Understanding Deep Learning

Chapter 15: Generative adversarial networks

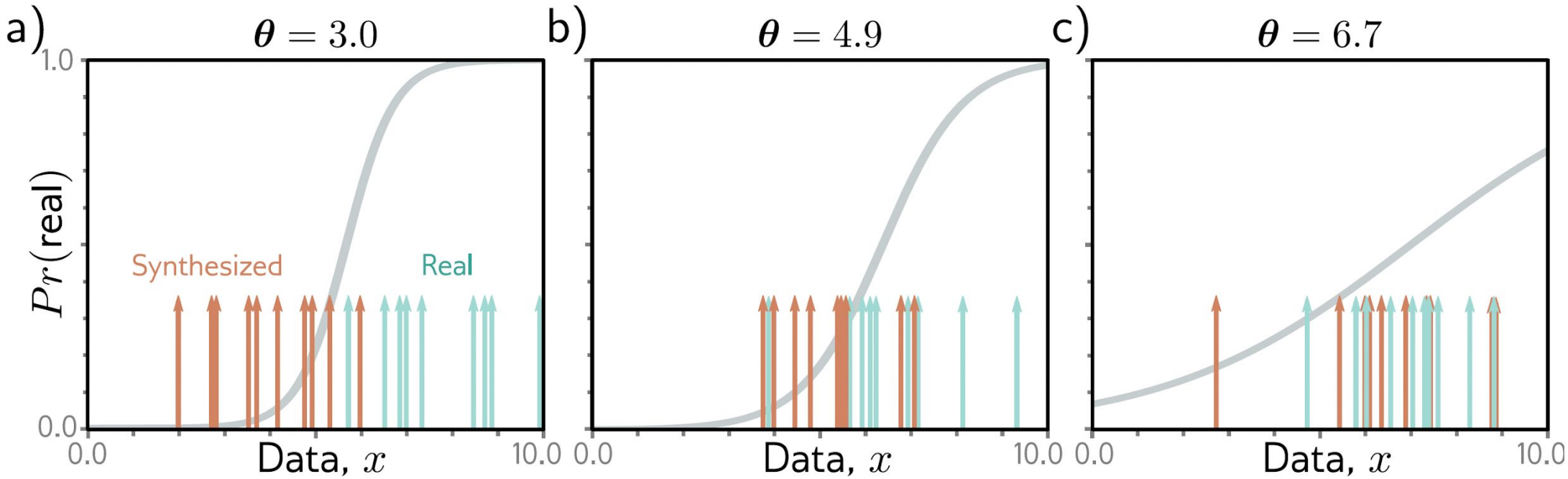


Figure 15.1 GAN mechanism. a) Given a parameterized function (a generator) that synthesizes samples (orange arrows) and a batch of real examples (cyan arrows), we train a discriminator to distinguish the real examples from the generated samples (sigmoid curve indicates the estimated probability that the data point is real). b) The generator is trained by modifying its parameters so that the discriminator becomes less confident the samples were synthetic (in this case, by moving the orange samples to the right). The discriminator is then updated. c) Alternating updates to the generator and discriminator cause the generated samples to become indistinguishable from real examples and the impetus to change the generator (i.e., the slope of the sigmoid function) to diminish.

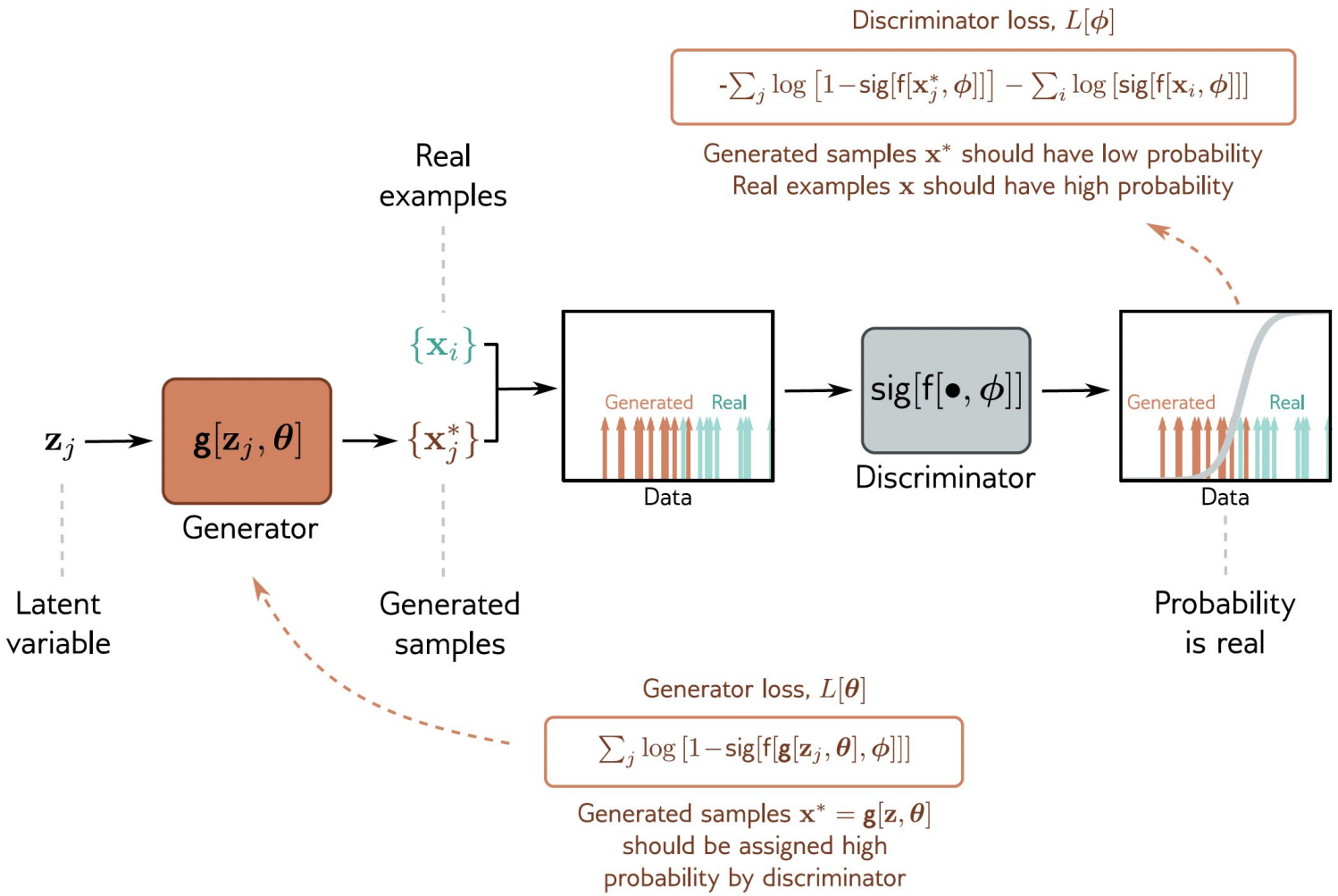


Figure 15.2 GAN loss functions. A latent variable $\mathbf{z}_j$ is drawn from the base distribution and passed through the generator to create a sample $\mathbf{x}^*$. A batch $\{\mathbf{x}_j^*\}$ of samples and a batch of real examples $\{\mathbf{x}_i\}$ are passed to the discriminator, which assigns a probability that each is real. The discriminator parameters $\boldsymbol{\phi}$ are modified to assigns high probability to the real examples and low probability to the generated samples. The generator parameters $\boldsymbol{\theta}$ are modified to "fool" the discriminator into assigning the generated samples a high probability.

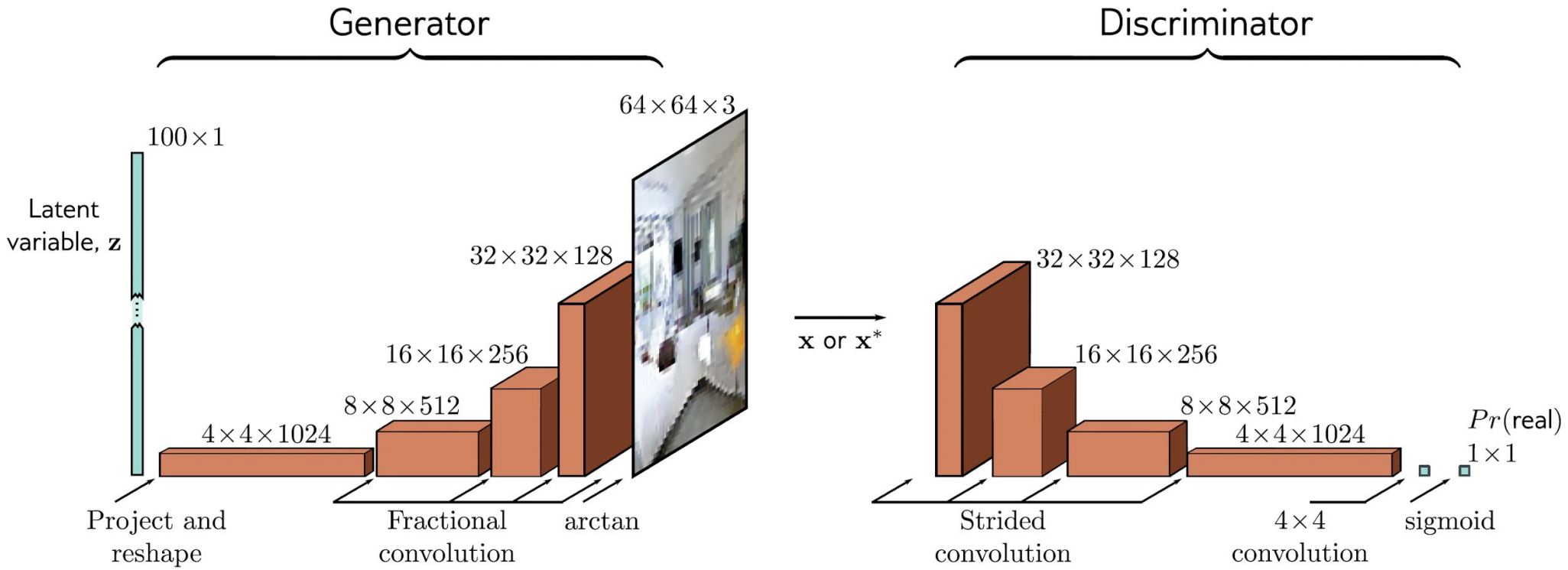


Figure 15.3 DCGAN architecture. In the generator, a 100D latent variable $\mathbf{z}$ is drawn from a uniform distribution and mapped by a linear transformation to a 4×4 representation with 1024 channels. This is then passed through a series of convolutional layers that gradually upsample the representation and decrease the number of channels. At the end is an arctan function that maps the $64\times64\times3$ representation to a fixed range so that it can represent an image. The discriminator consists of a standard convolutional net that classifies the input as either a real example or a generated sample.

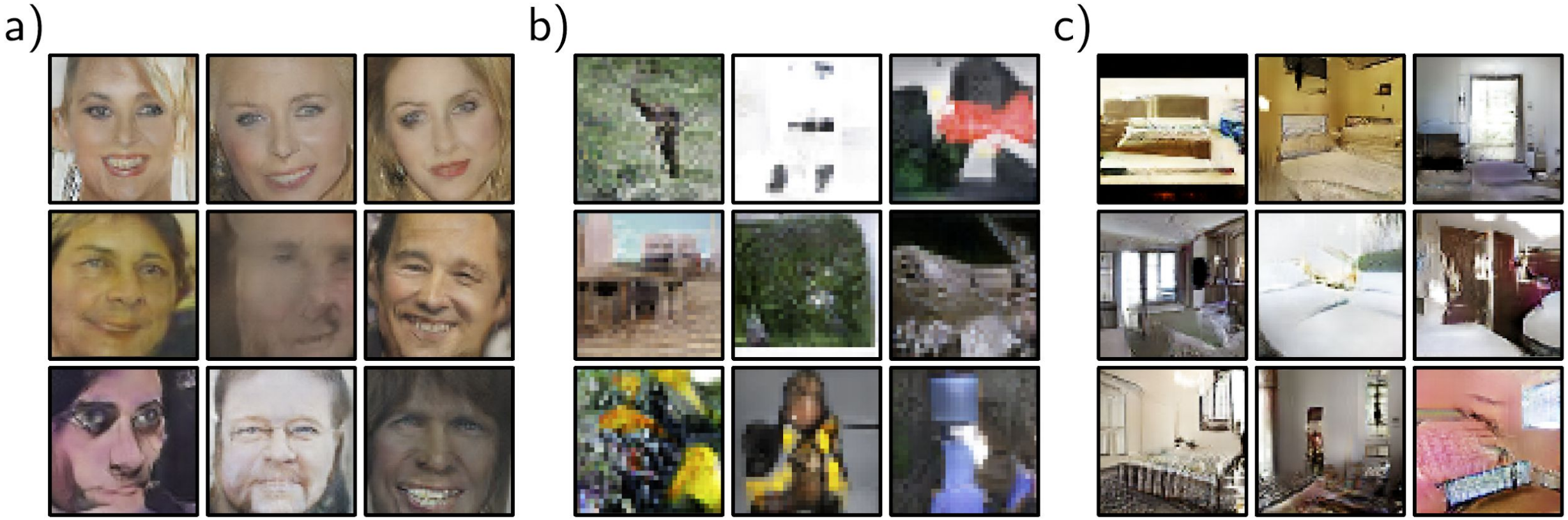


Figure 15.4 Synthesized images from the DCGAN model. a) Random samples drawn from DCGAN trained on a faces dataset. b) Random samples using the ImageNet database (see figure 10.15). c) Random samples drawn from the LSUN scene understanding dataset. Adapted from Radford et al. (2015).

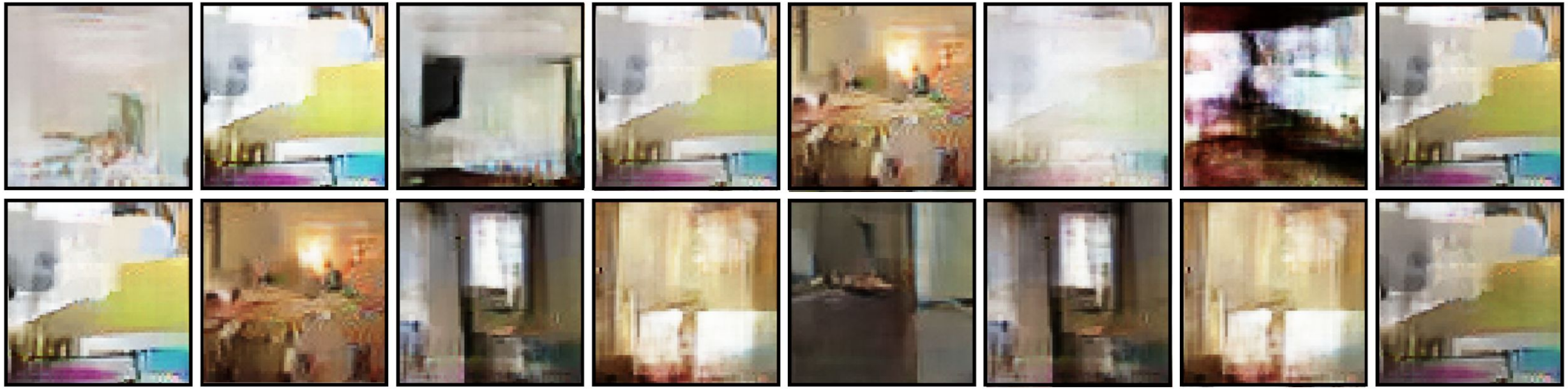
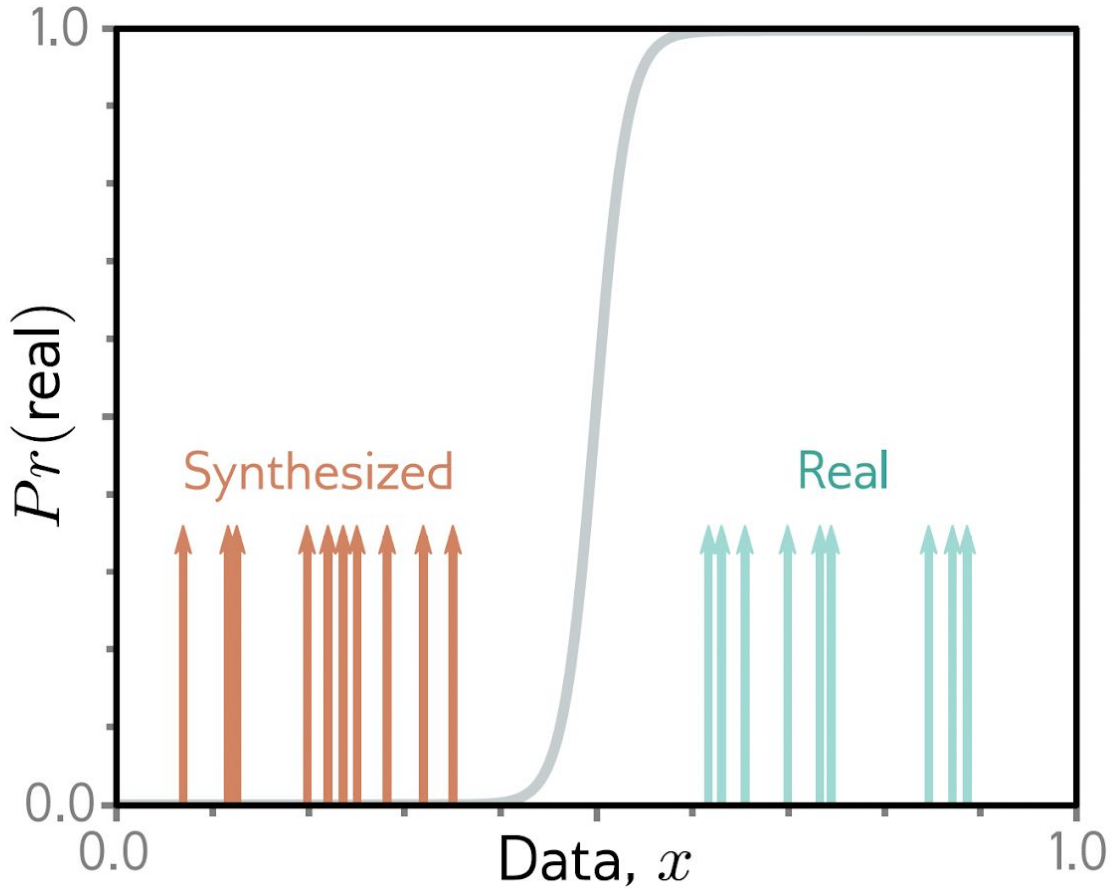


Figure 15.5 Mode collapse. Synthesized images from a GAN trained on the LSUN scene understanding dataset using an MLP generator with a similar number of parameters and layers to the DCGAN. The samples are low quality, and many are similar. Adapted from Arjovsky et al. (2017).

Figure 15.6 Problem with GAN loss function. If the generated samples (orange arrows) are easy to distinguish from the real examples (cyan arrows), then the discriminator (sigmoid) may have a very shallow slope at the positions of the samples; hence, the gradient to update the parameter of the generator may be tiny.



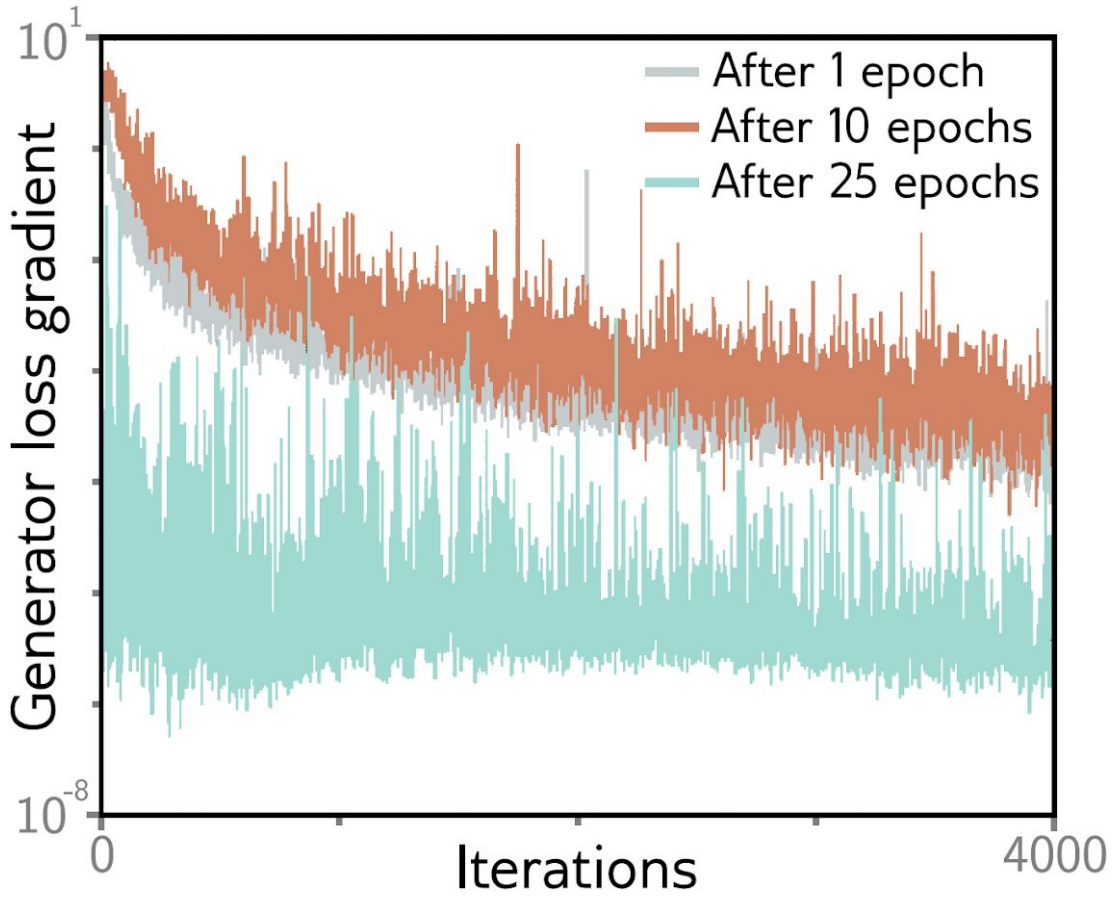


Figure 15.7 Vanishing gradients in the generator of a DCGAN. The generator is frozen after 1, 10, and 25 epochs, and the discriminator is trained further. The gradient of the generator decreases rapidly (note log scale); if the discriminator becomes too accurate, the gradients for the generator vanish. Adapted from Arjovsky & Bottou (2017).

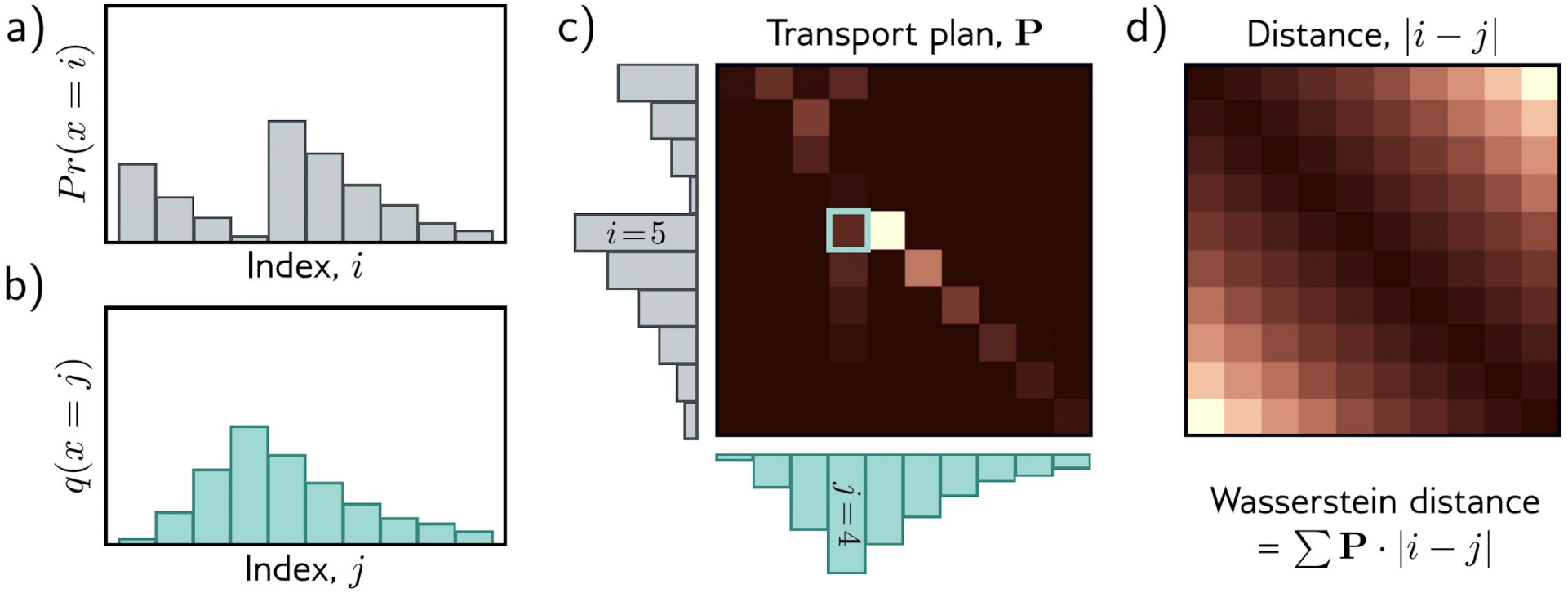


Figure 15.8 Wasserstein or earth mover's distance. a) Consider the discrete distribution $Pr(x=i)$. b) We wish to move the probability mass to create the target distribution $q(x=j)$. c) The transport plan $\mathbf{P}$ identifies how much mass will be moved from i to j. For example, the cyan highlighted square p_{54} indicates how much mass will be moved from $i=5$ to $j=4$. The elements of the transport plan must be non-negative, the sum over j must be $Pr(x=i)$, and the sum over i must be $q(x=j)$. Hence $\mathbf{P}$ is a joint probability distribution. d) The distance matrix between elements i and j. The optimal transport plan $\mathbf{P}$ minimizes the sum of the pointwise product of $\mathbf{P}$ and the distance matrix (termed the Wasserstein distance). Hence, the elements of $\mathbf{P}$ tend to lie close to the diagonal where the distance cost is lowest. Adapted from Hermann (2017).

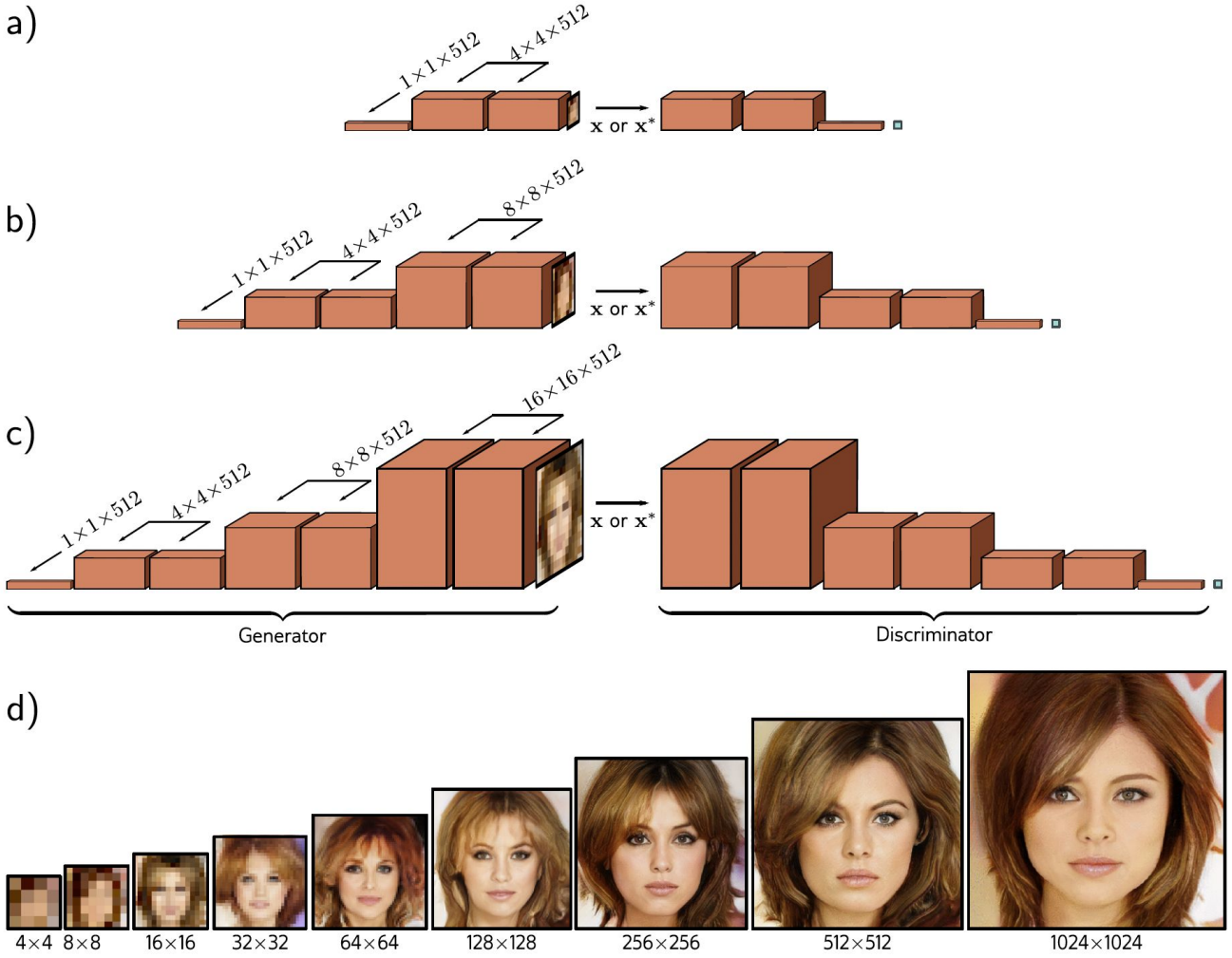


Figure 15.9 Progressive growing. a) The generator is initially trained to create very small (4×4) images, and the discriminator to identify if these images are synthesized or downsampled real images. b) After training at this low-resolution terminates, subsequent layers are added to the generator to generate (8×8) images. Similar layers are added to the discriminator to downsample back again. c) This process continues to create (16×16) images and so on. In this way, a GAN that produces very realistic high-resolution images can be trained. d) Images of increasing resolution generated at different stages from the same latent variable. Adapted from Wolf (2021), using method of Karras et al. (2018).

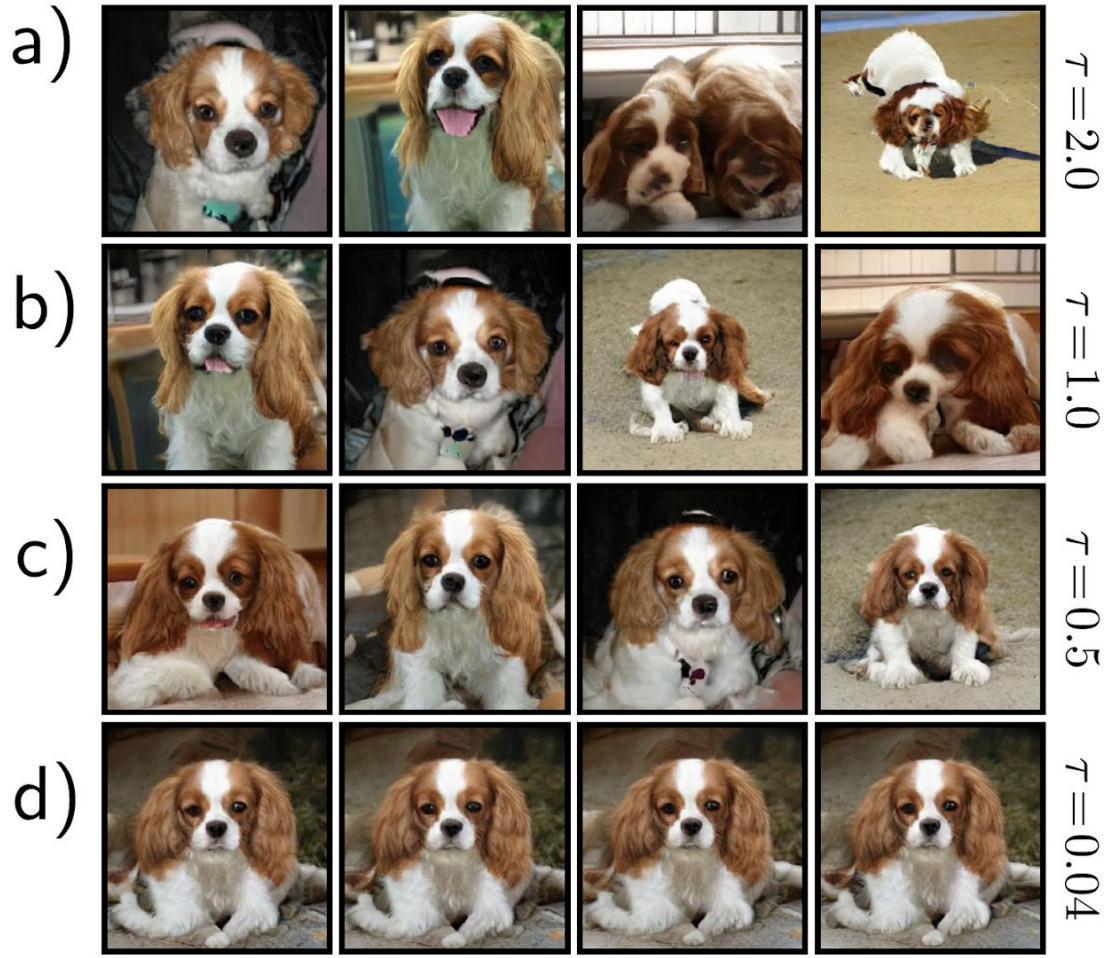


Figure 15.10 Truncation. The quality of GAN samples can be traded off against diversity by rejecting samples from the latent variable $\mathbf{z}$ that fall further than τ standard deviations from the mean. a) If this threshold is large ($\tau = 2.0$), the samples are visually varied but may have defects. b–c) As this threshold is decreased, the average visual quality improves, but the diversity decreases. d) With a very small threshold, the samples look almost identical. By judiciously choosing this threshold, it's possible to increase the average quality of GAN results. Adapted from Brock et al. (2019).

Figure 15.11 Progressive growing. This method generates realistic images of faces when trained on the CELEBA-HQ dataset and more complex, variable objects when trained on LSUN categories. Adapted from Karras et al. (2018).

Figure 15.12 Traversing latent space of progressive GAN trained on LSUN cars. Moving in the latent space produces car images that change smoothly. This usually only works for short trajectories; eventually, the latent variable moves to somewhere that produces unrealistic images. Adapted from Karras et al. (2018).

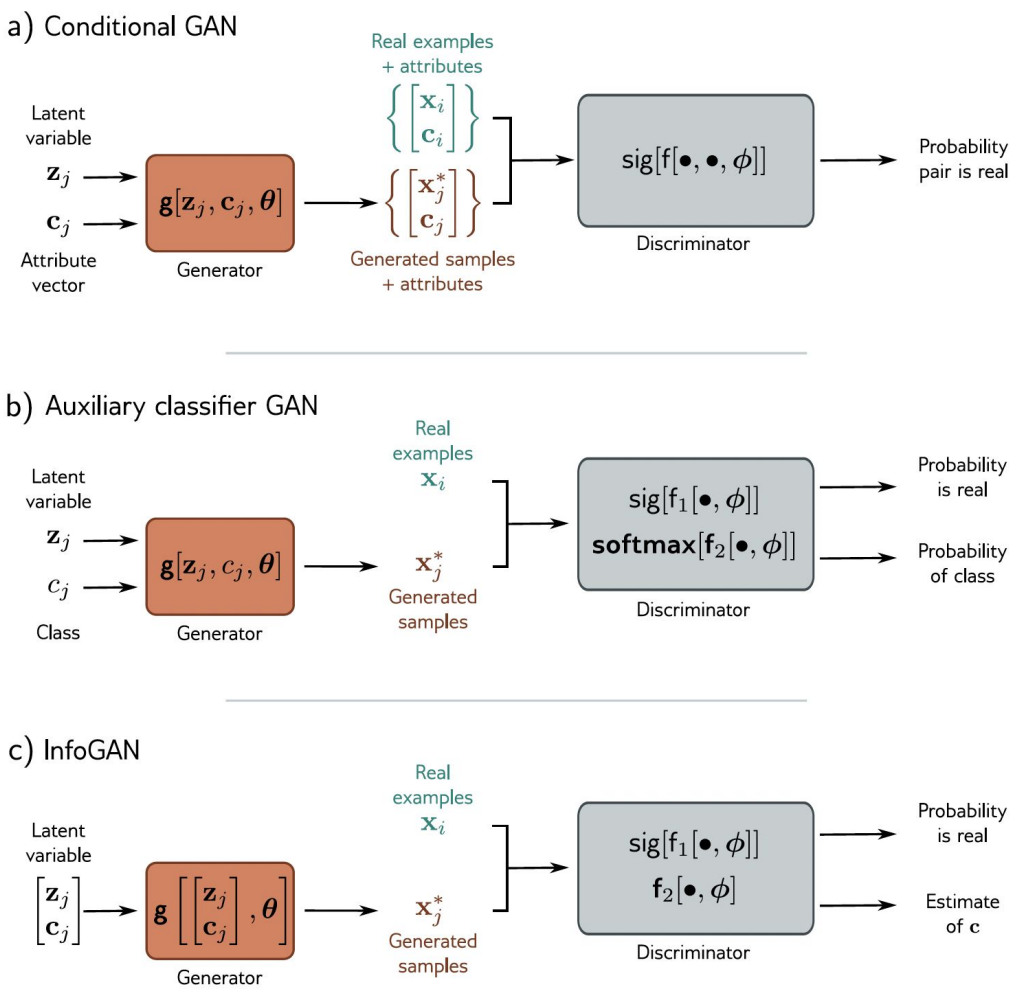


Figure 15.13 Conditional generation. a) The generator of the conditional GAN also receives an attribute vector $\mathbf{c}$ describing some aspect of the image. As usual, the discriminator receives either a real example or a generated sample, but now it also receives the attribute vector; this encourages the samples both to be realistic and compatible with the attribute. b) The generator of the auxiliary classifier GAN (ACGAN) takes a discrete attribute variable. The discriminator must both (i) determine if its input is real or synthetic and (ii) identify the class correctly. c) The InfoGAN splits the latent variable into noise $\mathbf{z}$ and unspecified random attributes $\mathbf{c}$. The discriminator must distinguish if its input is real and also reconstruct these attributes. In practice, this means that the variables $\mathbf{c}$ correspond to salient aspects of the data with real-world interpretations (i.e., the latent space is *disentangled*).

Figure 15.14 Auxiliary classifier GAN. The generator takes a class label as well as the latent vector. The discriminator must both identify if the data point is real *and* predict the class label. This model was trained on ten ImageNet classes. Left to right: generated examples of monarch butterflies, goldfinches, daisies, redshanks, and gray whales. Adapted from Odena et al. (2017).

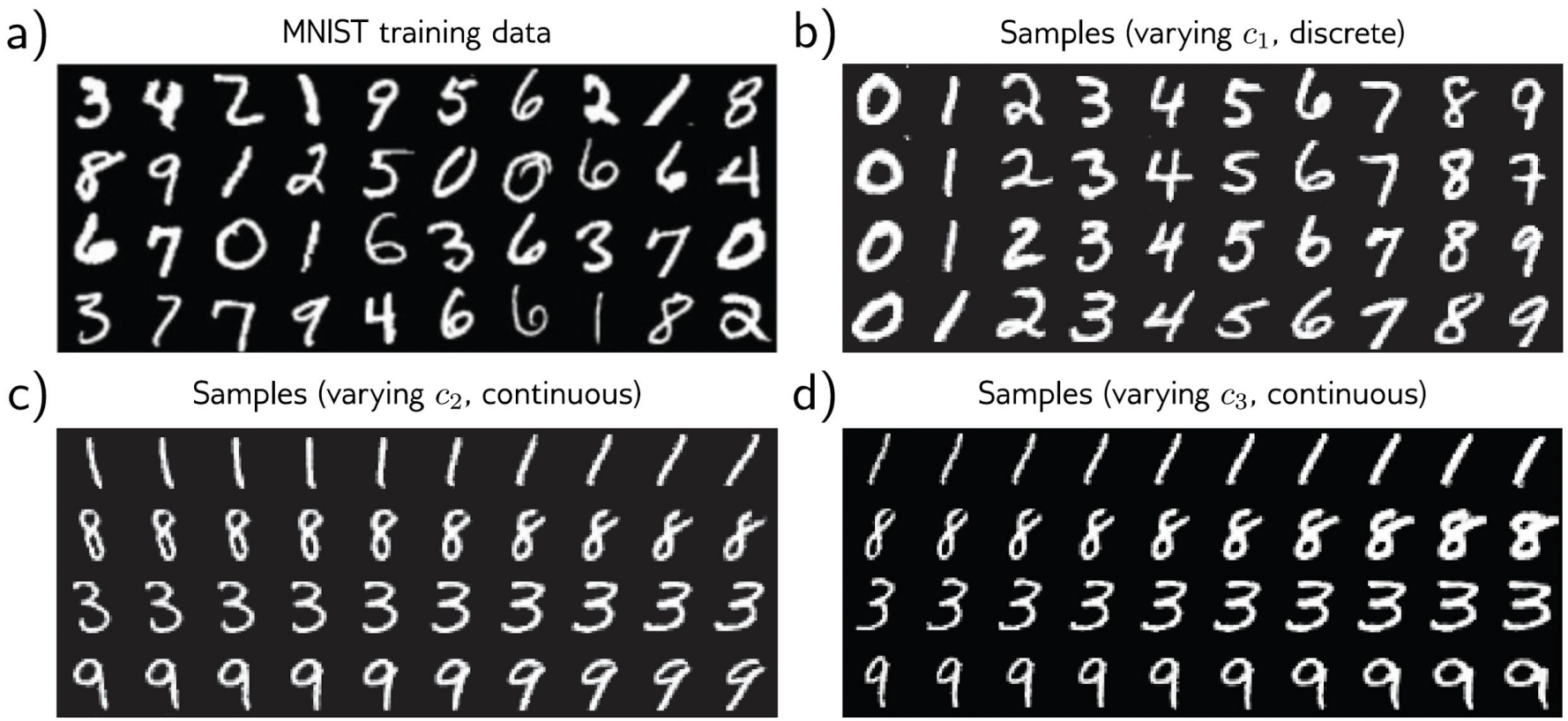


Figure 15.15 InfoGAN for MNIST. a) Training examples from the MNIST database, which consists of 28×28 pixel images of handwritten digits. b) The first attribute c_1 is categorical with 10 categories; each column shows samples generated with one of these categories. The InfoGAN recovers the ten digits. The attribute vectors c_2 and c_3 are continuous. c) Moving from left to right, each column represents a different value of c_2 while keeping the other latent variables constant. This attribute seems to correspond to the orientation of the character. d) The third attribute seems to correspond to the thickness of the stroke. Adapted from Chen et al. (2016b).

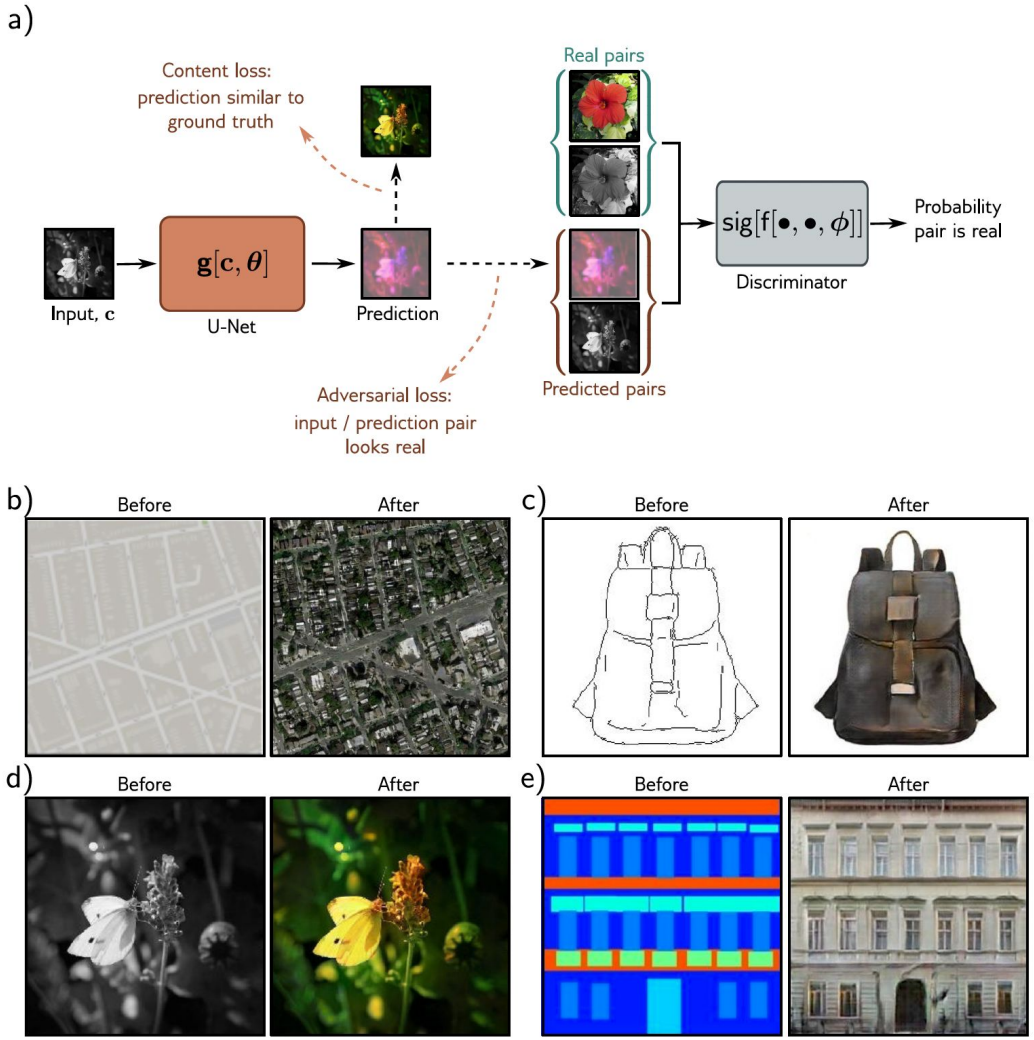


Figure 15.16 Pix2Pix model. a) The model translates an input image to a prediction in a different style using a U-Net (see figure 11.10). In this case, it maps a grayscale image to a plausibly colored version. The U-Net is trained with two losses. First, the content loss encourages the output image to have a similar structure to the input image. Second, the adversarial loss encourages the grayscale/color image pair to be indistinguishable from a real pair in each local region of these images. This framework can be adapted to many tasks, including b) translating maps to satellite imagery, c) converting sketches of bags to photorealistic examples, d) colorization, and e) converting label maps to photorealistic building facades. Adapted from Isola et al. (2017).

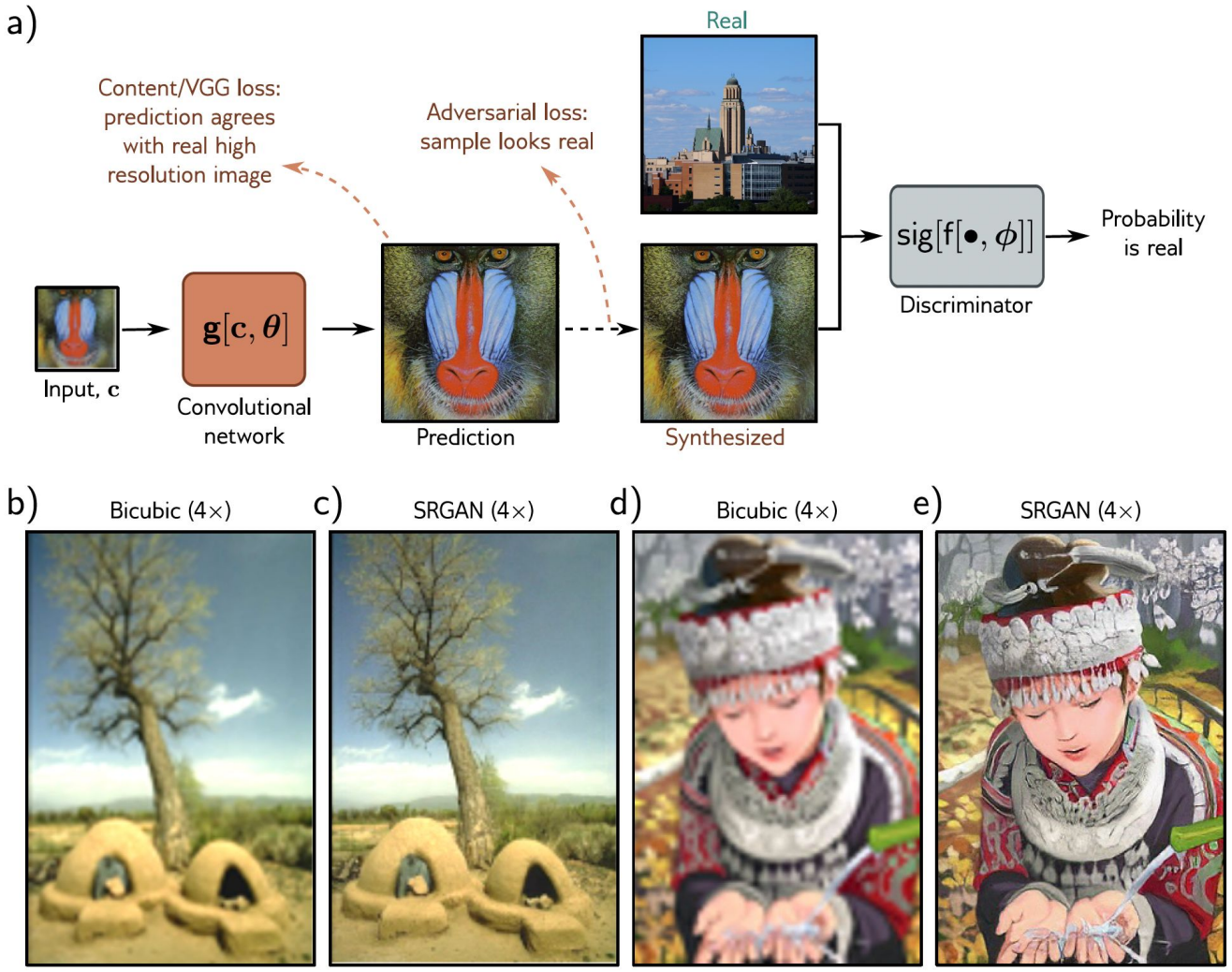


Figure 15.17 Super-resolution generative adversarial network (SRGAN). a) A convolutional network with residual connections is trained to increase the resolution of images by a factor of four. The model has losses that encourage the content to be close to the true high-resolution image. However, it also includes an adversarial loss, which penalizes results that can be distinguished from real high-resolution images. b) Upsampled image using bicubic interpolation. c) Upsampled image using SRGAN. d) Upsampled image using bicubic interpolation. e) Upsampled image using SRGAN. Adapted from Ledig et al. (2017).

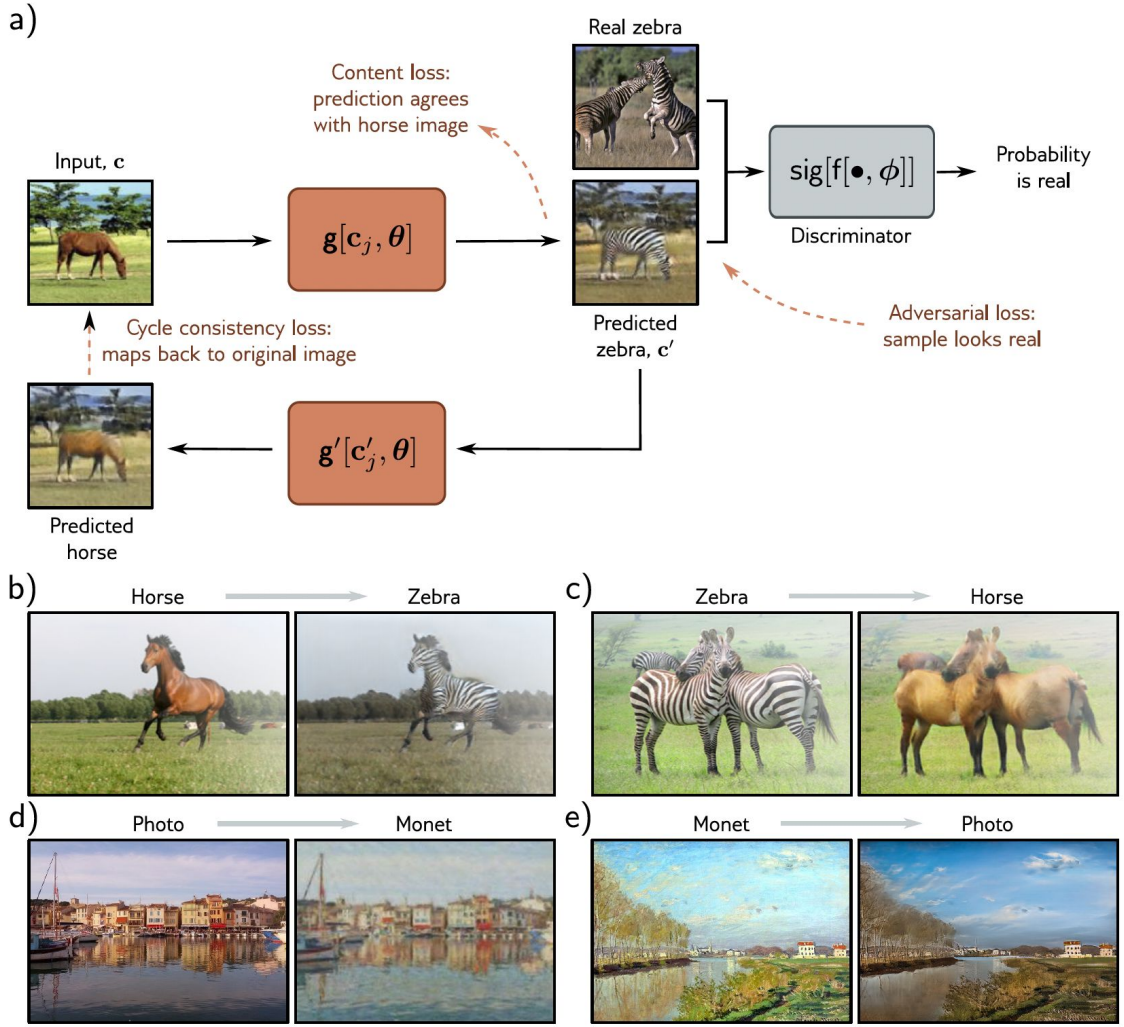


Figure 15.18 CycleGAN. Two models are trained simultaneously. The first $\mathbf{c}' = \mathbf{g}[\mathbf{c}_j, \boldsymbol{\theta}]$ translates from an image $\mathbf{c}$ in the first style (horse) to an image $\mathbf{c}'$ in the second style (zebra). The second model $\mathbf{c} = \mathbf{g}'[\mathbf{c}', \boldsymbol{\theta}]$ learns the opposite mapping. The cycle consistency loss penalizes both models if they cannot successfully convert an image to the other domain and back to the original. In addition, two adversarial losses encourage the translated images to look like realistic examples of the target domain (shown here for zebra only). Two content losses encourage the details and layout of the images before and after each mapping to be similar (i.e., the zebra is in the same position and pose that the horse was and against the same background and vice versa). Adapted from Zhu et al. (2017).

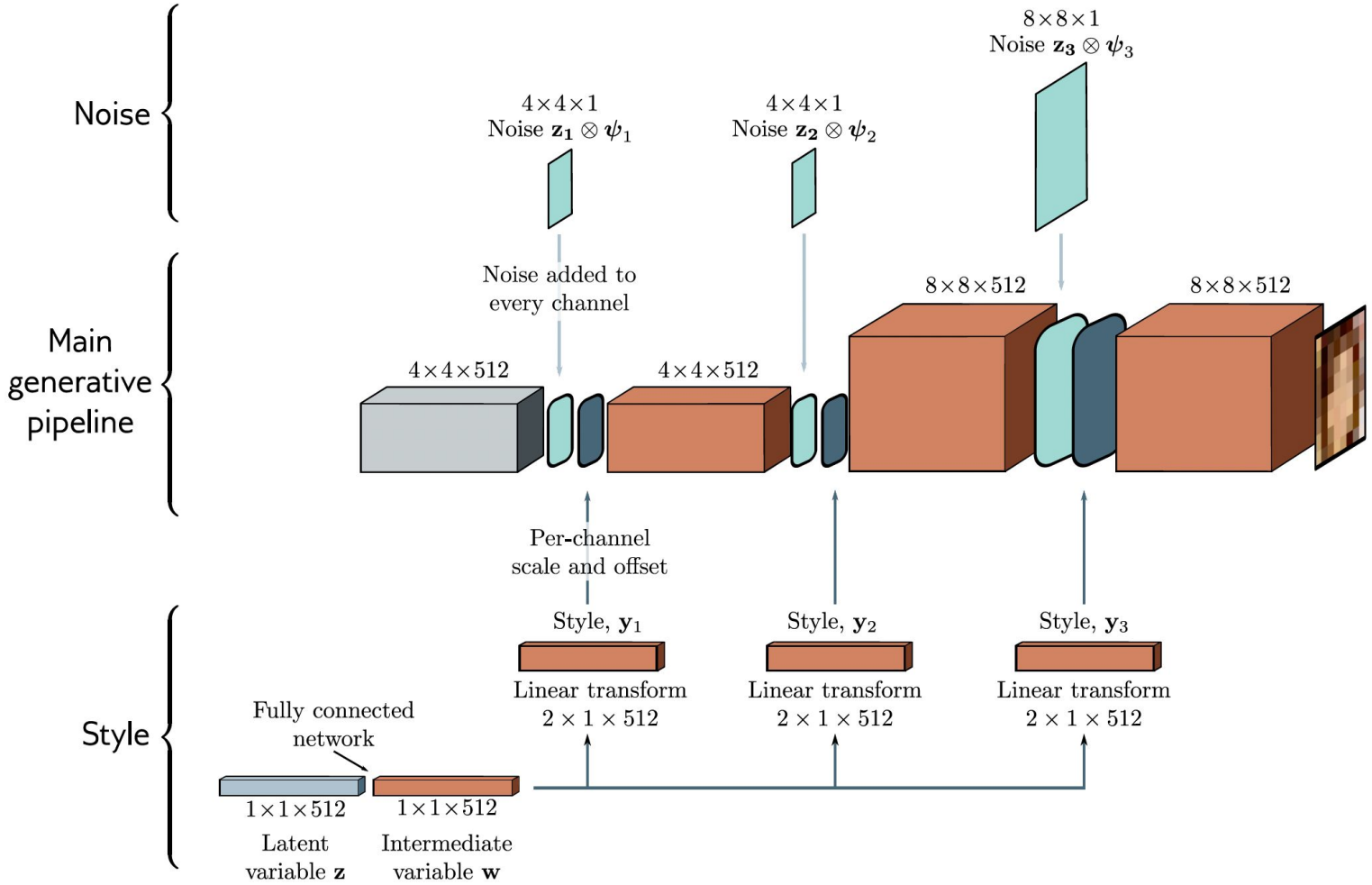


Figure 15.19 StyleGAN. The main pipeline (center row) starts with a constant learned representation (gray box). This is passed through a series of convolutional layers and gradually upsampled to create the output. Noise (top row) is added at different scales by periodically adding Gaussian variables $\mathbf{z}_\bullet$ with per-channel scaling $\boldsymbol{\psi}_\bullet$. The Gaussian style variable $\mathbf{z}$ is passed through a fully connected network to create intermediate variable $\mathbf{w}$ (bottom row). This is used to set the mean and variance of each channel at various points in the pipeline.

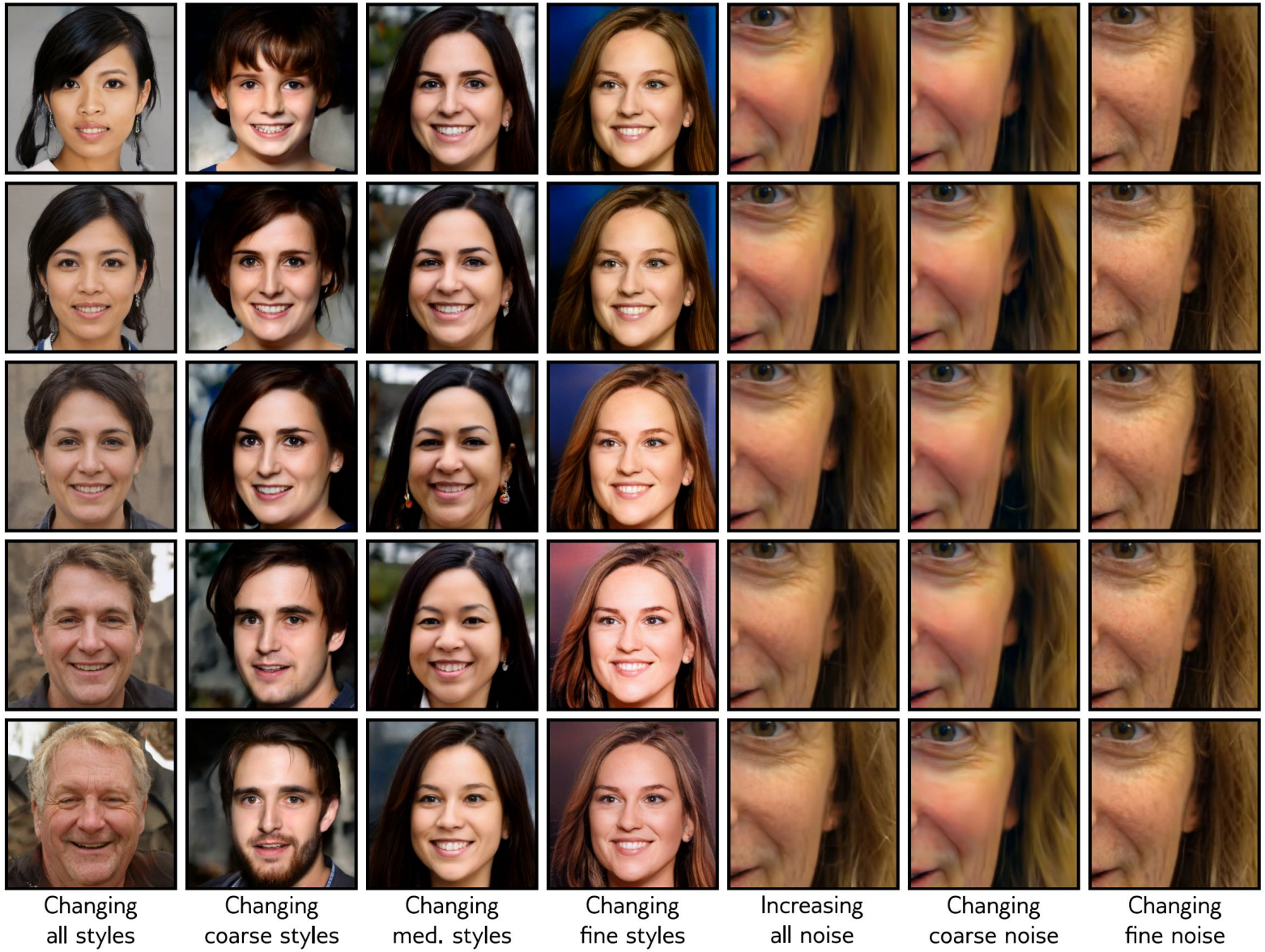


Figure 15.20 StyleGAN results. First four columns show systematic changes in style at various scales. Fifth column shows the effect of increasing noise magnitude. Last two columns show different noise vectors at two different scales.