$ and $\boldsymbol{x}_t$ given $\boldsymbol{x}$ (recall Figure 7.10):

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, y = \nu] = \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] + \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{\xi}} \log p_{y|\boldsymbol{x}_t}(\nu \mid \boldsymbol{\xi}). \tag{7.4.7}$$

In other words, the representation (7.4.7) for the optimal conditional denoiser holds regardless of whether the explicit observation model $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$ is valid. This means that if we can learn such a denoiser (7.4.7) in the paired data setting, we can follow the denoising-diffusion methodology we have developed in Chapter 3 to perform class-conditional sampling.

A natural idea is then to directly implement the likelihood correction term in (7.4.7) using a deep network f_{θ_c} with parameters θ_c, as in Equation (7.4.4):

$$f_{\theta_c} : (t, \boldsymbol{x}_t) \mapsto \operatorname{softmax}(\boldsymbol{W}_{\text{head}} \boldsymbol{z}(t, \boldsymbol{x}_t)). \tag{7.4.8}$$

This expression combines the final representations $\boldsymbol{z}(t, \boldsymbol{x}_t)$ (which also depend on θ_{c}) of the noisy inputs $\boldsymbol{x}_t$ with a classification head $\boldsymbol{W}_{\mathrm{head}} \in \mathbb{R}^{K \times d}$, which maps the representations to a probability distribution over the K possible classes. As is common in practice, it also takes the time t in the noising process as input. Thus, with appropriate training, it provides an approximation to the log-likelihood $\log p_{y|\boldsymbol{x}_t}$, and differentiating $\log f_{\theta_{\mathrm{c}}}$ with respect to its input $\boldsymbol{x}_t$ allows an approximation to the second term in Equation (7.4.7):

$$\bar{\boldsymbol{x}}_\theta^{\mathrm{naive}}(t, \boldsymbol{x}_t, y) = \bar{\boldsymbol{x}}_{\theta_{\mathrm{d}}}(t, \boldsymbol{x}_t) + \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{x}_t} \langle \log f_{\theta_{\mathrm{c}}}(t, \boldsymbol{x}_t), \boldsymbol{e}_y \rangle \tag{7.4.9}$$

where, as usual, we approximate the first term in Equation (7.4.7) via a learned unconditional denoiser for $\boldsymbol{x}_t$ with parameters θ_{d}, and where we write $\boldsymbol{e}_k$ for $k \in [K]$ to denote the k-th canonical basis vector for $\mathbb{R}^K$ (i.e., the vector with a one in the k-th position, and zeros elsewhere). The reader should note that the conditional denoiser $\bar{\boldsymbol{x}}_\theta$ requires two separate training runs, with separate losses: one for the classifier parameters θ_{c}, on a classification loss,[7] and one for the denoiser parameters θ_{d}, on a denoising loss. Such an approach to conditional sampling was already recognized and exploited to perform conditional sampling in pioneering early works on diffusion models, notably those by Sohl-Dickstein et al. [SWM+15] and by Song et al. [SSK+21].

However, this straightforward methodology has two key drawbacks (which is why we label it as "naive"). The first is that, empirically, such a trained deep network classifier frequently does not provide a strong enough guidance signal (in Equation (7.4.7)) to ensure that generated samples reflect the conditioning information y. This was first emphasized by Dhariwal and Nichol [DN21b], who noted that in the setting of class-conditional ImageNet generation, the learned deep network classifier's probability outputs for the class y being conditioned on were frequently around 0.5—large enough to be the dominant class, but not large enough to provide a strong guidance signal—and that upon inspection, generations were not consistent with the conditioning class y. Dhariwal and Nichol [DN21b] proposed to address this heuristically by incorporating an "inverse temperature" hyperparameter $\gamma > 0$ into the definition of the naive conditional denoiser (7.4.9), referring to the resulting conditional denoiser as having incorporated **"classifier guidance"** (CG):

$$\bar{\boldsymbol{x}}_\theta^{\mathrm{CG}}(t, \boldsymbol{x}_t, y) = \bar{\boldsymbol{x}}_{\theta_{\mathrm{d}}}(t, \boldsymbol{x}_t) + \gamma \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{x}_t} \langle \log f_{\theta_{\mathrm{c}}}(t, \boldsymbol{x}_t), \boldsymbol{e}_y \rangle \tag{7.4.10}$$

with the case $\gamma = 1$ coinciding with (7.4.9).

Dhariwal and Nichol [DN21b] found that a setting $\gamma > 1$ performed best empirically. One possible interpretation for this is as follows: note that, in the context of the *true* likelihood term Equation (7.4.7), scaling by γ gives equivalently

$$\gamma \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi}) = \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{\xi}} \log \left(p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi})^\gamma \right), \tag{7.4.11}$$

[7] In Chapter 8, we review the process of training such a classifier in full detail.

which suggests the natural interpretation of the parameter γ performing (inverse) *temperature scaling* on the likelihood $p_{\boldsymbol{y}|\boldsymbol{x}_t}$, which is precise if we consider the renormalized distribution $p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu} \mid \boldsymbol{\xi})^\gamma / \int p_{\boldsymbol{y}|\boldsymbol{x}_t}(\boldsymbol{\nu}' \mid \boldsymbol{\xi})^\gamma \mathrm{d}\boldsymbol{\nu}'$. However, note that this *is not* a rigorous interpretation in the context of Equation (7.4.7), because the gradients are taken with respect to $\boldsymbol{\xi}$, and the normalization constant in the temperature-scaled distribution is in general a function of $\boldsymbol{\xi}$. Instead, the parameter γ should simply be understood as amplifying large values of the deep network classifier's output probabilities $f_{\theta_\mathrm{c}}(t, \boldsymbol{x}_t)$ *relative to* smaller ones, which effectively amplifies the guidance signal provided in cases where the deep network f assigns it the largest probability among the K classes.

Simplifications and Improvements via Classifier-Free Guidance

Nevertheless, classifier guidance does not address the second key drawback of the naive methodology: it is both cumbersome and wasteful to have to train an auxiliary classifier f_{θ_c} in addition to the unconditional denoiser $\bar{\boldsymbol{x}}_{\theta_\mathrm{d}}$, given that it is not possible to directly adapt a pretrained classifier due to the need for it to work well on noisy inputs $\boldsymbol{x}_t$ and incorporate other empirically-motivated architecture modifications. In particular, Dhariwal and Nichol [DN21b] found that it was necessary to *explicitly design the architecture of the deep network implementing the classifier to match that of the denoiser*.

Moreover, from a purely practical perspective—trying to obtain the best possible performance from the resulting sampler—the best-performing configuration of classifier guidance-based sampling departs even further from the idealized and conceptually sound framework we have presented above. To obtain the best performance, Dhariwal and Nichol [DN21b] found it necessary to provide the class label y as an additional input to the denoiser $\bar{\boldsymbol{x}}_{\theta_\mathrm{d}}$. As a result, the idealized classifier-guided denoiser (7.4.10), derived by Dhariwal and Nichol [DN21b] as we have done above from the conditional posterior denoiser decomposition (7.4.7), is not exactly reflective of the best-performing denoiser in practice—such a denoiser actually combines a *conditional* denoiser for $\boldsymbol{x}_t$ given y with an additional guidance signal from an auxiliary classifier!

This state of affairs, empirically motivated as it is, led Ho and Salimans [HS22a] in subsequent work to propose a more empirically pragmatic methodology, known as **classifier-free guidance (CFG)**. Instead of representing the conditional denoiser (7.4.7) as a weighted sum of an unconditional denoiser for $\boldsymbol{x}_t$ with a log-likelihood correction term (with possibly modified weights, as in classifier guidance), they accept the apparent necessity of training a conditional denoiser for $\boldsymbol{x}_t$ given y, as demonstrated by the experimental results of Dhariwal and Nichol [DN21b], and replace the log-likelihood gradient term with a correctly-weighted sum of this conditional denoiser with an *unconditional* denoiser for $\boldsymbol{x}$ given $\boldsymbol{x}_t$.[8]

[8] That said, Ho and Salimans [HS22a] actually proposed to use a different weighting than what we present here, based on the fact that Dhariwal and Nichol [DN21b] heuristically replaced the unconditional denoiser in (7.4.7) with a conditional denoiser. In fact, the weighting we derive and present here reflects modern practice, and in particular is used in state-of-the-art

Deriving Classifier-Free Guidance. To see how this structure arises, we begin with an 'idealized' version of the classifier guidance denoiser $\bar{\boldsymbol{x}}_\theta^{\mathrm{CG}}$ defined in (7.4.10), for which the denoiser $\bar{\boldsymbol{x}}_{\theta_{\mathrm{d}}}$ and the classifier $f_{\theta_{\mathrm{c}}}$ perfectly approximate their targets, via (7.4.7):

$$\bar{\boldsymbol{x}}_\theta^{\mathrm{CG,\,ideal}}(t, \boldsymbol{\xi}, \nu) = \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] + \gamma \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{\xi}} \log p_{y|\boldsymbol{x}_t}(\nu \mid \boldsymbol{\xi}). \tag{7.4.12}$$

We then use Bayes' rule, in the form

$$\log p_{y|\boldsymbol{x}_t} = \log p_{\boldsymbol{x}_t|y} + \log p_y - \log p_{\boldsymbol{x}_t}, \tag{7.4.13}$$

together with Tweedie's formula (Theorem 3.2, modified as in Equation (3.2.86)) to convert between score functions and denoisers, to obtain

$$\begin{aligned} \bar{\boldsymbol{x}}_\theta^{\mathrm{CG,\,ideal}}(t, \boldsymbol{\xi}, \nu) &= \frac{1}{\alpha_t}\boldsymbol{\xi} + (1-\gamma)\frac{\sigma_t^2}{\alpha_t}\nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{x}_t}(\boldsymbol{\xi}) + \gamma \frac{\sigma_t^2}{\alpha_t} \nabla_{\boldsymbol{\xi}} \log p_{\boldsymbol{x}_t|y}(\boldsymbol{\xi} \mid \nu) \\ &= (1-\gamma)\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] + \gamma \mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, y = \nu], \end{aligned} \tag{7.4.14}$$

where in the last line, we apply Equation (7.3.11). Now, Equation (7.4.14) suggests a natural approximation strategy: we combine a learned unconditional denoiser for $\boldsymbol{x}$ given $\boldsymbol{x}_t$, as previously, with a learned *conditional* denoiser for $\boldsymbol{x}$ given $\boldsymbol{x}_t$ and y.

However, following Ho and Salimans [HS22a] and the common practice of training deep network denoisers, it is standard to use the *same* deep network to represent both the conditional and unconditional denoisers by introducing an additional label, which we will denote by $\varnothing$, to denote the "unconditional" case. This leads to the form of the CFG denoiser:

$$\bar{\boldsymbol{x}}_\theta^{\mathrm{CFG}}(t, \boldsymbol{x}_t, y) = (1-\gamma)\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{x}_t, \varnothing) + \gamma \bar{\boldsymbol{x}}_\theta(t, \boldsymbol{x}_t, y). \tag{7.4.15}$$

To train a denoiser $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{x}_t, y^+)$ for use with classifier-free guidance sampling, where $y^+ \in \{1, \ldots, K, \varnothing\}$, we proceed almost identically to the unconditional training procedure in Algorithm 3.2, but with two modifications:

1. When we sample from the dataset, we sample a pair $(\boldsymbol{x}, y)$ rather than just a sample $\boldsymbol{x}$.
2. Every time we sample a pair from the dataset, we sample the augmented label y^+ via
$$y^+ = \begin{cases} \varnothing & \text{with probability } p_{\mathrm{uncond}}; \\ y & \text{else.} \end{cases} \tag{7.4.16}$$
Here, $p_{\mathrm{uncond}} \in [0, 1]$ is a new hyperparameter. This can be viewed as a form of dropout [SHK+14].

In this way, we train a conditional denoiser suitable for use in classifier-free guidance sampling. We summarize the overall sampling process for class-conditioned sampling with classifier-free guidance in Algorithm 7.2.

diffusion models such as Stable Diffusion 3.5 [EKB+24].

Algorithm 7.2 Conditional sampling with classification data, using class-conditioned denoiser.

Input: An ordered list of timesteps $0 \leq t_0 < \cdots < t_L \leq T$ to use for sampling.
Input: Class label $\nu \in \{1, \ldots, K\}$ to condition on.
Input: A denoiser $\bar{\boldsymbol{x}}_\theta \colon \{t_\ell\}_{\ell=1}^L \times \mathbb{R}^D \times \{1, \ldots, K, \varnothing\} \to \mathbb{R}^D$ for $p_{\boldsymbol{x}|y}$ and $p_{\boldsymbol{x}}$ (input $\varnothing$ for $p_{\boldsymbol{x}}$).
Input: Scale and noise level functions $\alpha, \sigma \colon \{t_\ell\}_{\ell=0}^L \to \mathbb{R}_{\geq 0}$.
Input: Guidance strength $\gamma \geq 0$ ($\gamma > 1$ preferred for performance).
Output: A sample $\hat{\boldsymbol{x}}$, approximately from $p_{\boldsymbol{x}|y}(\,\cdot \mid \nu)$.

function DDIMSamplerConditionalCFG($\bar{\boldsymbol{x}}_\theta, \nu, \gamma, (t_\ell)_{\ell=0}^L$)
  Initialize $\hat{\boldsymbol{x}}_{t_L} \sim$ approximate distribution of $\boldsymbol{x}_{t_L}$ (VP $\implies \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, VE $\implies \mathcal{N}(\mathbf{0}, t_L^2 \boldsymbol{I})$).
  for $\ell = L, L-1, \ldots, 1$ **do**
    Compute

$$\hat{\boldsymbol{x}}_{t_{\ell-1}} \doteq \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \hat{\boldsymbol{x}}_{t_\ell} + \left(\alpha_{t_{\ell-1}} - \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \alpha_{t_\ell} \right) \left((1-\gamma) \bar{\boldsymbol{x}}_\theta(t_\ell, \hat{\boldsymbol{x}}_{t_\ell}, \varnothing) + \gamma \bar{\boldsymbol{x}}_\theta(t_\ell, \hat{\boldsymbol{x}}_{t_\ell}, \nu) \right)$$

  end for
  return $\hat{\boldsymbol{x}}_{t_0}$
end function

Ho and Salimans [HS22a] reports strong empirical performance for class-conditional image generation with classifier-free guidance, and it has become a mainstay of the largest-scale practical diffusion models, such as Stable Diffusion [RBL+22] and its derivatives.

What Denoisers Does Classifier-Free Guidance Promote Learning? As we have seen, the derivation of classifier-free guidance is rather opaque and empirically motivated, giving little insight into the mechanisms behind its strong performance. A number of theoretical works have attempted to address this [BN24b; LWQ25; WCL+24]. They provide explanations for some parts of the overall CFG methodology—itself encompassing denoiser parameterization and training, as well as configuration of the guidance strength and performance at sampling time. Below, following our running theme in the book, we will give an interpretation in the simplifying setting of a Gaussian mixture model data distribution and denoiser. First, we consider *the effect of CFG*: what does a large weight $\gamma \gg 1$ promote?

Example 7.5. Let us recall the low-rank mixture of Gaussians data generating process we studied in Example 3.3 (and specifically, the form in Equation (3.2.69)). Given $K \in \mathbb{N}$ classes, we assume that

$$\boldsymbol{x} \sim \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{0}, \boldsymbol{U}_k \boldsymbol{U}_k^\top), \tag{7.4.17}$$

where each $\boldsymbol{U}_k \in \mathsf{O}(D, P) \subseteq \mathbb{R}^{D \times P}$ is a matrix with orthogonal columns, and $P \ll D$. Moreover, we assume that the class label $y \in [K]$ is a deterministic function of $\boldsymbol{x}$ mapping an example to its corresponding mixture component. We consider the simple noise process (3.2.1) studied in Chapter 3, so that $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}$ for all $t \in [0, T]$, with $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Applying the analysis in Example 3.3 (and the subsequent analysis of the low-rank case, culminating in Equation (3.2.80)), we obtain for the class-conditional optimal denoisers

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}, y = \nu] = \frac{1}{1+t^2} \boldsymbol{U}_\nu \boldsymbol{U}_\nu^\top \boldsymbol{\xi} \tag{7.4.18}$$

for each $\nu \in [K]$, and for the optimal unconditional denoiser, we obtain

$$\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}] = \frac{1}{1+t^2} \sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_k^\top \boldsymbol{\xi}\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_i^\top \boldsymbol{\xi}\|_2^2\right)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{\xi}. \tag{7.4.19}$$

As a result, we can express the CFG denoiser with guidance strength $\gamma > 1$ as

$$\begin{aligned} \bar{\boldsymbol{x}}^{\text{CFG, ideal}}(t, \boldsymbol{x}_t, y) = \frac{1}{1+t^2} \Bigg(& (1-\gamma) \sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_k^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \\ & + \gamma \boldsymbol{U}_y \boldsymbol{U}_y^\top \Bigg) \boldsymbol{x}_t. \end{aligned} \tag{7.4.20}$$

This denoiser has a simple, interpretable form:

1. The first term, corresponding to *the unconditional denoiser*, performs denoising of the signal $\boldsymbol{x}_t$ against an average of the denoisers associated with each subspace, weighted by how correlated $\boldsymbol{x}_t$ is with each subspace.

2. The second term, corresponding to *the conditional denoiser*, simply performs denoising with the conditioning class's denoiser.

The CFG scheme averages these two denoisers. The effect of this averaging can be gleaned from the refactoring

$$\begin{aligned} \bar{\boldsymbol{x}}^{\text{CFG, ideal}}(t, \boldsymbol{x}_t, y) = \frac{1}{1+t^2} \Bigg(& \left[\gamma + (1-\gamma) \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_y^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} \right] \boldsymbol{U}_y \boldsymbol{U}_y^\top \\ & + (1-\gamma) \sum_{k \neq y} \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_k^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \Bigg) \boldsymbol{x}_t, \end{aligned} \tag{7.4.21}$$

where we combined the conditional denoiser with the corresponding $k = y$ term in the sum over $k \in [K]$. We have

$$\sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_k^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} = 1, \tag{7.4.22}$$

and each summand is nonnegative, hence also bounded above by 1. So we can conclude two regimes for the terms in Equation (7.4.21):

1. **Well-correlated regime:** In this case, suppose the noisy input $\boldsymbol{x}_t$ correlates well with one of the subspaces $\boldsymbol{U}_y$. If $\boldsymbol{x}_t$ correlates well with $\boldsymbol{U}_y$, then the normalized weight corresponding to the $k = y$ summand in the unconditional denoiser is near to 1. Then

$$\gamma + (1-\gamma)\frac{\exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_y^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} \approx 1, \tag{7.4.23}$$

   all other weights are necessarily near to zero, and the CFG denoiser is approximately equal to the denoiser associated to the conditioning class y.

   This implies: *if the CFG denoiser's input is highly correlated with one of the class subspaces, the denoiser is approximately equal to the corresponding class-conditional denoiser*! Hence, we expect the CFG denoising sampler to converge.

2. **Poorly-correlated regime:** In contrast, if $\boldsymbol{x}_t$ *does not* correlate well with $\boldsymbol{U}_y$ (say because t is large), then the normalized weight corresponding to the $k = y$ summand in the unconditional denoiser is near to 0. As a result,

$$\gamma + (1-\gamma)\frac{\exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_y^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)} \approx \gamma, \tag{7.4.24}$$

   and thus the guidance strength $\gamma \gg 1$ places a large positive weight on the denoiser associated to y.

   Meanwhile, in the second term of Equation (7.4.21), any classes $k \neq y$ that are well-correlated with $\boldsymbol{x}_t$ receive a large *negative* weight from the $1 - \gamma$ coefficient. This simultaneously has the effect of making the denoised signal vastly more correlated with the conditioning class y, and making it negatively correlated with the previous iterate (i.e., the iterate before denoising).

   In other words, *CFG steers the iterative denoising process towards the conditioning class and away from the previous iterate*, a different dynamics from purely conditional sampling (i.e., the case $\gamma = 1$).

■

Example 7.5 shows that in the mixture of Gaussians setting, the CFG denoiser provides a strong bias towards the conditioning class, while preserving local correctness (hence convergence of sampling) when the noise level t is small. This is suggestive of why a large weight $\gamma \gg 1$ has been found essential in practice.

Next, the following example will demonstrate an insight into the *parameterization* of the denoiser in the presence of low-dimensional structure, again in the Gaussian mixture model.

Example 7.6. Consider again the mixture of Gaussians model studied in Example 7.5:

$$\boldsymbol{x} \sim \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(\boldsymbol{0}, \boldsymbol{U}_k \boldsymbol{U}_k^\top), \tag{7.4.25}$$

where each $\boldsymbol{U}_k \in \mathsf{O}(D, P) \subseteq \mathbb{R}^{D\times P}$ is a matrix with orthogonal columns, and $P \ll D$. The class label $y \in [K]$ is a deterministic function of $\boldsymbol{x}$ mapping an example to its corresponding mixture component. For this distribution, and again under the simple noise process (3.2.1) studied in Chapter 3, where $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}$ for all $t \in [0, T]$, with $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, we saw in Example 7.5 that the ideal CFG denoiser takes the form

$$\bar{\boldsymbol{x}}^{\mathrm{CFG,\,ideal}}(t, \boldsymbol{x}_t, y) = \frac{1}{1+t^2}\left((1-\gamma)\sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_k^\top \boldsymbol{x}_t\|_2^2\right)}{\sum_{i=1}^{K}\exp\left(\frac{1}{2t^2(1+t^2)}\|\boldsymbol{U}_i^\top \boldsymbol{x}_t\|_2^2\right)}\boldsymbol{U}_k\boldsymbol{U}_k^\top + \gamma \boldsymbol{U}_y\boldsymbol{U}_y^\top\right)\boldsymbol{x}_t. \tag{7.4.26}$$

Now we consider the problem of parameterizing a learnable denoiser $\boldsymbol{x}_\theta^{\mathrm{CFG}}$ to represent the optimal denoiser (7.4.26). For tractability, we add an additional assumption associated to the subspaces $\boldsymbol{U}_k$ being 'distinguishable' from one another, which is natural in practice: specifically, we assume that for any pair of indices $k, k' \in [K]$ with $k \neq k'$, we can find a set of K nonzero directions $\boldsymbol{v}_k \in \mathbb{R}^D$ such that

$$\boldsymbol{U}_k\boldsymbol{U}_k^\top \boldsymbol{v}_k = \boldsymbol{v}_k, \quad \boldsymbol{U}_{k'}\boldsymbol{U}_{k'}^\top \boldsymbol{v}_k = \boldsymbol{0}, \;\; k' \neq k. \tag{7.4.27}$$

This is a slightly stronger assumption than simple distinguishability, but it should be noted that it is not overly restrictive: for example, it still allows the subspaces $\boldsymbol{U}_k$ to have significant correlations with one another.[9]

These vectors $\boldsymbol{v}_k$ can then be thought of as *embeddings* of the class label $y \in [K]$, and we can use them to define a more general operator that can represent both the unconditional and class-conditional denoisers. More precisely, consider

[9] More generally, this assumption is naturally formulated as an *incoherence condition* between the subspaces $\boldsymbol{U}_{[K]}$, familiar from the theory of compressive sensing [WM22].

the mapping

$$(\boldsymbol{x}_t, \boldsymbol{v}) \mapsto \sum_{k=1}^{K} \frac{\exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{v}|\Big)}{\sum_{i=1}^{K} \exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{U}_i \boldsymbol{U}_i^\top \boldsymbol{v}|\Big)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t. \tag{7.4.28}$$

Then if we plug in $\boldsymbol{v} = \boldsymbol{x}_t$ to the operator (7.4.28), we obtain a result proportional to the first term in (7.4.19), corresponding to the optimal unconditional denoiser for $\boldsymbol{x}_t$. On the other hand, if we substitute $\boldsymbol{v} = \boldsymbol{v}_y$ for some $y \in [K]$, we get by construction of the embedding vectors $\boldsymbol{v}_k$

$$\begin{aligned} &\sum_{k=1}^{K} \frac{\exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{v}_y|\Big)}{\sum_{i=1}^{K} \exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{U}_i \boldsymbol{U}_i^\top \boldsymbol{v}_y|\Big)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t \\ = \quad & \frac{\exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{v}_y|\Big)}{\exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{v}_y|\Big) + K - 1} \boldsymbol{U}_y \boldsymbol{U}_y^\top \boldsymbol{x}_t \qquad (7.4.29) \\ & + \sum_{k \neq y} \frac{1}{\exp\Big(\frac{1}{2t^2(1+t^2)}|\boldsymbol{x}_t^\top \boldsymbol{v}_y|\Big) + K - 1} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t. \qquad (7.4.30) \end{aligned}$$

We aim to simplify the preceding expression further. To this end, recall that when using this denoiser for sampling, we have $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}$, with y denoting the label of $\boldsymbol{x}$. In other words, conditioned on y, we have that $\boldsymbol{x}_t = \boldsymbol{U}_y \boldsymbol{z} + t\boldsymbol{g}$, where $\boldsymbol{z}$ is an independent P-dimensional Gaussian noise vector with zero mean and identity covariance. We have

$$\begin{aligned} \left|\boldsymbol{x}_t^\top \boldsymbol{v}_y\right| &= |\langle \boldsymbol{U}_y \boldsymbol{z}, \boldsymbol{v}_y \rangle + t\langle \boldsymbol{g}, \boldsymbol{v}_y \rangle| \\ &= \left|\langle \boldsymbol{z}, \boldsymbol{U}_y^\top \boldsymbol{v}_y \rangle + t\langle \boldsymbol{g}, \boldsymbol{v}_y \rangle\right|. \end{aligned}$$

Because $\boldsymbol{z}$ and $\boldsymbol{g}$ are independent Gaussian random variables with identity covariance matrices, it follows from *rotational invariance* of the Gaussian distribution (see [Ver18]) that

$$\begin{aligned} \left|\boldsymbol{x}_t^\top \boldsymbol{v}_y\right| &\overset{\mathrm{d}}{=} \left|\|\boldsymbol{U}_y^\top \boldsymbol{v}_y\|_2 z + t\|\boldsymbol{v}_y\|_2 g\right| \\ &= \left|\|\boldsymbol{v}_y\|_2 z + t\|\boldsymbol{v}_y\|_2 g\right|, \end{aligned}$$

where z and g are independent *scalar* $\mathcal{N}(0,1)$ random variables, and $\overset{\mathrm{d}}{=}$ denotes equality in distribution (the random variables have the same distribution). The second line above follows from the definition of $\boldsymbol{v}_y$. Now, by independence, the random variable $z + tg \sim \mathcal{N}(0, (1+t^2))$. Then using the well-known result that $\mathbb{E}[|g|] = \sqrt{2/\pi}$ if $g \sim \mathcal{N}(0,1)$, we conclude

$$\mathbb{E}\left[\left|\boldsymbol{x}_t^\top \boldsymbol{v}_y\right|\right] = \sqrt{(2/\pi)(1+t^2)}\|\boldsymbol{v}_y\|_2.$$

In practice, the quantity $\left|\boldsymbol{x}_t^\top \boldsymbol{v}_y\right|$ is close to its expectation *with overwhelming probability*, by Gaussian concentration (see, again, [Ver18]). On the other hand, we have by similar reasoning

$$\mathbb{E}\left[\|\boldsymbol{x}_t\|_2^2\right] = P + t^2 D,$$

and properties of the Gaussian distribution again imply that $\|\boldsymbol{x}_t\|_2$ will concentrate around $\sqrt{P + t^2 D}$ with overwhelming probability. It is then sensible to renormalize the embeddings $\boldsymbol{v}_y$ so that their dot-product with $\boldsymbol{x}_t$ has the same scale: defining

$$\boldsymbol{v}_{t,y} = \frac{(P + t^2 D)}{\sqrt{(2/\pi)(1 + t^2)}} \frac{\boldsymbol{v}_y}{\|\boldsymbol{v}_y\|_2}$$

for the normalized embeddings, our previous argument implies

$$\left|\boldsymbol{x}_t^\top \boldsymbol{v}_{t,y}\right| \approx (P + t^2 D)$$

with overwhelming probability. Returning our attention to our starting point, i.e. (7.4.28) evaluated at $(\boldsymbol{x}_t, \boldsymbol{v}_y)$ and our subsequent simplification, we have for the "softmax" argument

$$\frac{1}{2t^2(1 + t^2)} |\boldsymbol{x}_t^\top \boldsymbol{v}_y| \approx \frac{P + t^2 D}{2t^2(1 + t^2)}$$

with overwhelming probability with respect to $\boldsymbol{x}_t$. Hence if t is bounded and the subspace dimension P is sufficiently large,[10] the following approximation holds with overwhelming probability:

$$\sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \boldsymbol{x}_t^\top \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{v}_y\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \boldsymbol{x}_t^\top \boldsymbol{U}_i \boldsymbol{U}_i^\top \boldsymbol{v}_y\right)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t \approx \boldsymbol{U}_y \boldsymbol{U}_y^\top \boldsymbol{x}_t. \tag{7.4.31}$$

In words, **evaluating the operator (7.4.28) at the embedding for a class $\boldsymbol{v}_{t,y}$ yields the class-conditional denoiser for y!**

Thus, the family of operators (7.4.28) provides a unified way to parameterize the constituent operators in the optimal denoiser for $\boldsymbol{x}$ *within a single 'network'*. More precisely, it is enough to add the output of an instantiation of (7.4.28) with input $(\boldsymbol{x}_t, \boldsymbol{x}_t)$ to an instantiation with input $(\boldsymbol{x}_t, \boldsymbol{v}_{t,y})$. The resulting operator is a function of $(t, \boldsymbol{x}_t, y)$, and computationally, the subspaces $(\boldsymbol{U}_k)_{k=1}^K$ and embeddings $y \mapsto \boldsymbol{v}_{t,y}$ become its learnable parameters. ■

[10] For example, it suffices to consider a large-scale, asymptotic regime where $P, D \to \infty$ with their ratio P/D converging to a fixed constant. In such a scenario, the *aspect ratio* of the subspaces is fixed while their sizes go to infinity, which can act as a model for some discretization phenomena (e.g., a fixed scene sampled as an image with the number of pixels going to infinity).

Example 7.6 shows that in the special case of a low-rank mixture of Gaussians data distribution for $\boldsymbol{x}$ with incoherent components, operators of the form

$$(\boldsymbol{x}_t, \boldsymbol{v}) \mapsto \sum_{k=1}^{K} \frac{\exp\left(\frac{1}{2t^2(1+t^2)} \boldsymbol{x}_t^\top \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{v}\right)}{\sum_{i=1}^{K} \exp\left(\frac{1}{2t^2(1+t^2)} \boldsymbol{x}_t^\top \boldsymbol{U}_i \boldsymbol{U}_i^\top \boldsymbol{v}\right)} \boldsymbol{U}_k \boldsymbol{U}_k^\top \boldsymbol{x}_t \tag{7.4.32}$$

provide a sufficiently rich class of operators to parameterize the ideal denoiser for noisy observations $\boldsymbol{x}_t$ of $\boldsymbol{x}$ when using classifier-free guidance. For such operators, the auxiliary input $\boldsymbol{v}$ can be taken as either $\boldsymbol{x}_t$ or a suitable embedding of the class label $y \mapsto \boldsymbol{v}_y$ in order to realize such a denoiser—as the example shows, the ideal denoiser is a weighted sum of such operators, with weights derived from the guidance strength $\gamma > 1$.

Based on the framework in Chapter 5, which develops deep network architectures suitable for transforming more general data distributions to structured representations using the low-rank mixture of Gaussians model as a primitive, it is natural to imagine that operators of the type (7.4.32) may be leveraged in denoisers for general data distributions $\boldsymbol{x}$ with low-dimensional geometrically-structured components that are sufficiently distinguishable (say, incoherent) from one another. The next section demonstrates that this is indeed the case.

7.4.2 Caption-Conditioned Image Generation

In Section 7.4.1 we have seen, conceptually, how to set up denoisers for use with *classifier-free guidance*, the dominant practical methodology for performing conditional sampling via the diffusion-denoising paradigm. In short, such denoisers take as input the noisy data $\boldsymbol{x}_t$ as well as either the conditioning signal $\boldsymbol{y}$ or a "null input" $\varnothing$, which determines whether the denoiser should perform *unconditional denoising* or *conditional denoising* (conditioned on $\boldsymbol{y}$).

Given this design, the most important practical question is *how to realize this family of operators as a neural network?* An empirically-successful neural network layer design known as "cross attention", which we will present in due course, provides a practical answer to this question. However, even before this, there are several more fundamental issues to sort out:

1. *Input modality mismatch*: In the most interesting practical applications, the data $\boldsymbol{x}$ and the conditioning signal $\boldsymbol{y}$ *do not lie in directly-comparable spaces*, although they share many underlying structural relationships. For example, we saw this in the case of a class label y in Section 7.4.1. More generally, one could consider a natural language description $\boldsymbol{y}$ of an image $\boldsymbol{x}$ (referred to as a "caption"), where $\boldsymbol{y}$ would correspond to a sequence of characters in different alphabets.

2. *Parameterization and flexibility*: Downstream of the input modality issue, there is a fundamental question of whether the same basic architecture for the network can be used for different modalities, or whether this needs to be tuned in a very precise way per-modality.

3. *Training*: This entails the proper way to use our paired samples $(\boldsymbol{x}, \boldsymbol{y})$ to train and deploy the conditional denoisers. The design space is vast: one could imagine applying various modality-specific encoder-decoder architectures, as we have developed in Chapters 4 and 6, together with different denoising backbones in various combinations. The key questions here are of *expressivity*, getting the best conditional denoising performance; *efficiency*, getting the best performance per unit of training data; and *stability*, guaranteeing that the training process converges quickly without collapse.

We will cover these three issues in detail below, providing guidance on how to explore the design space and, where possible, generally-useful prescriptions.

Modality-Agnostic Input Embedding

Although we do not know the precise relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ in the general paired data setting, in all practical settings of interest, they have significant correlations. For example, when the conditioning signal $\boldsymbol{y}$ is a "caption" describing the image $\boldsymbol{x}$ in natural language, it is easy to imagine a human artist being able to produce an accurate sketch of $\boldsymbol{x}$ based solely on a high-quality caption. In our probabilistic setting, one natural way to quantify these "correlations" is in terms of the mutual information, which we saw previously in Chapter 4 (Equation (4.1.12)):

$$I(\boldsymbol{x}; \boldsymbol{y}) = h(\boldsymbol{x}) - h(\boldsymbol{x} \mid \boldsymbol{y}). \tag{7.4.33}$$

The mutual information (7.4.33) is a purely information-theoretic quantity that does not depend on modality-specific aspects of $\boldsymbol{x}$ and $\boldsymbol{y}$. Hence it offers a natural criterion for learning a useful representation of either input $\boldsymbol{x}$, $\boldsymbol{y}$: one simply seeks to learn, say for $\boldsymbol{x}$, a representation $\boldsymbol{z} = f(\boldsymbol{x})$ that preserves the mutual information, so that $I(\boldsymbol{z}; \boldsymbol{y}) \approx I(\boldsymbol{x}; \boldsymbol{y})$. That is:

$$\text{Learn } \boldsymbol{z} = f(\boldsymbol{x}) \text{ such that } I(\boldsymbol{z}; \boldsymbol{y}) \approx I(\boldsymbol{x}; \boldsymbol{y}). \tag{7.4.34}$$

Actually, we can be even more prescriptive here. For an encoder f which is deterministic (matching all the cases we have studied throughout Chapters 4 and 6), a fundamental result in information theory known as the *data processing inequality* tells us that processing $\boldsymbol{x}$ with f to produce $\boldsymbol{z} = f(\boldsymbol{x})$ changes the mutual information in a predictable way: *it can only reduce it.*

Theorem 7.1 (Data Processing Inequality)**.** *Consider random variables $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}$ defined such that $\boldsymbol{z}$ is independent of $\boldsymbol{y}$, conditioned on $\boldsymbol{x}$.*[11] *Then the data processing inequality holds:*

$$I(\boldsymbol{x}; \boldsymbol{y}) \geq I(\boldsymbol{z}; \boldsymbol{y}). \tag{7.4.35}$$

Proof. Consider the mutual information $I(\boldsymbol{y}; \boldsymbol{x}, \boldsymbol{z})$ between $\boldsymbol{y}$ and $(\boldsymbol{x}, \boldsymbol{z})$. Because $\boldsymbol{z}$ is independent of $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$, we have

$$h(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) = h(\boldsymbol{y} \mid \boldsymbol{x}),$$

[11] In other words, the random variables form a Markov chain $\boldsymbol{y} \to \boldsymbol{x} \to \boldsymbol{z}$.

so $I(\boldsymbol{y}; \boldsymbol{x}, \boldsymbol{z}) = I(\boldsymbol{y}; \boldsymbol{x})$. Meanwhile, we have by Bayes' rule that

$$p_{\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z}} = p_{\boldsymbol{y}|\boldsymbol{z}} \frac{p_{\boldsymbol{y},\boldsymbol{x},\boldsymbol{z}}}{p_{\boldsymbol{y},\boldsymbol{z}} p_{\boldsymbol{x}|\boldsymbol{z}}},$$

which implies by the definition of conditional differential entropy that

$$I(\boldsymbol{y}; \boldsymbol{x}, \boldsymbol{z}) = I(\boldsymbol{y}; \boldsymbol{z}) - \mathbb{E}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}\left[\log\left(\frac{p_{\boldsymbol{x}|\boldsymbol{z}} p_{\boldsymbol{y},\boldsymbol{z}}}{p_{\boldsymbol{y},\boldsymbol{x},\boldsymbol{z}}}\right)\right].$$

Applying Jensen's inequality to the expectation in the previous line then implies the claim, because log is a concave function (and the mutual information is symmetric, i.e. $I(\boldsymbol{x}; \boldsymbol{y}) = I(\boldsymbol{y}; \boldsymbol{x})$). □

For a deterministic encoder f, $\boldsymbol{z} = f(\boldsymbol{x})$ is always independent of $\boldsymbol{y}$ when conditioned on $\boldsymbol{x}$. The data processing inequality therefore suggests the following computational procedure to find the encoder f: *seek to maximize the mutual information between the representation and the conditioning signal.* Equation (7.4.34) then leads to the objective

$$\max_f I(f(\boldsymbol{x}); \boldsymbol{y}). \tag{7.4.36}$$

This is known as the *Infomax principle* [Lin88], and it dates to the earliest years of representation learning.

Computational considerations in applying the Infomax principle. There are two key conceptual issues to keep in mind when thinking about how to apply the (abstract) Infomax principle (7.4.36) for learning a representation. The first is that in general it is computationally hard to estimate the mutual information, which constitutes the objective in (7.4.36). As we discussed in the context of rate distortion and lossy coding when we introduced the mutual information in Chapter 4, the mutual information can be relaxed to remain well-defined even when $\boldsymbol{x}$ and $\boldsymbol{y}$ are low-dimensional and have differential entropy approaching $-\infty$ (e.g., by connection to the Gaussian rate reduction function that we have studied in Chapters 4 to 6). Moreover, empirically such relaxations have non-worst-case statistical properties in practice, as we saw in the experiments of Chapter 4; and the accuracy of these approximations can be improved by incremental processing, as we saw in Chapter 5. This makes a criterion like Equation (7.4.36) an excellent foundation for learning representations of paired data $(\boldsymbol{x}, \boldsymbol{y})$.

The second issue to keep in mind is that for the objective (7.4.36) to represent a sensible criterion for learning f, it is necessary that f itself is parameterized in a specific way, and/or that some additional structural enforcement is done on the representations $f(\boldsymbol{x})$. Indeed, if f is completely unconstrained, there are simple trivial solutions to (7.4.36) that do not correspond to useful representations: for example, just take $f(\boldsymbol{x}) = \boldsymbol{x}$ to be the identity! We discussed principled techniques for structuring the representation space in Chapter 6 when we discussed

autoencoding and closed loop transcription; many of these techniques can be brought to bear in our present joint embedding setting, and we will detail this further later in the section, when we move to practical implementation.

Joint embedding with the Infomax principle. Now, the question of *joint* representation of $(\boldsymbol{x}, \boldsymbol{y})$ remains. Applying the data processing inequality once more, we obtain for another encoder g that produces a representation of $\boldsymbol{y}$ that

$$I(\boldsymbol{x}; \boldsymbol{y}) \geq I(f(\boldsymbol{x}); \boldsymbol{y}) \geq I(f(\boldsymbol{x}); g(\boldsymbol{y})).$$

The joint objective

$$\max_{f,g} I(f(\boldsymbol{x}); g(\boldsymbol{y})) \tag{7.4.37}$$

is then a well-founded consequence of the Infomax principle.

To design the encoders f and g, recall the intuition that we obtained through our work in Example 7.6. There, we saw that in the case of a mixture-of-Gaussians model for $\boldsymbol{x}$, with the conditioning signal y identifying which specific mixture component generates the observation, a flexible denoiser architecture could be constructed via a suitable embedding of the class label y into the *same space* as the data $\boldsymbol{x}$. More precisely, such an embedding $\boldsymbol{u}_y \in \mathbb{R}^D$ allows, when correlated with the embeddings $\boldsymbol{U}_{\boldsymbol{x}}$ of $\boldsymbol{x} \in \mathbb{R}^D$ (generally a matrix) via $\boldsymbol{u}_y^\top \boldsymbol{U}_{\boldsymbol{x}}$, to 'pick out' only the embeddings relevant to the correct mixture component of the distribution. How exactly to design these layers for use in neural networks being trained on more complex nonlinear high-dimensional data is a challenging question that we will dig into in more detail below. But a necessary condition is clear: the embeddings f and g should map their inputs to a *common embedding space*, say $\mathbb{R}^d$, to facilitate downstream comparison when performing conditional sampling. This leads to the objective

$$\max_{f:\mathbb{R}^{D_{\boldsymbol{x}}} \to \mathbb{R}^d, g:\mathbb{R}^{D_{\boldsymbol{y}}} \to \mathbb{R}^d} I(f(\boldsymbol{x}); g(\boldsymbol{y})). \tag{7.4.38}$$

The next step is to instantiate a suitable computationally-convenient relaxation of the mutual information in (7.4.38) for use with large-scale datasets.

Practical Prescription for Learning Joint Embeddings: CLIP

A concrete and highly influential realization of learning joint embeddings for paired data is the method of Contrastive Language-Image Pre-training (CLIP) [RKH+21b]. Take the simple case of image-text pairs $[(\boldsymbol{x}_i, \boldsymbol{y}_i)]_{i=1}^N$ sampled from a dataset $\mathcal{D}$, where $\boldsymbol{x}_i$ is an image and $\boldsymbol{y}_i$ is a text caption. CLIP trains two encoders—an image encoder $f_\theta(\boldsymbol{x})$ and a text encoder $g_\phi(\boldsymbol{y})$—to map paired samples into a shared d-dimensional unit sphere (i.e. $\|f_\theta(\boldsymbol{x}_i)\|_2 = 1$ and $\|g_\phi(\boldsymbol{y}_i)\|_2 = 1$). To achieve this, we can adopt a simple symmetric cross-entropy loss that pushes the cosine similarity of matched pairs $(f_\theta(\boldsymbol{x}_i), g_\phi(\boldsymbol{y}_i))$ closer together while repelling unmatched pairs:

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{N} \mathbb{E}_{[(\boldsymbol{x}_i, \boldsymbol{y}_i)]_{i=1}^N \sim \mathcal{D}} \left[\sum_{i=1}^N \log \frac{\exp\left(f_\theta(\boldsymbol{x}_i)^\top g_\phi(\boldsymbol{y}_i)\right)}{\sum_{j=1}^N \exp\left(f_\theta(\boldsymbol{x}_i)^\top g_\phi(\boldsymbol{y}_j)\right)} \right.$$

$$+\sum_{i=1}^{N}\log\frac{\exp\left(g_\phi(\boldsymbol{y}_i)^\top f_\theta(\boldsymbol{x}_i)\right)}{\sum_{j=1}^{N}\exp\left(g_\phi(\boldsymbol{y}_i)^\top f_\theta(\boldsymbol{x}_j)\right)}\Bigg], \quad (7.4.39)$$

From the perspective of mutual information maximization, CLIP is a tractable surrogate for the lower bound on the mutual information between the image and text representations:[12]

$$I\big(f_\theta(\boldsymbol{x}), g_\phi(\boldsymbol{y})\big) \geq -\mathcal{L}_{\text{CLIP}} + \log N. \quad (7.4.40)$$

Thus, CLIP provides a practical recipe for learning joint embeddings for paired data that increases the mutual information between the paired samples. Empirically, CLIP has been shown to be effective at learning joint embeddings for image-text pairs, and has been used in a variety of applications, including image captioning, image-text retrieval, image generation and beyond. We will see more specific examples in Chapter 8 (with key implementation details described in detail in Section 8.3).

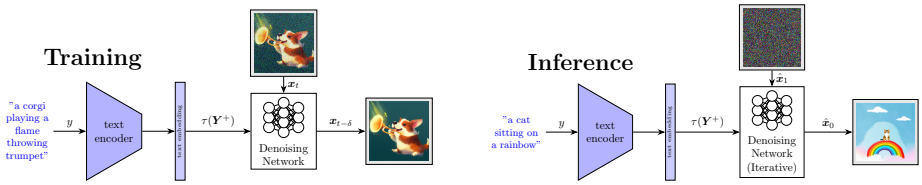
Application to Caption-Conditioned Image Generation

Finally, we will describe an end-to-end system that integrates joint image-text encoders, learned with the CLIP loss as described above, with conditioning operators that are evocative of the mixture-of-Gaussians operator (7.4.32) we saw in Example 7.6. These techniques formed the basis for the original open-source Stable Diffusion implementation [RBL+22], where the embedding and subsequent conditioning is performed not on a class label, but a text prompt, which describes the desired image content (Figure 7.14). This section will give a high-level overview of StableDiffusion-style image generation. For full implementation details sufficient to re-implement such a system from scratch, we refer to Chapter 8: specifically, Section 8.11 describes the process of encoding strings of text to a sequence of integer IDs; Section 8.3 describes the training of the necessary image-text encoders; and Sections 8.6 and 8.7 describe the training of the (latent) diffusion model.

Stable Diffusion follows the conditional generation methodology we outline in Section 7.4.1, with two key modifications: (i) The conditioning signal is a tokenized text prompt $\boldsymbol{Y} \in \mathbb{R}^{D_{\text{text}} \times N}$, rather than a class label; (ii) Image denoising is performed in "latent" space rather than on raw pixels, using a specialized, pretrained variational autoencoder pair $f : \mathbb{R}^{D_{\text{img}}} \to \mathbb{R}^{d_{\text{img}}}$, $g : \mathbb{R}^{d_{\text{img}}} \to \mathbb{R}^{D_{\text{img}}}$ (see Sections 6.1.4 and 8.6), where f is the encoder and g is the decoder. For issue (i), in the context of the iterative conditional denoising framework we have developed in Section 7.4.1, this concerns the parameterization of the denoisers $\bar{\boldsymbol{z}}_\theta(t, \boldsymbol{z}_t, \boldsymbol{Y}^+)$.[13] Rombach et al. [RBL+22] implement text conditioning

[12] This result holds under the assumption that the pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are i.i.d. within each batch of N [OLC+25, Lemma 1]. The tightness of the bound improves as N increases [OLV18]. However, for certain data distributions, it can remain loose for all N [MS19; WI20].

[13] In the setting of text conditioning, the 'augmented' label $\boldsymbol{Y}^+$, which is either the encoded text prompt or $\varnothing$, denoting unconditional denoising, is often implemented by mapping $\varnothing$ to the empty string "", then encoding this text prompt with the tokenizer as usual. This gives a simple, unified way to treat conditional and unconditional denoising with text conditioning.



(a) (b)

Figure 7.14: A high-level schematic of training and applying a text-to-image generative model, via conditional generation with a text prompt. **Left:** To train a text-to-image model, a large dataset of images paired with corresponding text captions is used. An encoder is used to map the captions to sequences of vectors, which are used as conditioning signals for a conditional denoiser, trained as described algorithmically in Section 7.4.1 (see Section 8.6 for implementation details). The text encoder may be pretrained and frozen, or jointly trained with the denoiser. **Right:** When applying a trained model, a desired text prompt is used as conditioning, then sampling is performed with the trained model, as in Algorithm 7.2 (*mutatis mutandis* for use with an encoded text prompt).

in the denoiser using a layer known as cross attention, inspired by the original encoder-decoder transformer architecture of Vaswani et al. [VSP+17b]. Cross attention is implemented as follows. We let $\tau : \mathbb{R}^{D_{\text{text}} \times N} \to \mathbb{R}^{d_{\text{model}} \times N_{\text{text}}}$ denote an encoding network for the text embeddings (often a causal transformer—see Section 8.11), and let $\psi : \mathbb{R}^{d_{\text{img}}} \to \mathbb{R}^{d_{\text{model}} \times N_{\text{img}}}$ denote the mapping corresponding to one of the intermediate representations in the denoiser.[14] Here, N_{text} is the maximum tokenized text prompt length, and N_{img} roughly corresponds to the number of image channels (layer-dependent) in the representation, which is fixed if the input image resolution is fixed. Cross attention (with K heads, and no bias) is defined as

$$\text{MHCA}(\boldsymbol{z}_t, \boldsymbol{Y}^+) = \boldsymbol{U}_{\text{out}} \begin{bmatrix} \text{SA}([\boldsymbol{U}^1_{\text{qry}}]^\top \psi(\boldsymbol{z}_t), [\boldsymbol{U}^1_{\text{key}}]^\top \tau(\boldsymbol{Y}^+), [\boldsymbol{U}^1_{\text{val}}]^\top \tau(\boldsymbol{Y}^+)) \\ \vdots \\ \text{SA}([\boldsymbol{U}^K_{\text{qry}}]^\top \psi(\boldsymbol{z}_t), [\boldsymbol{U}^K_{\text{key}}]^\top \tau(\boldsymbol{Y}^+), [\boldsymbol{U}^K_{\text{val}}]^\top \tau(\boldsymbol{Y}^+)) \end{bmatrix}, \tag{7.4.41}$$

where SA denotes the scaled dot-product attention operation in the transformer (which we recall in detail in Chapter 8: see Equations (8.2.17) and (8.2.18)), and $\boldsymbol{U}^k_* \in \mathbb{R}^{d_{\text{model}} \times d_{\text{attn}}}$ for $* \in \{\text{qry}, \text{key}, \text{val}\}$ (as well as the output projection $\boldsymbol{U}_{\text{out}}$) are the learnable parameters of the layer.

Notice that, by the definition of the self-attention operation, cross attention *outputs linear combinations of the value-projected text embeddings weighted by correlations between the image features and the text embeddings.* In the denoiser architecture used by Rombach et al. [RBL+22], self-attention residual blocks in the denoiser architecture, applied to the image representation at the current

[14] In practice, text-conditioned denoisers add cross attention layers at regular intervals within the forward pass of the denoiser, so ψ should be seen as layer-dependent, in contrast to τ.

layer and defined analogously to those in Equation (8.2.15) for the vision transformer, are followed by cross attention residual blocks of the form (7.4.41). Such a structure requires the text encoder τ to, in a certain sense, share some structure in its output with the image feature embedding ψ: this can be achieved via joint embedding, learned by a procedure such as CLIP as described above. Conceptually, this joint text-image embedding space and the cross attention layer itself bear a strong resemblance to the conditional mixture of Gaussians denoiser that we derived in the previous section (recall (7.4.28)), in the special case of a single token sequence. Deeper connections can be drawn in the multi-token setting following the rate reduction framework for deriving deep network architectures discussed in Chapter 5, and manifested in the derivation of the CRATE transformer-like architecture.

This same basic design has been further scaled to even larger model and dataset sizes, in particular in modern instantiations of Stable Diffusion [EKB+24], as well as in competing models such as FLUX.1 [LBB+25], Imagen [SCS+22], and DALL-E [RDN+22]. The conditioning mechanism of cross attention has also become ubiquitous in other applications, as in EgoAllo (Section 8.10) for conditioned pose generation and in Michelangelo (Section 8.8) for conditional 3D shape generation based on images or texts.

7.5 Conditional Inference with Measurement Self-Consistency

In this last section, we consider the more extreme, but actually ubiquitous, case for distribution learning in which we only have a set of observed samples $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ of the data $\boldsymbol{x}$, but no samples of $\boldsymbol{x}$ directly! In general, the observation $\boldsymbol{y} \in \mathbb{R}^d$ is of lower dimension than $\boldsymbol{x} \in \mathbb{R}^D$. To make the problem well-defined, we do assume that the observation model between $\boldsymbol{y}$ and $\boldsymbol{x}$ is known to belong to a certain family of analytical models, denoted as $\boldsymbol{y} = h(\boldsymbol{x}, \theta) + \boldsymbol{w}$, with θ either known or not known.

Let us first try to understand the problem conceptually with the simple case when the measurement function h is known and the observed $\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{w}$ is informative about $\boldsymbol{x}$. That is, we assume that h is surjective from the space of $\boldsymbol{x}$ to that of $\boldsymbol{y}$ and the support of the distribution $\boldsymbol{y}_0 = h(\boldsymbol{x}_0)$ is low-dimensional. This typically requires that the *extrinsic* dimension d of $\boldsymbol{y}$ is higher than the *intrinsic* dimension of the support of the distribution of $\boldsymbol{x}$. Without loss of generality, we may assume that there exist functions:

$$F(\boldsymbol{x}) = \boldsymbol{0}, \quad G(\boldsymbol{y}) = \boldsymbol{0}. \tag{7.5.1}$$

Notice that here we may assume that we know $G(\boldsymbol{y})$ but not $F(\boldsymbol{x})$. Let $\mathcal{S}_{\boldsymbol{y}} \doteq \{\boldsymbol{y} \mid G(\boldsymbol{y}) = \boldsymbol{0}\}$ be the support of $p(\boldsymbol{y})$. In general, $h^{-1}(\mathcal{S}_{\boldsymbol{y}}) = \{\boldsymbol{x} \mid G(h(\boldsymbol{x})) = \boldsymbol{0}\}$ is a superset of $\mathcal{S}_{\boldsymbol{x}} \doteq \{\boldsymbol{x} \mid F(\boldsymbol{x}) = \boldsymbol{0}\}$. That is, we have $h(\mathcal{S}_{\boldsymbol{x}}) \subseteq \mathcal{S}_{\boldsymbol{y}}$.

7.5.1 Linear Measurement Models

First, for simplicity, let us consider that the measurement is a linear function of the data $\boldsymbol{x}$ of interest:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}. \tag{7.5.2}$$

Here the matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is of full row rank and m is typically smaller than n. We assume $\boldsymbol{A}$ is known for now. We are interested in how to learn the distribution of $\boldsymbol{x}$ from such measurements. Since we no longer have direct samples of $\boldsymbol{x}$, we wonder whether we can still develop a denoiser for $\boldsymbol{x}$ with observations $\boldsymbol{y}$. Let us consider the following diffusion process:

$$\boldsymbol{y}_t = \boldsymbol{y}_0 + t\boldsymbol{g}, \quad \boldsymbol{y}_0 = \boldsymbol{A}(\boldsymbol{x}_0), \tag{7.5.3}$$

where $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Without loss of generality, we assume $\boldsymbol{A}$ is of full row rank, i.e., underdetermined. Let us define the corresponding process $\boldsymbol{x}_t$ as one that satisfies:

$$\boldsymbol{y}_t = \boldsymbol{A}\boldsymbol{x}_t. \tag{7.5.4}$$

From the denoising process of $\boldsymbol{y}_t$, we have

$$\boldsymbol{y}_{t-s} \approx \boldsymbol{y}_t + st\nabla \log p_t(\boldsymbol{y}_t). \tag{7.5.5}$$

Then we have:

$$\boldsymbol{A}\boldsymbol{x}_{t-s} \approx \boldsymbol{A}\boldsymbol{x}_t + st\nabla \log p_t(\boldsymbol{A}\boldsymbol{x}_t), \tag{7.5.6}$$

for a small $s > 0$. So $\boldsymbol{x}_{t-s}$ and $\boldsymbol{x}_t$ need to satisfy:

$$\boldsymbol{A}(\boldsymbol{x}_{t-s} - \boldsymbol{x}_t) \approx st\nabla \log p_t(\boldsymbol{A}\boldsymbol{x}_t). \tag{7.5.7}$$

Among all $\boldsymbol{x}_{t-s}$ that satisfy the above constraint, we arbitrarily choose the one that minimizes the distance $\|\boldsymbol{x}_{t-s} - \boldsymbol{x}_t\|_2^2$. Therefore, we obtain a "denoising" process for $\boldsymbol{x}_t$:

$$\boldsymbol{x}_{t-s} \approx \boldsymbol{x}_t + st\boldsymbol{A}^\dagger \nabla \log p_t(\boldsymbol{A}\boldsymbol{x}_t). \tag{7.5.8}$$

Notice that this process does not sample from the distribution of $\boldsymbol{x}_t$. In particular, there are components of $\boldsymbol{x}$ in the null space/kernel of $\boldsymbol{A}$ that can never be recovered from observations. Thus more information is needed to recover the full distribution of $\boldsymbol{x}$, strictly speaking. But this recovers the component of $\boldsymbol{x}$ that is orthogonal to the null space of $\boldsymbol{A}$.

7.5.2 Learning 3D World Model from 2D Images

In practice, the measurement model is often nonlinear or only partially known. A typical problem of this kind is actually behind how we can learn a working model of the external world from the images perceived, say through our eyes, telescopes or microscopes. In particular, humans and animals are able to build a model of the 3D world (or 4D for a dynamical world) through a sequence

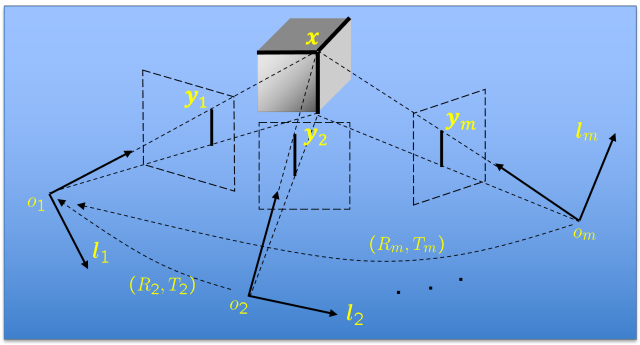


Figure 7.15: Relationship between a 3D object/scene and its 2D projections. Here we illustrate the projection of a point $\boldsymbol{x}$ and a line intersecting the point.

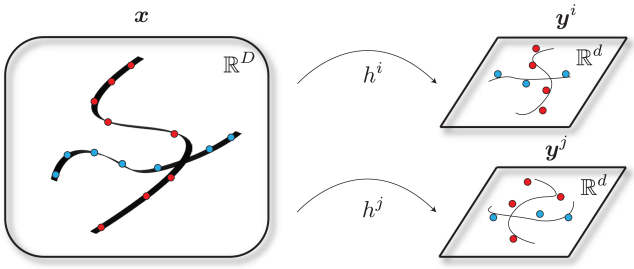


Figure 7.16: **Inference with distributed measurements.** We have a low-dimensional distribution $\boldsymbol{x}$ (here, similarly to Figure 7.1, depicted as a union of two 2-dimensional manifolds in $\mathbb{R}^3$) and a measurement model $\boldsymbol{y}^i = h^i(\boldsymbol{x}) + \boldsymbol{w}^i$. As before, we want to infer various properties of the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$, where $\boldsymbol{y}$ is the collection of all the measurements $\boldsymbol{y}^i$.

of its 2D projections—a sequence of 2D images (or stereo image pairs). The mathematical or geometric model of the projection is generally known:

$$\boldsymbol{y}^i = h(\boldsymbol{x}, \theta^i) + \boldsymbol{w}^i, \tag{7.5.9}$$

where $h(\cdot)$ represents a (perspective) projection of the 3D (or 4D) scene from a certain camera view at time t_i to a 2D image (or a stereo pair) and $\boldsymbol{w}$ is some possibly additive small measurement noise. Figure 7.15 illustrates this relationship concretely, while Figure 7.16 illustrates the model problem in the abstract. A full exposition of geometry related to multiple 2D views of a 3D scene is beyond the scope of this book. Interested readers may refer to the book [MKS+04]. For now, all we need to proceed is that such projections are well understood and multiple images of a scene contain sufficient information about the scene.

In general, we would like to learn the distribution $p(\boldsymbol{x})$ of the 3D (or 4D)

world scene $\boldsymbol{x}$[15] from the perceived 2D images of the world so far. The primary function of such a (visual) world model is to allow us to recognize places where we had been before or predict what the current scene would look like in a future time at a new viewpoint.

Let us first examine the special but important case of stereo vision. In this case, we have two calibrated views of the 3D scene $\boldsymbol{x}$:

$$\boldsymbol{y}^0 = h(\boldsymbol{x}, \theta^0) + \boldsymbol{w}^0, \quad \boldsymbol{y}^1 = h(\boldsymbol{x}, \theta^1) + \boldsymbol{w}^1, \tag{7.5.10}$$

where parameters θ^0 and θ^1 for the view poses can be assumed to be known. $\boldsymbol{y}^0$ and $\boldsymbol{y}^1$ are two 2D-projections of the 3D scene $\boldsymbol{x}$. We may also assume that they have the same marginal distribution $p(\boldsymbol{y})$ and we have learned a diffusion and denoising model for it. That is, we know the denoiser:

$$\mathbb{E}[\boldsymbol{y} \mid \boldsymbol{y}_t = \boldsymbol{\nu}] = \boldsymbol{\nu} + t^2 \nabla_{\boldsymbol{\nu}} \log p_t(\boldsymbol{\nu}). \tag{7.5.11}$$

Or, furthermore, we may assume that we have a sufficient number of samples of stereo pairs $(\boldsymbol{y}^0, \boldsymbol{y}^1)$ and have also learned the joint distribution of the pairs. By a little abuse of notation, we also use $\boldsymbol{y} = h(\boldsymbol{x})$ to indicate the pair $\boldsymbol{y} = (\boldsymbol{y}^0, \boldsymbol{y}^1)$ and $p(\boldsymbol{y})$ as the learned probability distribution of the pair (say via a denoiser as above).

The main question now is: How to learn (a representation for) the distribution of the 3D scene $\boldsymbol{x}$ from its two projections with known relationships? People might question the rationale for doing this: why is this necessary if the function $h(\cdot)$ is largely invertible? That is, the observation $\boldsymbol{y}$ can largely determine the unknown $\boldsymbol{x}$, which is kind of the case for stereo—in general, two (calibrated) images contain sufficient information about the scene depth, from the given vantage point. However, 2D images are far from the most compact representation of the 3D scene as the same scene can produce infinitely many (highly correlated) 2D images or image pairs. In fact, a good representation of a 3D scene should be invariant to the viewpoint. Hence, a correct representation of the distribution of 3D scenes should be much more compact and structured than the distribution of 2D images, stereo pairs, or image-depth pairs.

Consider the (inverse) denoising process for the diffusion: $\boldsymbol{y}_t = \boldsymbol{y} + t\boldsymbol{g}$ in (7.5.11), where $\boldsymbol{g}$ is standard Gaussian. From the denoising process of (7.5.11), we have

$$\boldsymbol{y}_{t-s} = \boldsymbol{y}_t + st \nabla_{\boldsymbol{y}} \log p_t(\boldsymbol{y}_t). \tag{7.5.12}$$

We try to find a corresponding "denoising" process of $\boldsymbol{x}_t$ such that $\boldsymbol{x}$ is related to $\boldsymbol{y}$ as:

$$\boldsymbol{y} = h(\boldsymbol{x}). \tag{7.5.13}$$

Then we have:

$$h(\boldsymbol{x}_{t-s}) \approx h(\boldsymbol{x}_t) + st \nabla_{\boldsymbol{y}} \log p_t(h(\boldsymbol{x}_t)), \tag{7.5.14}$$

[15] Here by abuse of notation, we use $\boldsymbol{x}$ to represent either a point in 3D or a sample of an entire 3D object or a scene that consists of many points.

for a small $s > 0$. Suppose $\boldsymbol{x}_{t-s} = \boldsymbol{x}_t + s\boldsymbol{v}$ for some vector $\boldsymbol{v}$ and small increment s. We have

$$h(\boldsymbol{x}_{t-s}) \approx h(\boldsymbol{x}_t) + \frac{\partial h}{\partial \boldsymbol{x}}(\boldsymbol{x}_t) \cdot \boldsymbol{v}s \doteq h(\boldsymbol{x}_t) + \boldsymbol{A}(\boldsymbol{x}_t)\boldsymbol{v}s. \tag{7.5.15}$$

Hence, we have

$$\boldsymbol{A}(\boldsymbol{x}_t)\boldsymbol{v} = t\nabla_{\boldsymbol{y}} \log p_t(h(\boldsymbol{x}_t)). \tag{7.5.16}$$

Geometrically the vector $\boldsymbol{v}$ in the domain of $\boldsymbol{x}$ can be viewed as the pullback of the vector field $t\nabla \log p_t(\boldsymbol{y})$ under the map $\boldsymbol{y} = h(\boldsymbol{x})$. In general, as before, we may (arbitrarily) choose $\boldsymbol{v}$ to be the minimum 2-norm vector that satisfies the pullback relationship. Hence, we can express $\hat{\boldsymbol{x}}_{t-s}$ approximately as:

$$\hat{\boldsymbol{x}}_{t-s} \approx \boldsymbol{x}_t + st\boldsymbol{A}(\boldsymbol{x}_t)^\dagger \nabla_{\boldsymbol{y}} \log p_t(h(\boldsymbol{x}_t)). \tag{7.5.17}$$

Remark 7.2 (Parallel Sensing and Distributed Denoising.). There is something very interesting about the above equation (7.5.17). It seems to suggest we could try to learn the distribution of $\boldsymbol{x}$ through a process that is coupled with (many of) its (partial) observations:

$$\boldsymbol{y}^i = h^i(\boldsymbol{x}) + \boldsymbol{w}^i, i = 1, \ldots, K. \tag{7.5.18}$$

In this case, we obtain a set of equations that the vector field $\boldsymbol{v}$ in the domain of $\boldsymbol{x}$ should satisfy:

$$\boldsymbol{A}^i(\boldsymbol{x}_t)\boldsymbol{v} = t\nabla_{\boldsymbol{y}^i} \log p_t(h^i(\boldsymbol{x}_t)), \tag{7.5.19}$$

where $\boldsymbol{A}^i(\boldsymbol{x}_t) = \frac{\partial h^i}{\partial \boldsymbol{x}}(\boldsymbol{x}_t)$. The final $\boldsymbol{v}$ can be chosen as a "centralized" solution that satisfies all the above equations, or it could be chosen as a certain (stochastically) "aggregated" version of all $\boldsymbol{v}^i$:

$$\boldsymbol{v}^i = t\boldsymbol{A}^i(\boldsymbol{x}_t)^\dagger \big[\nabla_{\boldsymbol{y}^i} \log p_t(h^i(\boldsymbol{x}_t))\big], \quad i = 1, \ldots, K, \tag{7.5.20}$$

that are computed in a parallel and distributed fashion? An open question here is exactly what the so-defined "denoising" process for $\boldsymbol{x}_t$ converges to, even in the linear measurement model case. When would it converge to a distribution that has the same low-dimensional support as the original $\boldsymbol{x}_0$, as $\boldsymbol{y}_t$ converges to $\boldsymbol{y} = h(\boldsymbol{x}_0)$?

Visual World Model from Uncalibrated Image Sequences. In the above derivation, we have assumed that the measurement model $h(\cdot)$ is fully known. In the case of stereo vision, this is rather reasonable as the relative pose (and calibration) of the two camera views (or two eyes[16]) is usually known in advance. Hence, through the stereo image pairs, in principle we should be able to learn the distribution of 3D scenes, at least the ego-centric distribution of 3D scenes. However, the low-dimensional structures of the so-called learned distribution contain variation caused by changing the viewpoints. That is, the appearance

[16]The relative pose of our two eyes is well known to our brain.

of the stereo images varies when we change our viewpoints with respect to the same 3D scene. For many practical vision tasks (such as localization and navigation), it is important that we can decouple this variation of viewpoints from an invariant representation of (the distribution of) 3D scenes.

Remark 7.3. Note that the above goal aligns well with Klein's Erlangen Program for modern geometry, which is to study invariants of a manifold under a group of transformations. Here, we may view the manifold of interest as the distribution of ego-centric representations of 3D scenes. We have learned that it admits a group of three-dimensional rigid-body motion acting on it. It is remarkable that our brain has learned to effectively decouple such transformations from the observed 3D world.

Notice that we have studied learning representations that are invariant to translation and rotation in a limited setting in Chapter 5. We know that the associated compression operators take the necessary form of (multi-channel) convolutions, hence leading to the (deep) convolutional neural networks. Nevertheless, operators that are associated with compression or denoising that are invariant to more general transformation groups remain elusive to characterize [CW16b]. For the 3D Vision problem in its most general setting, we know the change of our viewpoints can be well modeled as a rigid-body motion. However, the exact relative motion of our eyes between different viewpoints is usually not known. More generally, there could also be objects (e.g., cars, humans, hands) moving in the scene and we normally do not know their motion either. How can we generalize the problem of learning the distribution of 3D scenes with calibrated stereo pairs to such more general settings? More precisely, we want to learn a compact representation $\boldsymbol{x}$ of the 3D scenes that is invariant to the camera/eye motions. Once such a representation is learned, we could sample and generate a 3D scene and render images or stereo pairs from arbitrary poses.

To this end, note that we can model a sequence of stereo pairs as:

$$\boldsymbol{y}^k = h(\boldsymbol{x}^k, \theta^k), \quad k = 1, \dots, K, \tag{7.5.21}$$

where $h(\cdot)$ represents the projection map from 3D to 2D. θ^k denotes the rigid-body motion parameters of the kth view, with respect to some canonical frame in the world. $\boldsymbol{x}^k$ represents the 3D scene at time k. If the scene is static, $\boldsymbol{x}^k$ should all be the same $\boldsymbol{x}^k = \boldsymbol{x}$. To simplify the notation, we may denote the set of k equations as one:

$$\boldsymbol{Y} = H(\boldsymbol{x}, \Theta). \tag{7.5.22}$$

We may assume that we are given many samples of such stereo image sequences $\{\boldsymbol{Y}_i\}$. The problem is how to recover the associated motion sequence $\{\Theta_i\}$ and learn the distribution of the scene $\boldsymbol{x}$ (that is invariant to the motion). To the best of our knowledge, this remains an open challenging problem, probably as the final frontier for the 3D Vision problem.

7.6 Summary and Notes

Measurement matching without clean samples. In our development of conditional sampling, we considered measurement matching under an observation model (7.3.9), where we assume that we have paired data $(\boldsymbol{x}, \boldsymbol{y})$—i.e., ground truth for each observation $\boldsymbol{y}$. In many practically relevant inverse problems, this is not the case: one of the most fundamental examples is in the context of compressed sensing, which we recalled in Chapter 2, where we need to reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$ using prior knowledge about $\boldsymbol{x}$ (i.e., sparsity). In the setting of denoising-diffusion, we have access to an implicit prior for $\boldsymbol{x}$ via the learned denoisers $\bar{\boldsymbol{x}}_\theta(t, \boldsymbol{\xi})$. Can we still perform conditional sampling without access to ground truth samples $\boldsymbol{x}$?

For intuition as to why this might be possible, we recall a classical example from statistics known as Stein's unbiased risk estimator (SURE). Under an observation model $\boldsymbol{x}_t = \boldsymbol{x} + t\boldsymbol{g}$ with $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $t > 0$, it turns out that for any weakly differentiable $f : \mathbb{R}^D \to \mathbb{R}^D$,

$$\mathbb{E}_{\boldsymbol{g}}\left[\|\boldsymbol{x} - f(\boldsymbol{x} + t\boldsymbol{g})\|_2^2\right] = \mathbb{E}_{\boldsymbol{g}}\left[\|\boldsymbol{x} + t\boldsymbol{g} - f(\boldsymbol{x} + t\boldsymbol{g})\|_2^2 + 2t^2 \nabla \cdot f(\boldsymbol{x} + t\boldsymbol{g})\right] - t^2 D, \tag{7.6.1}$$

where $\nabla\cdot$ denotes the divergence operator:

$$\nabla \cdot f = \sum_{i=1}^{D} \partial_i f_i.$$

The $\boldsymbol{x}$-dependent part of the RHS of Equation (7.6.1) is called Stein's unbiased risk estimator (SURE). If we take expectations over $\boldsymbol{x}$ in Equation (7.6.1), note that the RHS can be written as an expectation with respect to $\boldsymbol{x}_t$—in particular, the mean-squared error of *any denoiser* f can be estimated *solely from noisy samples*! This remarkable fact, in refined forms, constitutes the basis for many practical techniques for performing image restoration, denoising-diffusion, etc. using only noisy data: notable examples include the "noise2noise" paradigm [LMH+18] and Ambient Diffusion [DSD+23a].

As a fun aside, we point out that Equation (7.6.1) leads to an alternate proof of Tweedie's formula (Theorem 3.2). At a high level, one takes expectations over $\boldsymbol{x}$ and expresses the main part of the RHS of Equation (7.6.1) equivalently, via integration by parts, as

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{x}_t}\left[\|\boldsymbol{x}_t - f(\boldsymbol{x}_t)\|_2^2 + 2t^2 \nabla \cdot f(\boldsymbol{x}_t)\right] \\ = \; & \mathbb{E}_{\boldsymbol{x}_t}\left[\|\boldsymbol{x}_t - f(\boldsymbol{x}_t)\|_2^2\right] - 2t^2 \int \langle \nabla p_{\boldsymbol{x}_t}(\boldsymbol{\xi}), f(\boldsymbol{\xi}) \rangle \, \mathrm{d}\boldsymbol{\xi}. \end{aligned} \tag{7.6.2}$$

This is a quadratic function of f, and formally taking derivatives gives that the optimal f satisfies Tweedie's formula (Theorem 3.2). This argument can be made rigorous using basic ideas from the calculus of variations.

Corrections to the Diffusion Posterior Sampling (DPS) approximation. In Example 7.4 and in particular in Figure 7.11, we pointed out a limitation of the DPS approximation Equation (7.3.26) at small levels of measurement noise. This limitation is well-understood, and a principled approach to ameliorating it has been proposed by Rozet et al. [RAL+24]. The approach involves incorporating an additional estimate for the variance of the noisy posterior $p_{\boldsymbol{x}|\boldsymbol{x}_t}$ to Equation (7.3.26)—we refer to the paper for details. Natural estimates for the posterior variance are slightly less scalable than DPS itself due to the need to invert an affine transformation of the Jacobian of the posterior denoiser $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}]$ (a large matrix). This is done relatively efficiently by Rozet et al. [RAL+24] using automatic differentiation and an approximation for the inverse based on conjugate gradients. It seems that it should be possible to improve further over this approach (say, using classical ideas from second-order optimization).

More about measurement matching and diffusion models for inverse problems. Diffusion models have become an extremely popular tool for solving inverse problems arising in scientific applications. Many more methods beyond the simple DPS algorithm we have presented in Algorithm 7.1 have been developed and continue to be developed, as the area is evolving rapidly. Popular and performant classes of approaches beyond DPS, which we have presented due to its generality, include variable splitting approaches like DAPS [ZCB+24], which allow for specific measurement constraints to be enforced much more strongly than in DPS, and exact approaches that can avoid the use of approximations as in DPS, such as TDS [WTN+23]. For more on this area, we recommend [ZCZ+25], which functions simultaneously as a survey and a benchmark of several popular methods on specific scientific inverse problem datasets.

History of the Infomax principle. The Infomax principle [Lin88] that we introduced in our discussion of joint embedding learning and CLIP in Section 7.4.2 is a well-known and important foundation for much of modern research into unsupervised representation learning. For further reading, we refer to [OLV18], which our conceptual discussion in Section 7.4.2 expands upon, and [OLC+25], which details further important theoretical issues associated with optimizing Infomax-type objectives from finite samples.

7.7 Exercises and Extensions

Exercise 7.1 (Posterior Variance Correction to DPS). 1. Using the code provided in the book GitHub for implementing Figure 7.11, implement the posterior variance correction proposed by Rozet et al. [RAL+24].

2. Verify that it ameliorates the posterior collapse at low noise variance issue observed in Figure 7.11.

3. Discuss any issues of sampling correctness that are retained or introduced by the corrected method, as well as its efficiency, relative to diffusion posterior sampling (DPS).

Exercise 7.2 (Conditional Sampling on MNIST). 1. Train a simple classifier for the MNIST dataset, using an architecture of your choice. Additionally train a denoiser suitable for use in conditional sampling (Algorithm 7.2, since this denoiser can be used for unconditional denoising as well).

2. Integrate the classifier into a conditional sampler based on classifier guidance, as described in the first part of Section 7.4.1. Evaluate the resulting samples in terms of faithfulness to the conditioning class (visually; in terms of nearest neighbor; in terms of the output of the classifier).

3. Integrate the classifier into a conditional sampler based on classifier-free guidance, as described in Section 7.4.1 and Algorithm 7.2. Perform the same evaluation as in the previous step, and compare the results.

4. Repeat the experiment on the CIFAR-10 dataset.