Understanding Deep Learning

Chapter 16: Normalizing flows

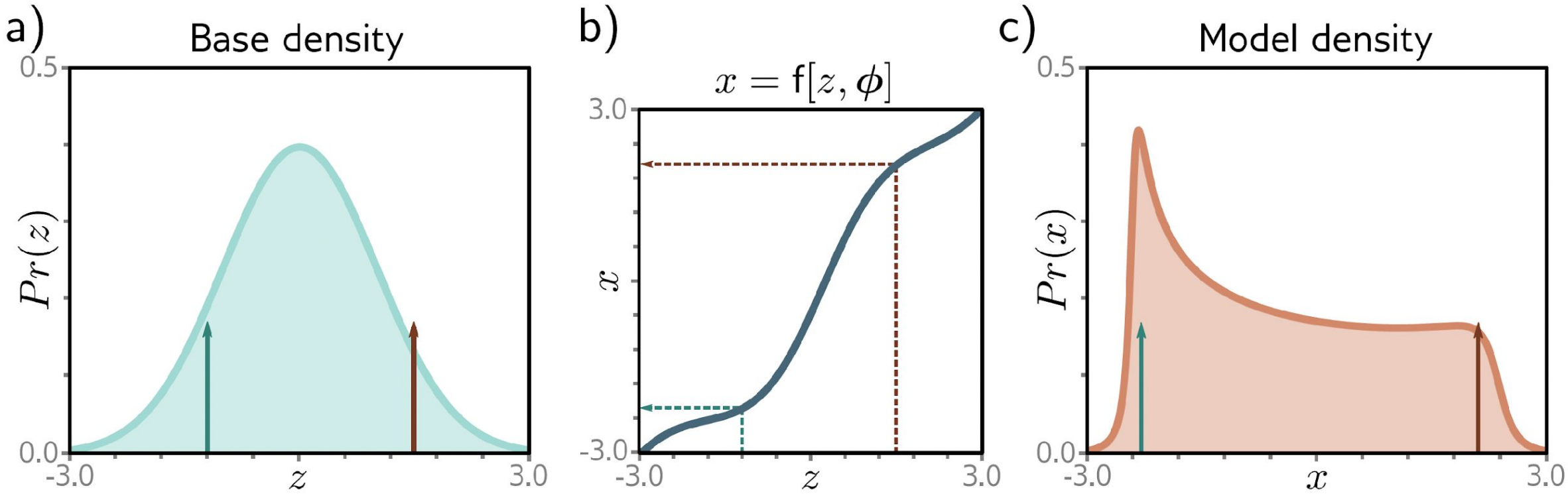

Figure 16.1 Transforming probability distributions. a) The base density is a standard normal defined on a latent variable z. b) This variable is transformed by a function $x = \mathrm{f}[z, \boldsymbol{\phi}]$ to a new variable x, which c) has a new distribution. To sample from this model, we draw values z from the base density (green and brown arrows in panel (a) show two examples). We pass these through the function $\mathrm{f}[z, \boldsymbol{\phi}]$ as shown by dotted arrows in panel (b) to generate the values of x, which are indicated as arrows in panel (c).

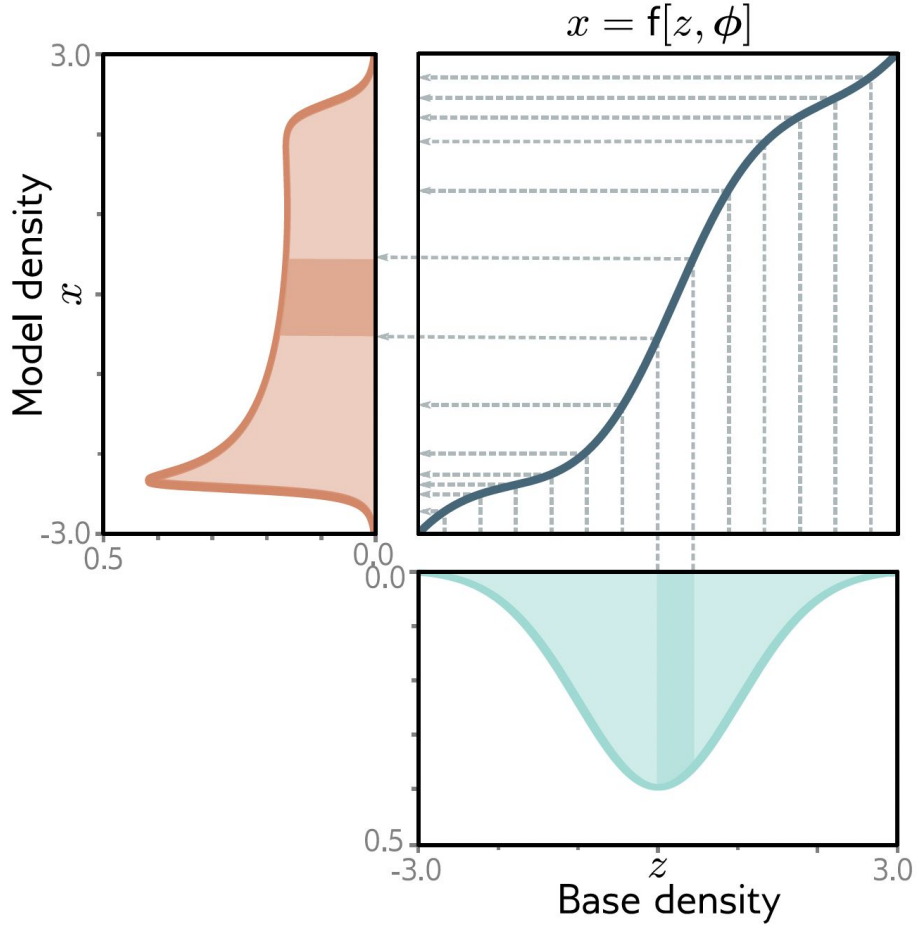

Figure 16.2 Transforming distributions. The base density (cyan, bottom) passes through a function (blue curve, top right) to create the model density (orange, left). Consider dividing the base density into equal intervals (gray vertical lines). The probability mass between adjacent lines must remain the same after transformation. The cyan-shaded region passes through a part of the function where the gradient is larger than one, so this region is stretched. Consequently, the height of the orange-shaded region must be lower so that it retains the same area as the cyan-shaded region. In other places (e.g., $z = -2$), the gradient is less than one, and the model density increases relative to the base density.

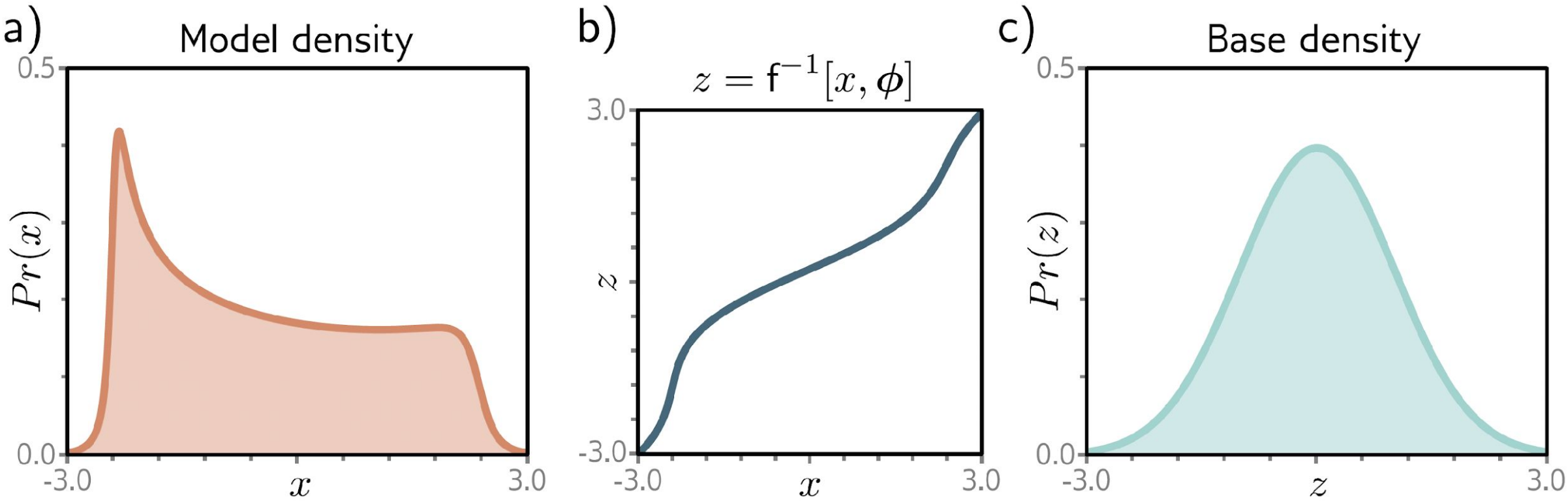

Figure 16.3 Inverse mapping (normalizing direction). If the function is invertible, then it's possible to transform the model density back to the original base density. The probability of a point x under the model density depends partly on the probability of the equivalent point z under the base density (see equation 16.1).

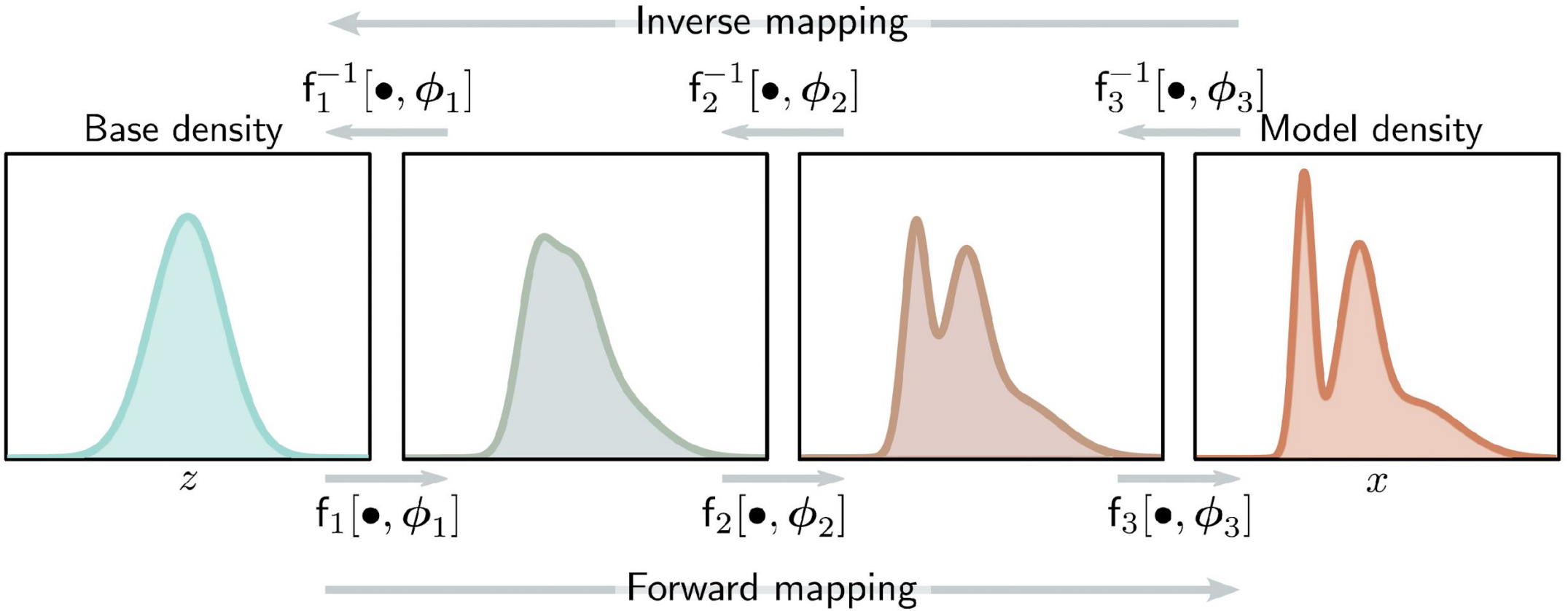

Figure 16.4 Forward and inverse mappings for a deep neural network. The base density (left) is gradually transformed by the network layers $f_1[\bullet, \phi_1], f_2[\bullet, \phi_2], \ldots$ to create the model density. Each layer is invertible, and we can equivalently think of the inverse of the layers as gradually transforming (or "flowing") the model density back to the base density.

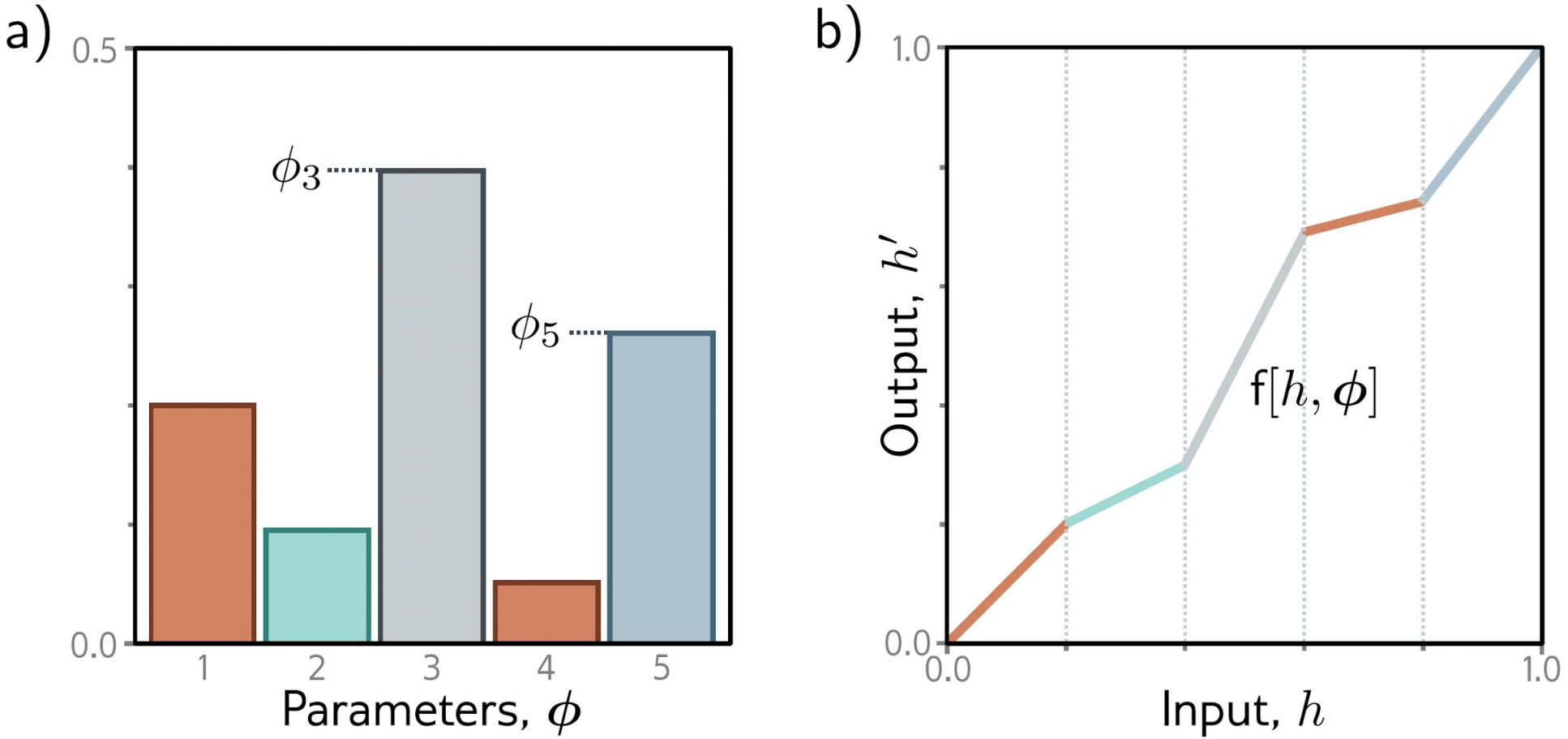

Figure 16.5 piecewise linear mapping. An invertible piecewise linear mapping $h' = \text{f}[h, \boldsymbol{\phi}]$ can be created by dividing the input domain $h \in [0, 1]$ into K equally sized regions (here $K = 5$). Each region has a slope with parameter, $\boldsymbol{\phi}_k$. a) If these parameters are positive and sum to one, then b) the function will be invertible and map to the output domain $h' \in [0, 1]$.

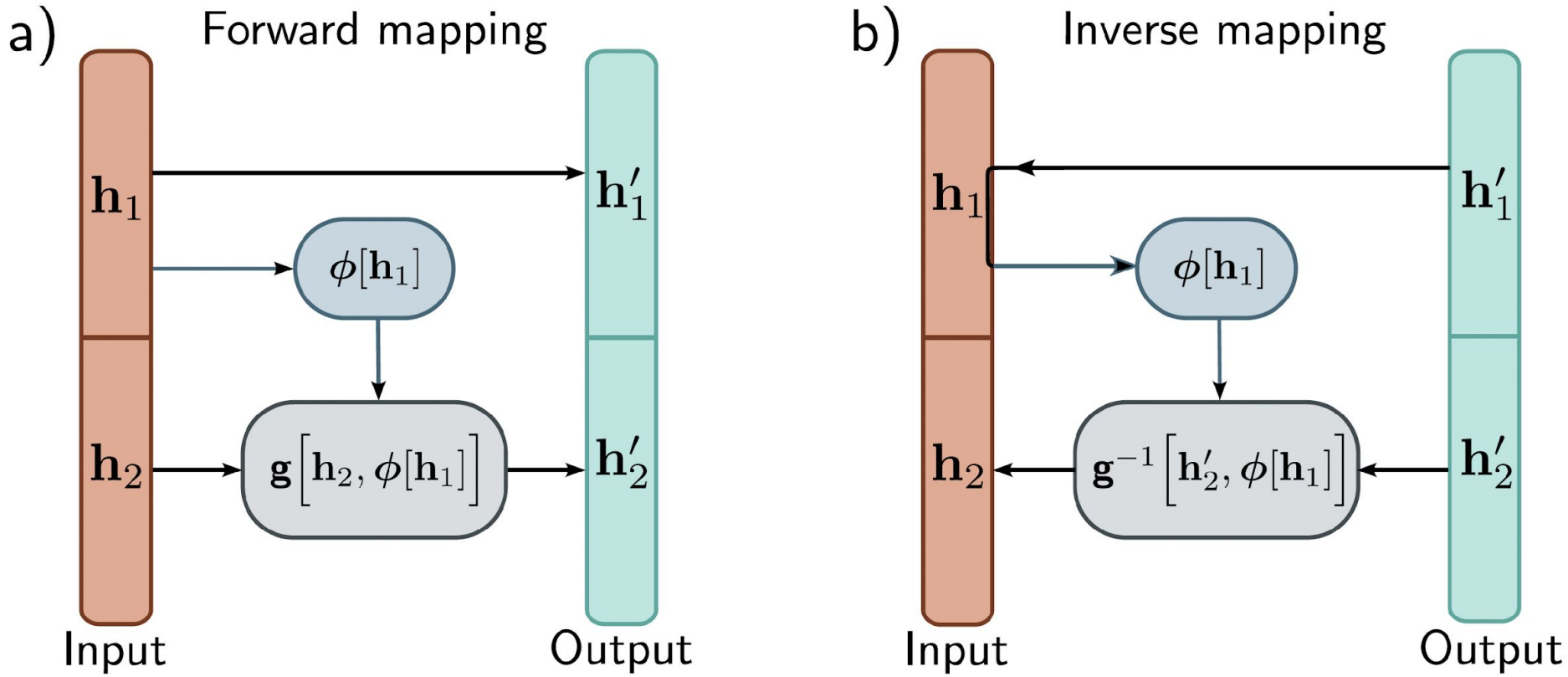

Figure 16.6 Coupling flows. a) The input (orange vector) is divided into $\mathbf{h}_1$ and $\mathbf{h}_2$. The first part $\mathbf{h}'_1$ of the output (cyan vector) is a copy of $\mathbf{h}_1$. The output $\mathbf{h}'_2$ is created by applying an invertible transformation $\mathbf{g}[\bullet, \boldsymbol{\phi}]$ to $\mathbf{h}_2$, where the parameters $\boldsymbol{\phi}$ are themselves a (not necessarily invertible) function of $\mathbf{h}_1$. b) In the inverse mapping, $\mathbf{h}_1 = \mathbf{h}'_1$. This allows us to calculate the parameters $\boldsymbol{\phi}[\mathbf{h}_1]$ and then apply the inverse $\mathbf{g}^{-1}[\mathbf{h}'_2, \boldsymbol{\phi}]$ to retrieve $\mathbf{h}_2$.

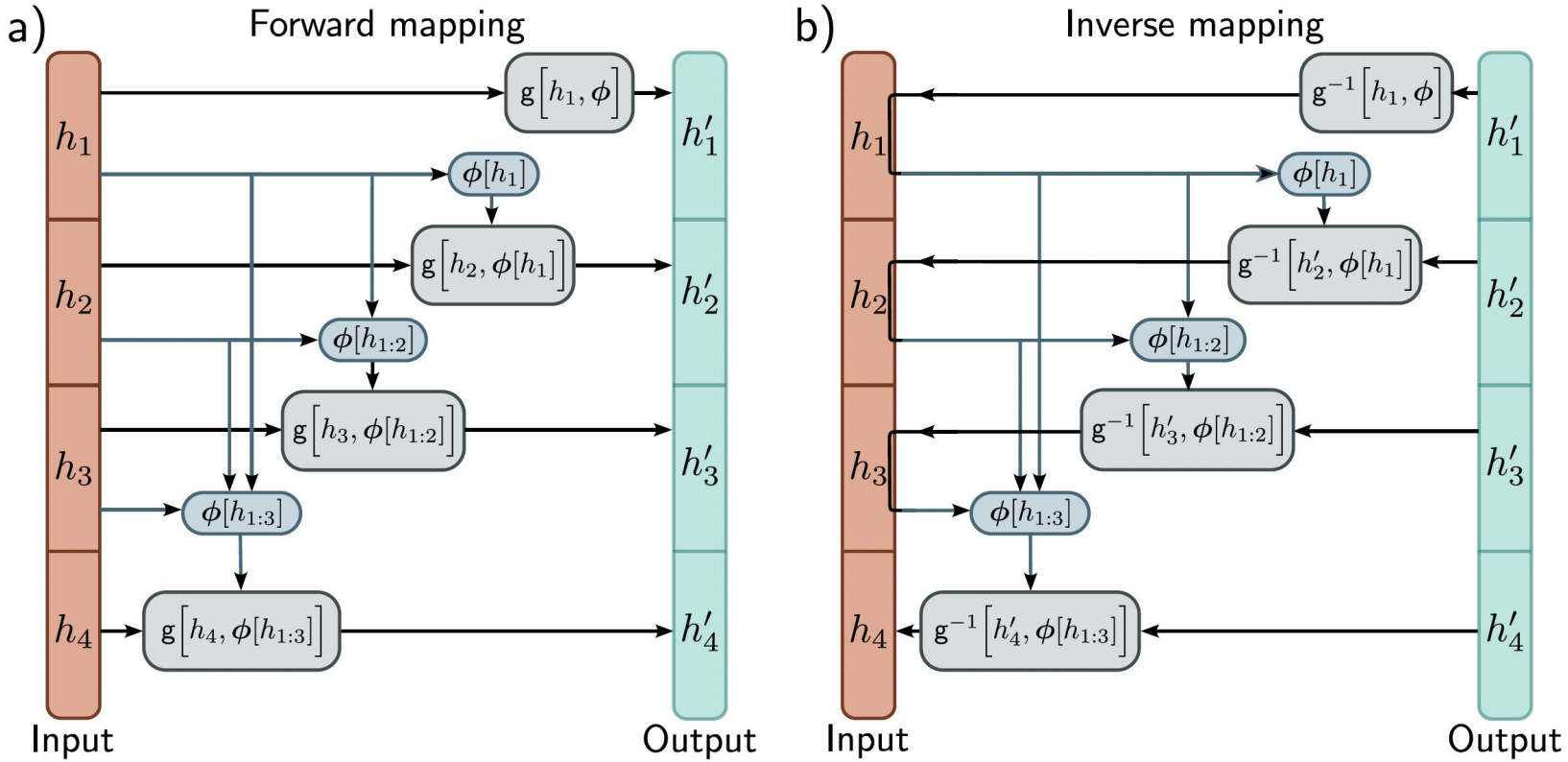

Figure 16.7 Autoregressive flows. The input $\mathbf{h}$ (orange column) and output $\mathbf{h}'$ (cyan column) are split into their constituent dimensions (here four dimensions). a) Output h'_1 is an invertible transformation of input h_1. Output h'_2 is an invertible function of input h_2 where the parameters depend on h_1. Output h'_3 is an invertible function of input h_3 where the parameters depend on previous inputs h_1 and h_2, and so on. None of the outputs depend on one another, so they can be computed in parallel. b) The inverse of the autoregressive flow is computed using a similar method as for coupling flows. However, notice that to compute h_2 we must already know h_1, to compute h_3, we must already know h_1 and h_2, and so on. Consequently, the inverse cannot be computed in parallel.

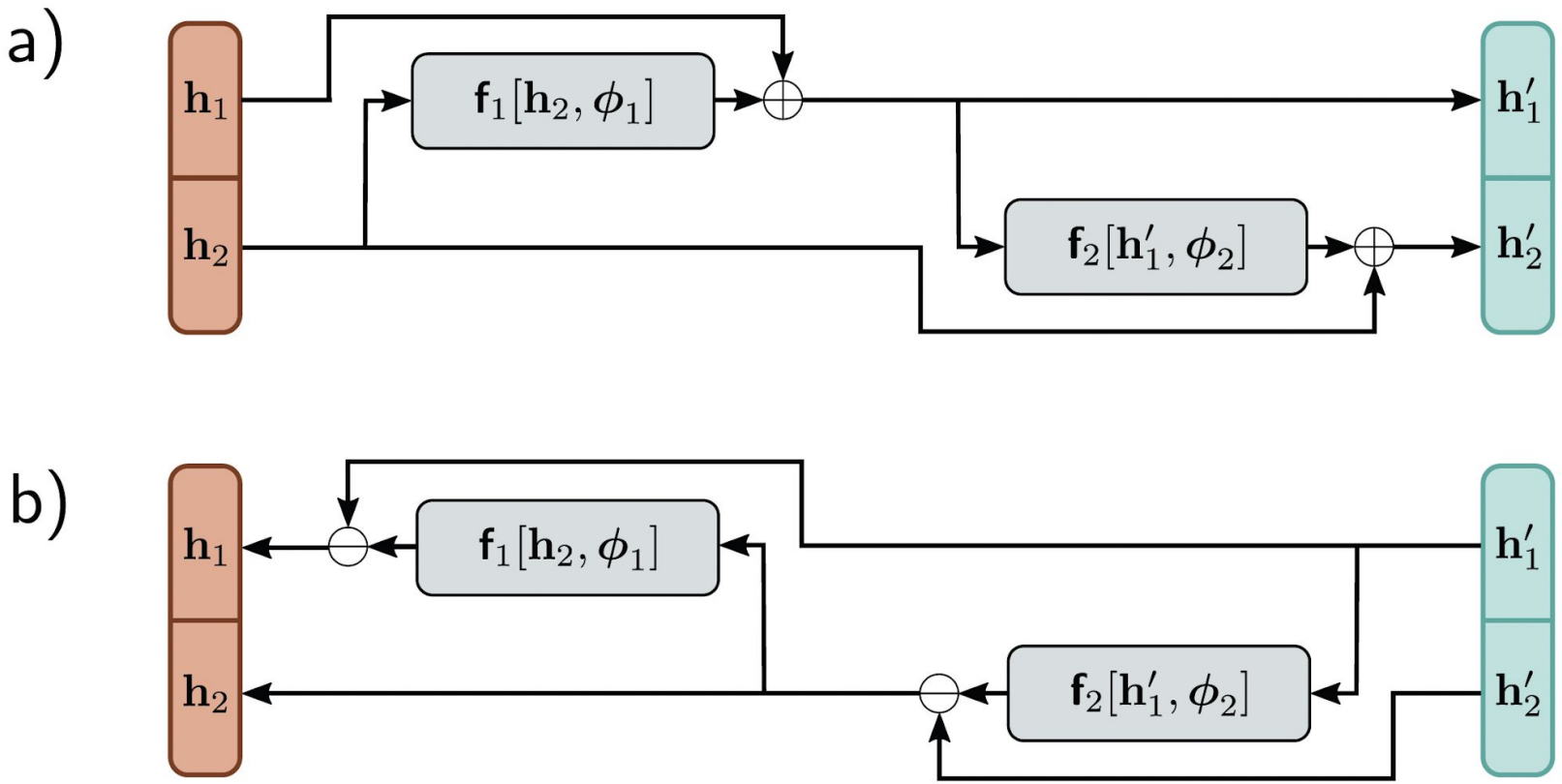

Figure 16.8 Residual flows. a) An invertible function is computed by splitting the input into $\mathbf{h}_1$ and $\mathbf{h}_2$ and creating two residual layers. In the first, $\mathbf{h}_2$ is processed and $\mathbf{h}_1$ is added. In the second, the result is processed, and $\mathbf{h}_2$ is added. b) In the reverse mechanism the functions are computed in the opposite order, and the addition operation becomes subtraction.

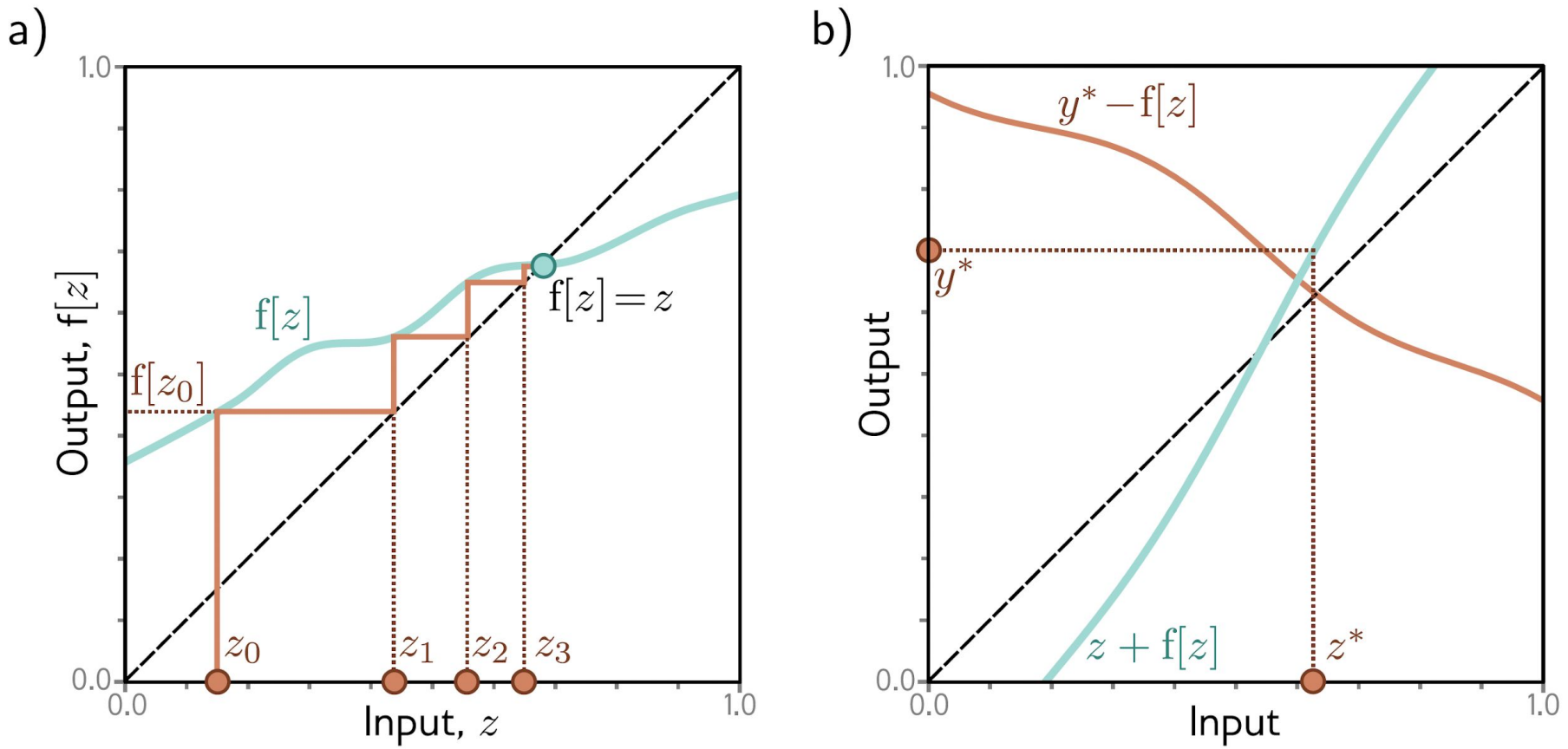

Figure 16.9 Contraction mappings. If a function has an absolute slope of less than one everywhere, iterating the function converges to a fixed point $\mathrm{f}[z] = z$. a) Starting at z_0, we evaluate $z_1 = \mathrm{f}[z_0]$. We then pass z_1 back into the function and iterate. Eventually, the process converges to the point where $\mathrm{f}[z] = z$ (i.e., where the function crosses the dashed diagonal identity line). b) This can be used to invert equations of the form $y = z + \mathrm{f}[z]$ for a value y^* by noticing that the fixed point of $y^* - f[z]$ (where the orange line crosses the dashed identity line) is at the same position as where $y^* = z + \mathrm{f}[z]$.

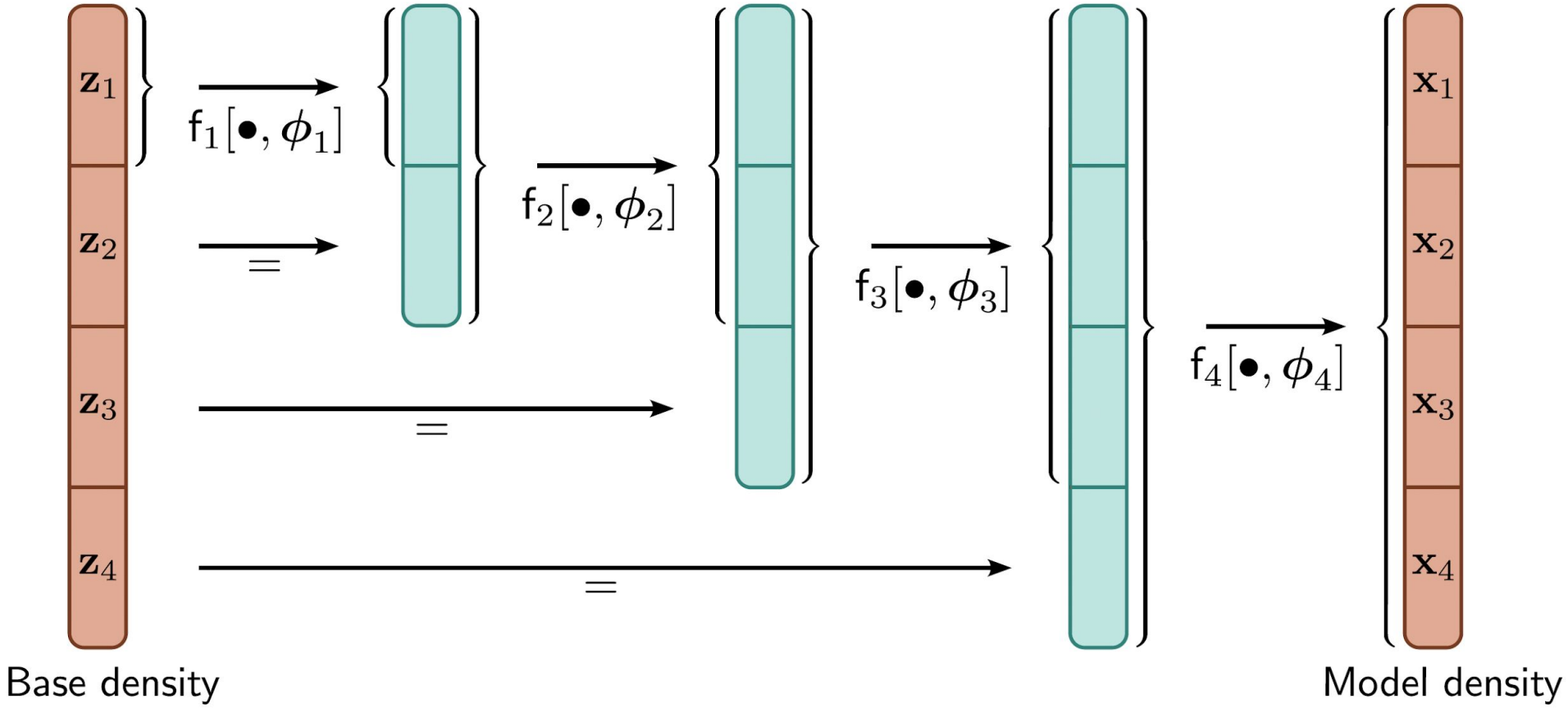

Figure 16.10 Multiscale flows. The latent space $\mathbf{z}$ must be the same size as the model density in normalizing flows. However, it can be partitioned into several components, which can be gradually introduced at different layers. This makes both density estimation and sampling faster. For the inverse process, the black arrows are reversed, and the last part of each block skips the remaining processing. For example, $\mathbf{f}_3^{-1}[\bullet, \boldsymbol{\phi}_3]$ only operates on the first three blocks, and the fourth block becomes $\mathbf{z}_4$ and is assessed against the base density.

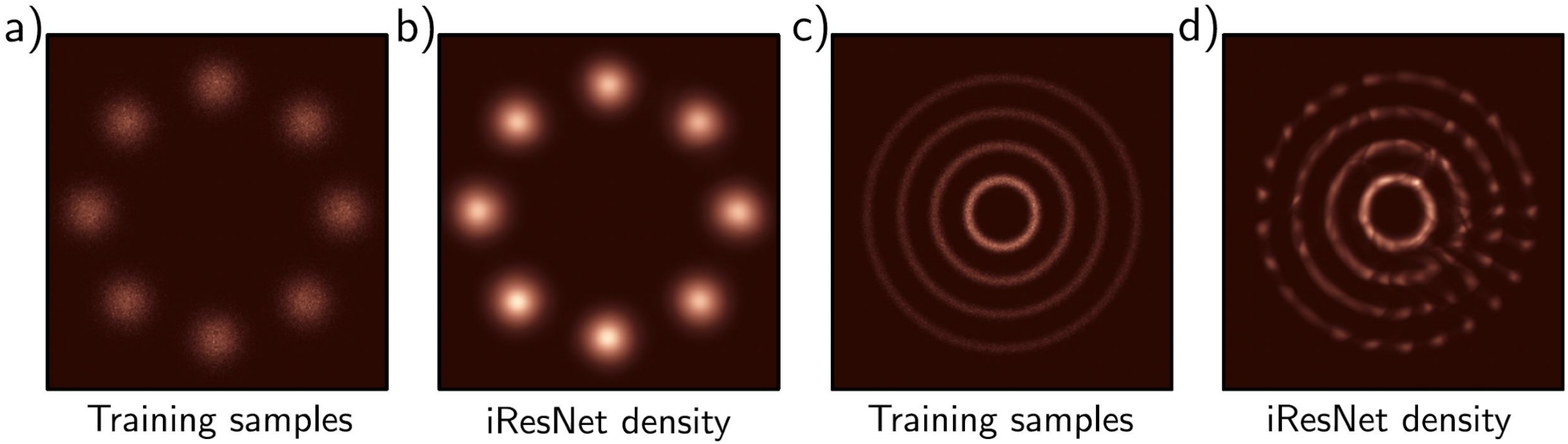

Figure 16.11 Modeling densities. a) Toy 2D data samples. b) Modeled density using iResNet. c–d) Second example. Adapted from Behrmann et al. (2019)

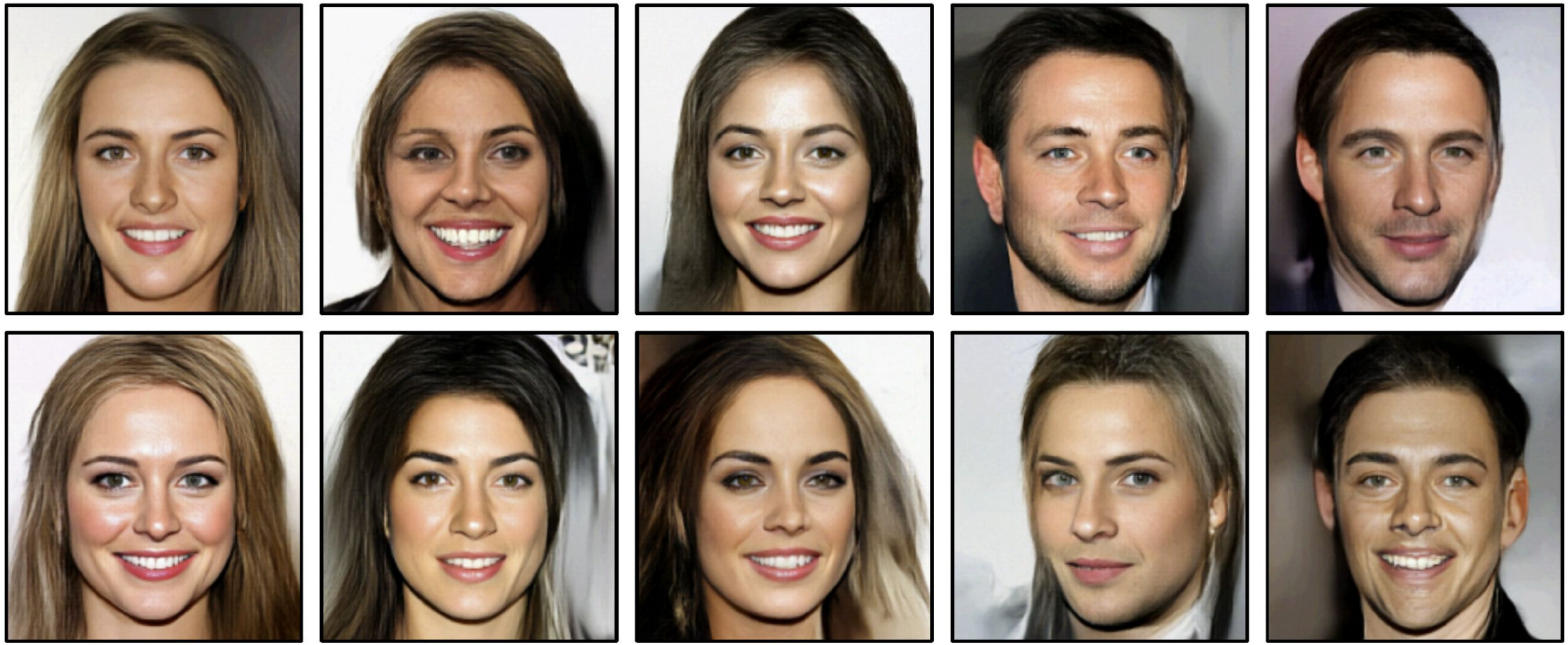

Figure 16.12 Samples from GLOW trained on the CelebA HQ dataset (Karras et al., 2018). The samples are of reasonable quality, although GANs and diffusion models produce superior results. Adapted from Kingma & Dhariwal (2018).

Figure 16.13 Interpolation using GLOW model. The left and right images are real people. The intermediate images were computed by projecting the real images to the latent space, interpolating, and then projecting the interpolated points back to image space. Adapted from Kingma & Dhariwal (2018).

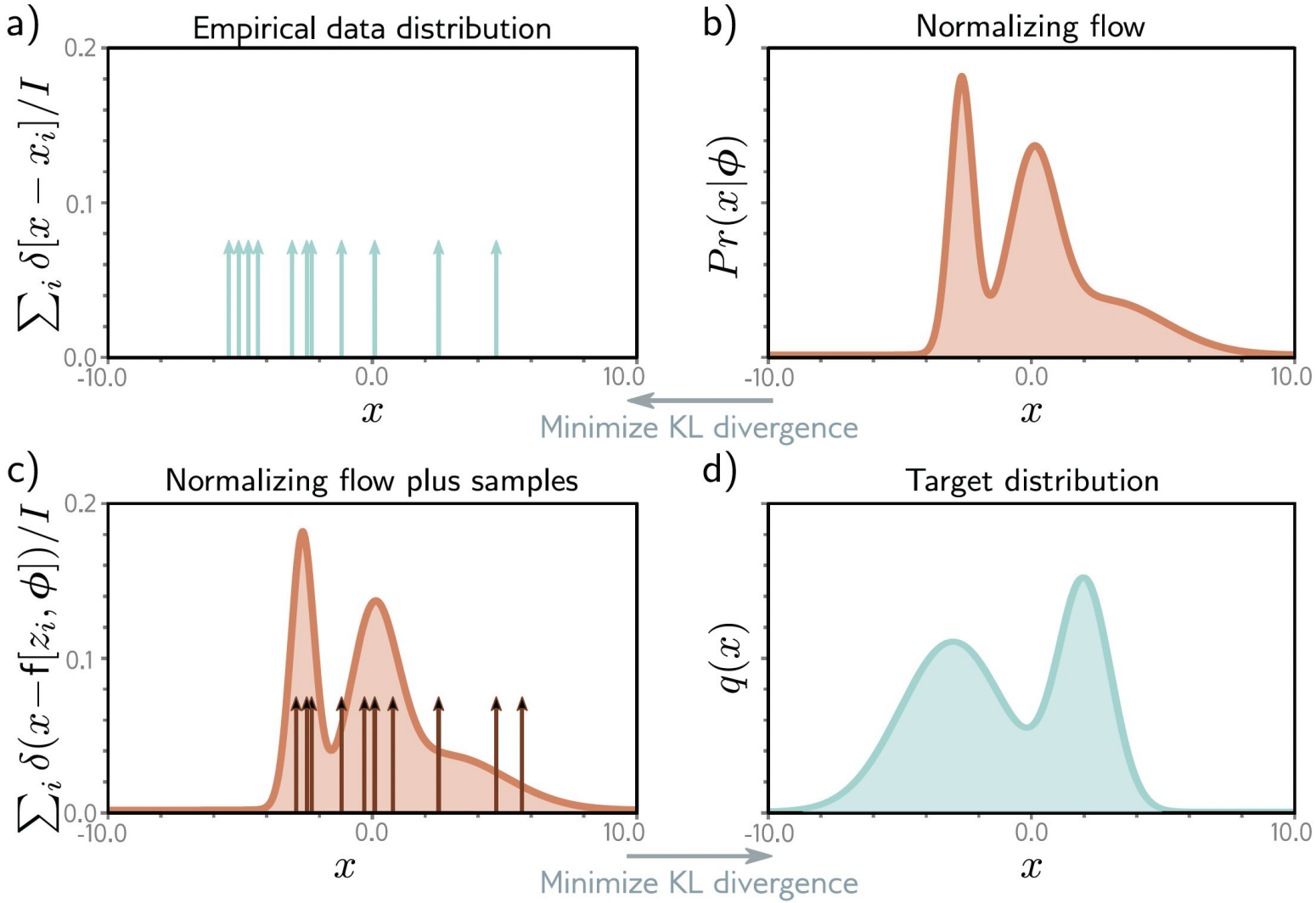

Figure 16.14 Approximating density models. a) Training data. b) Usually, we modify the flow model parameters to minimize the KL divergence from the training data to the flow model. This is equivalent to maximum likelihood fitting (section 5.7). c) Alternatively, we can modify the flow parameters $\boldsymbol{\phi}$ to minimize the KL divergence from the flow samples $x_i = \mathrm{f}[z_i, \boldsymbol{\phi}]$ to d) a target density.