KiSTi 한국과학기술정보연구원 Korea Institute of Science and Technology Information

2022

Advanced Topic in Research Data-centric Deep Learning

Lec 12: AutoEncoder

hsyi@kisti.re.kr

Hongsuk Yi (이홍석)

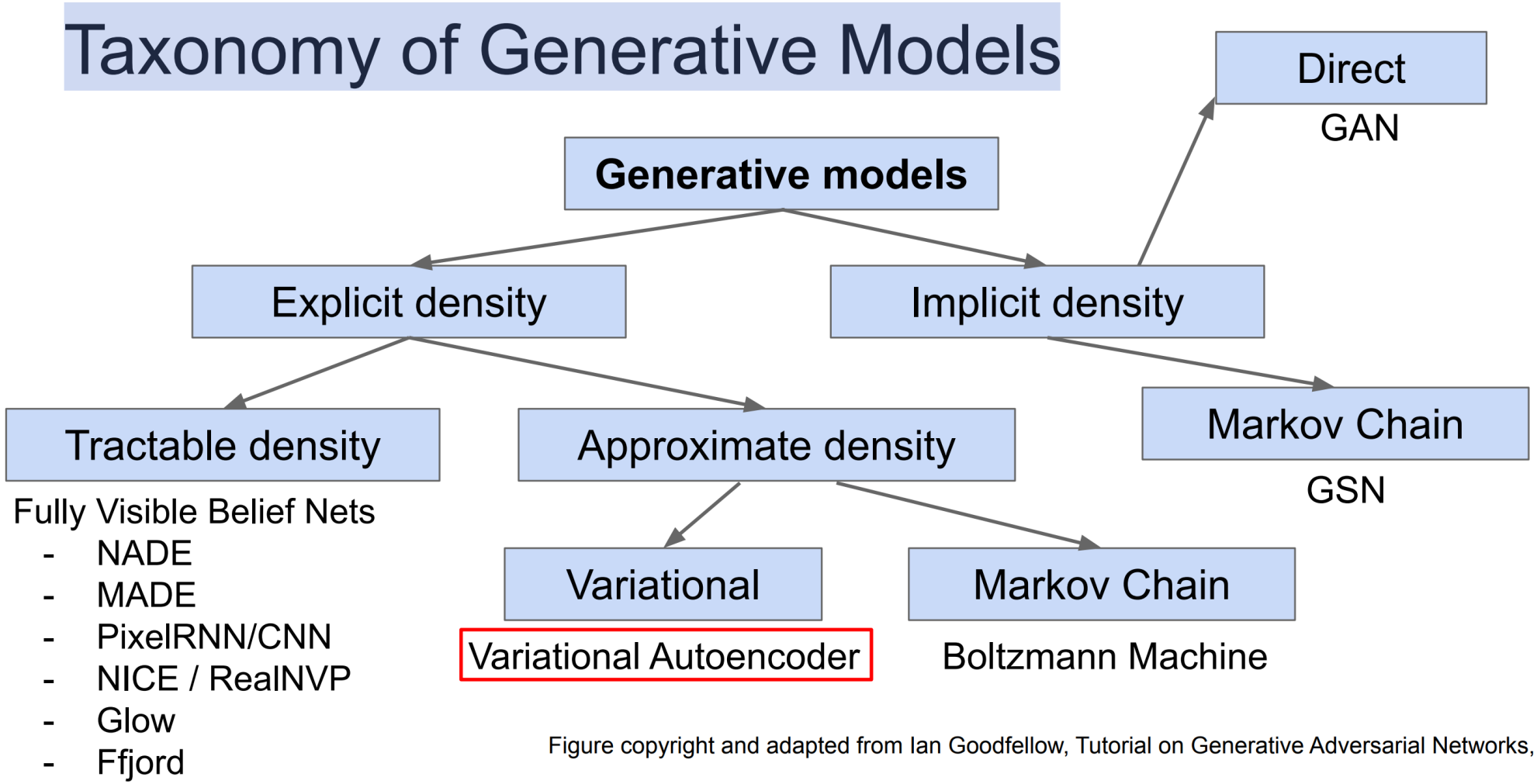


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Supervised vs unsupervised learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn function to map
$x \rightarrow y$

Examples: Classification, regression, object detection, semantic segmentation, etc.

Unsupervised Learning

Data: x
x is data, no labels!

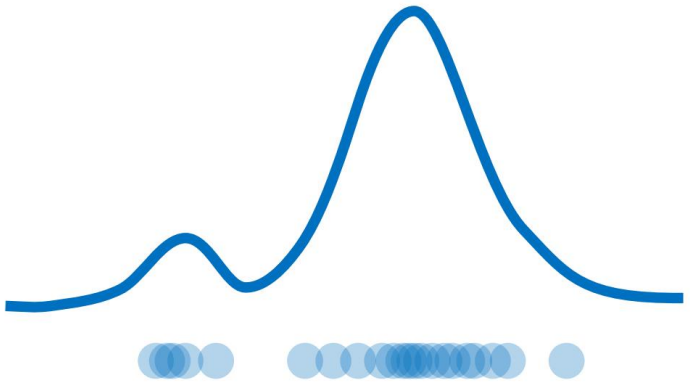
Goal: Learn some *hidden* or *underlying structure* of the data

Examples: Clustering, feature or dimensionality reduction, etc.

Generative modeling

- **Goal:**
  - ✓ Take as input training samples from some distribution and learn a model that represents that distribution

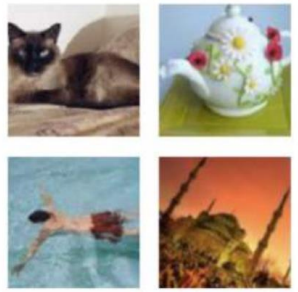
Density Estimation



Sample Generation

Input samples

Training data ~ $P_{data}(x)$



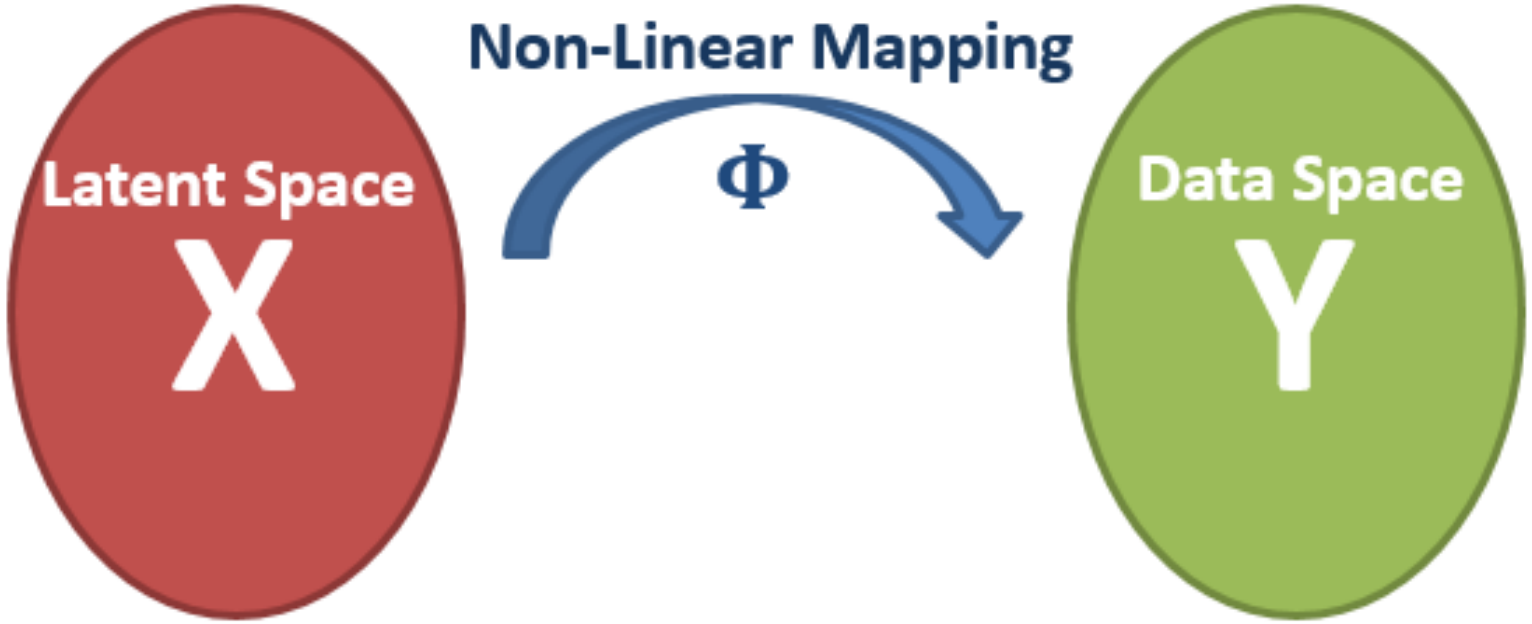
Generated samples

Generated ~ $P_{model}(x)$

How can we learn $P_{model}(x)$ similar to $P_{data}(x)$?

Latent variable models

- Autoencoders and Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)



What is a latent variable?

- Can we learn the true explanatory factors, e.g. latent variables, from only observed data?

Myth of the Cave

Autoencoders

Representation Learning and Generative Learning Using Autoencoders

Autoencoders

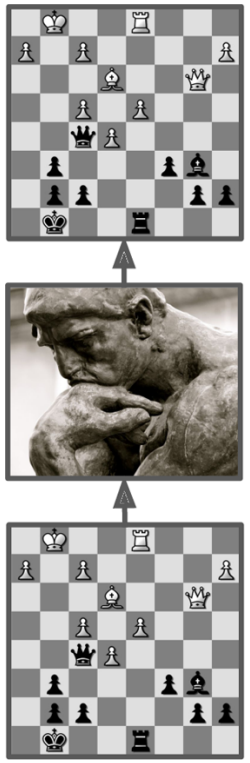
- Autoencoders are artificial neural networks capable of learning dense representations of the input data,
  - called latent representations or codings, without any supervision
- These codings typically have a much lower dimensionality than the input data,
  - making autoencoders useful for dimensionality reduction, especially for visualization purposes.
- Autoencoders also act as feature detectors,
  - and they can be used for unsupervised pretraining of deep neural networks
- Lastly, some autoencoders are generative models:
  - they are capable of randomly generating new data that looks very similar to the training data.

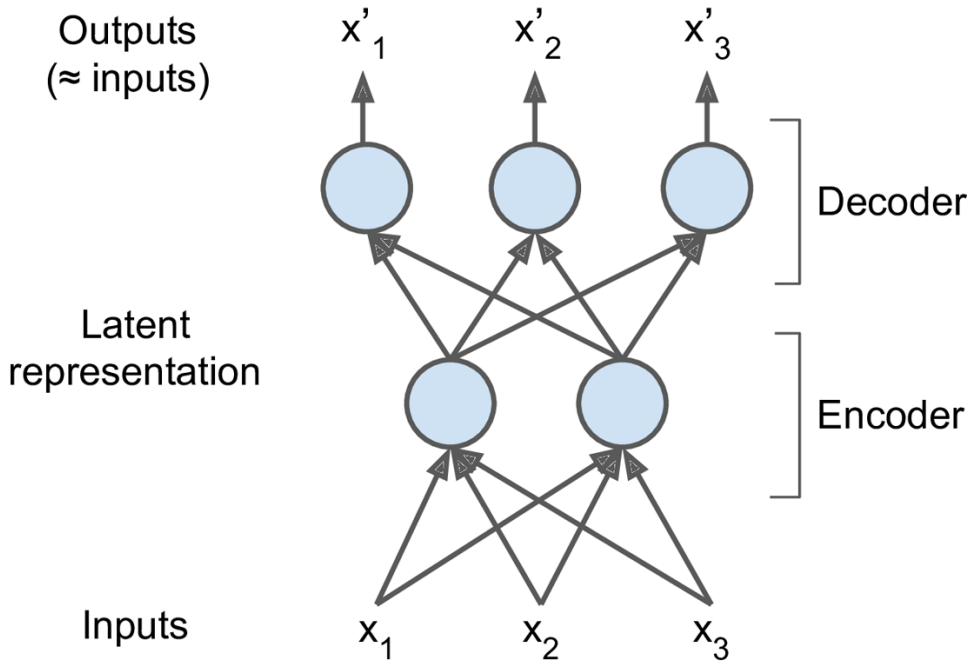
Efficient Data Representations

- Which of the following number sequences do you find the easiest to memorize?
  - ✓40, 27, 25, 36, 81, 57, 10, 73, 19, 68
  - ✓50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14
    - you only need to remember the pattern (i.e., decreasing even numbers) and the starting and ending numbers (i.e., 50 and 14)

Autoencoder

- An autoencoder is always composed of two parts:
  - an encoder (or recognition network) that converts the inputs to a latent representation,
  - a decoder (or generative network) that converts the internal representation to the outputs
- The chess memory experiment (left) and a simple autoencoder (right)





Undercomplete Linear Autoencoder

- Because the internal representation has a lower dimensionality than the input data (it is 2D instead of 3D),
  - the autoencoder is said to be undercomplete.

✓ If the autoencoder uses only linear activations and the cost function is the mean squared error (MSE), then it ends up performing Principal Component Analysis (PCA)

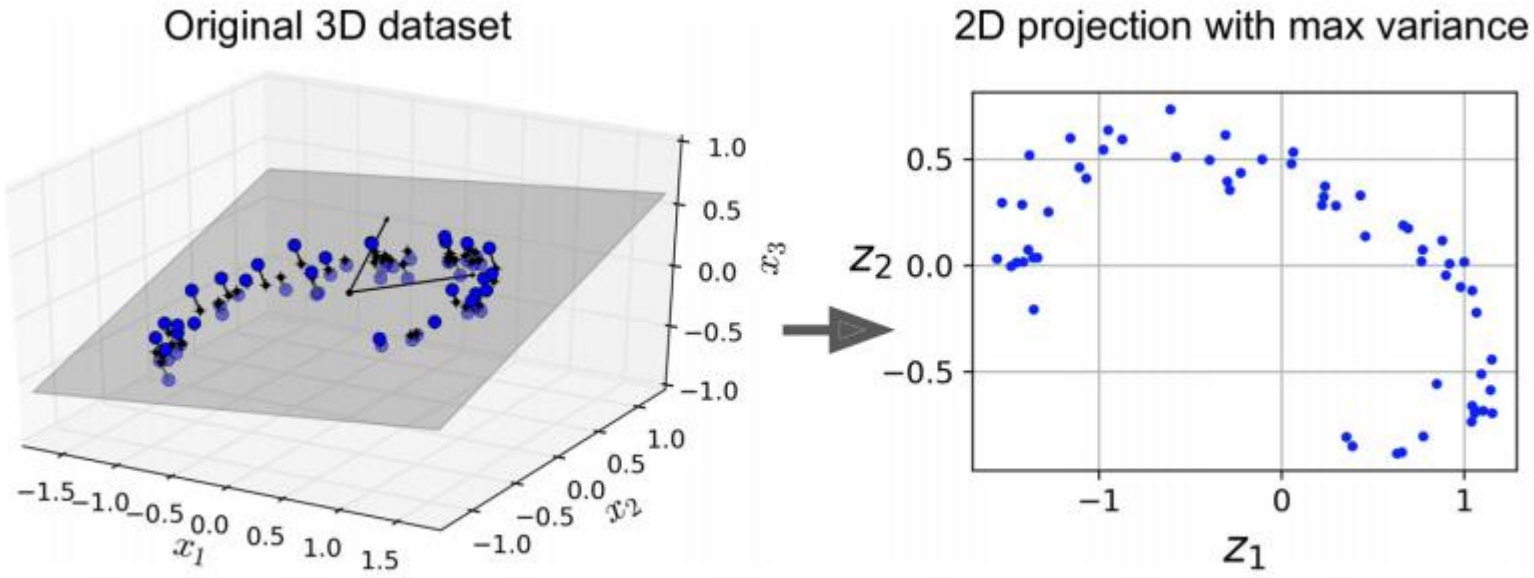


Figure 17-2. PCA performed by an undercomplete linear autoencoder

Stacked Autoencoders

- stacked autoencoders can have multiple hidden layers
  - The architecture of a stacked autoencoder is typically symmetrical with regard to the central hidden layer (the coding layer).

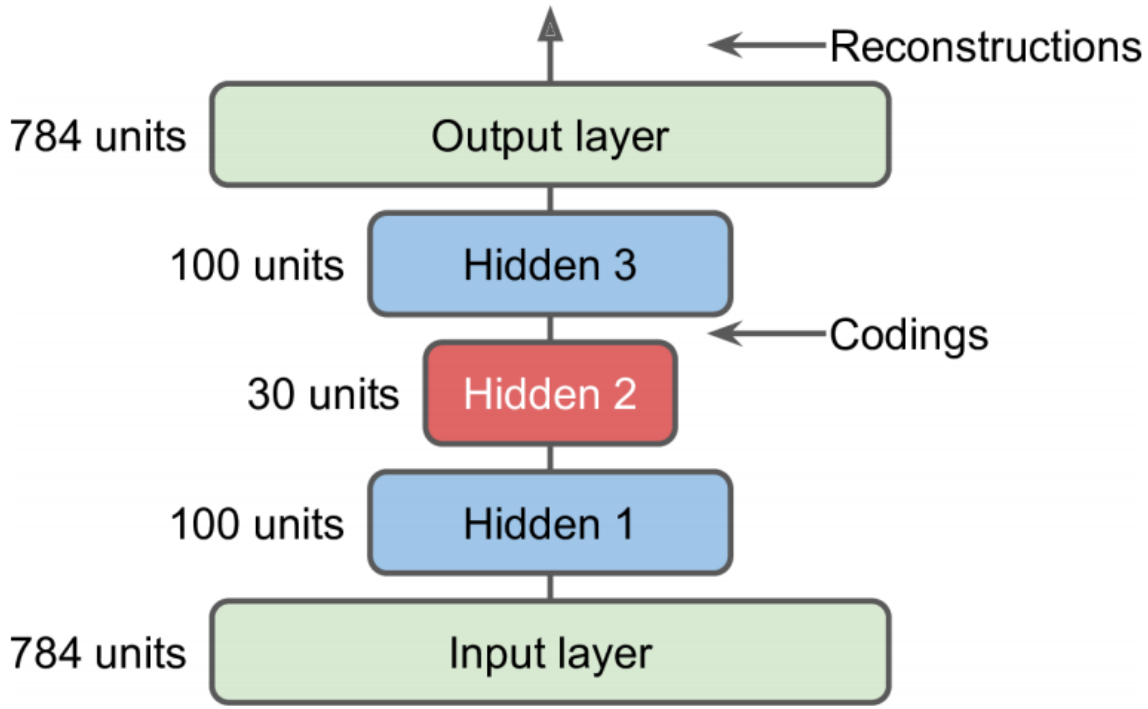


Figure 17-3. Stacked autoencoder

Unsupervised Pretraining Using Stacked Autoencoders

- A stacked autoencoder to perform unsupervised pretraining
  - if you really don't have much labeled training data, you may want to freeze the pretrained layers (at least the lower ones)

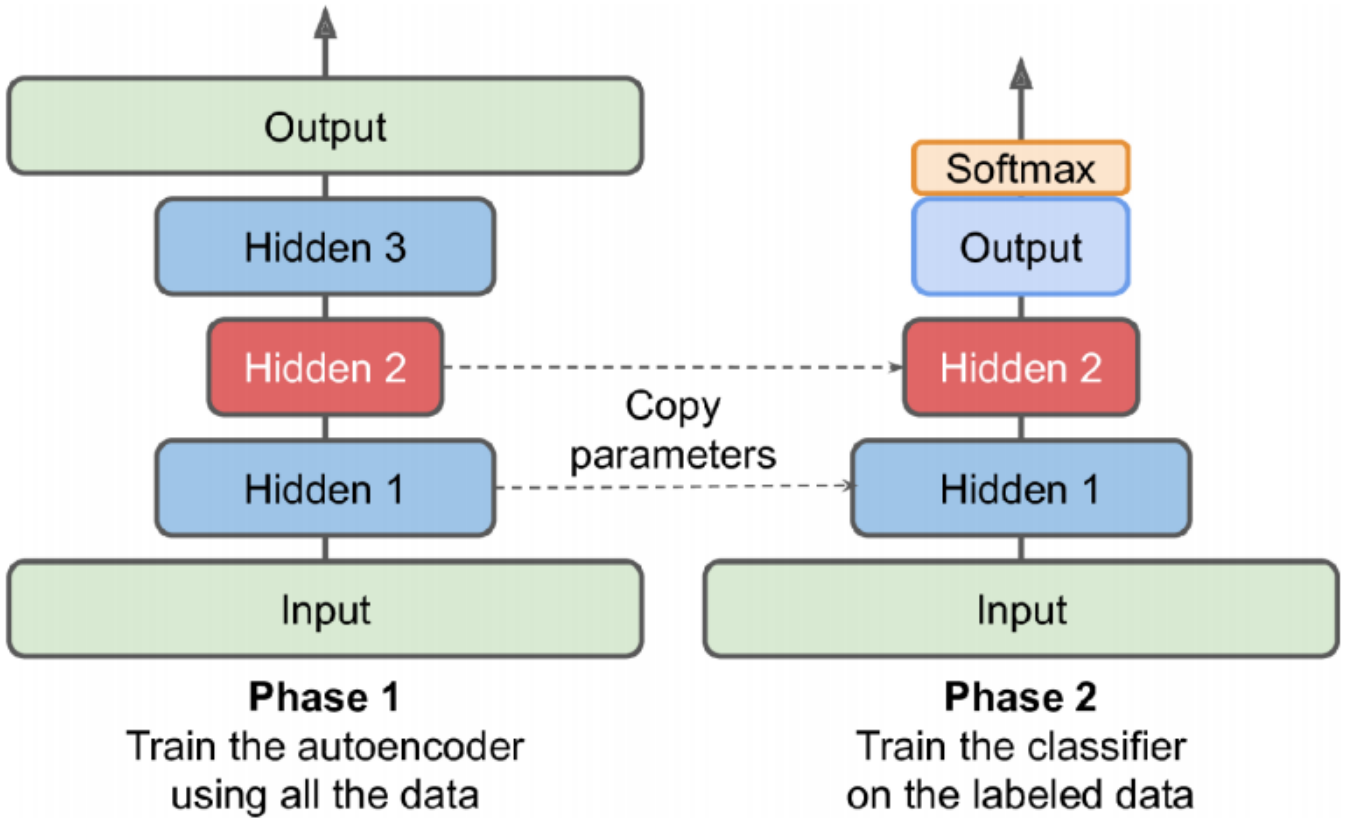
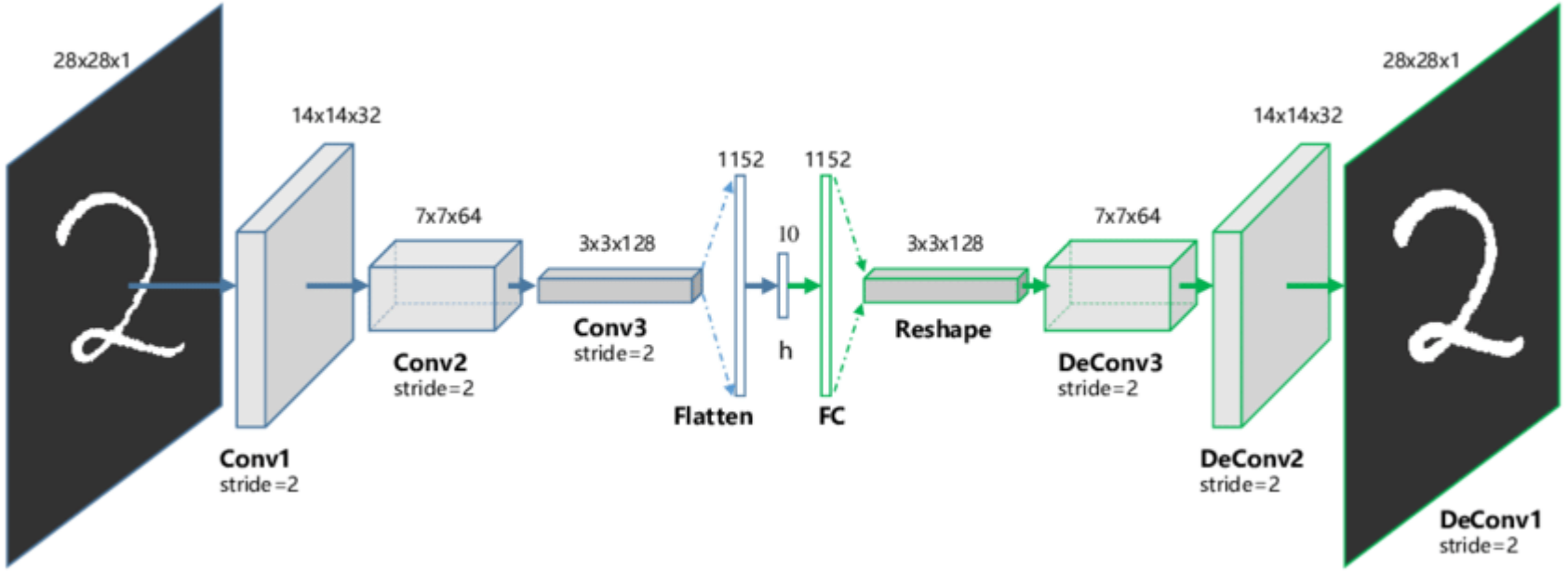
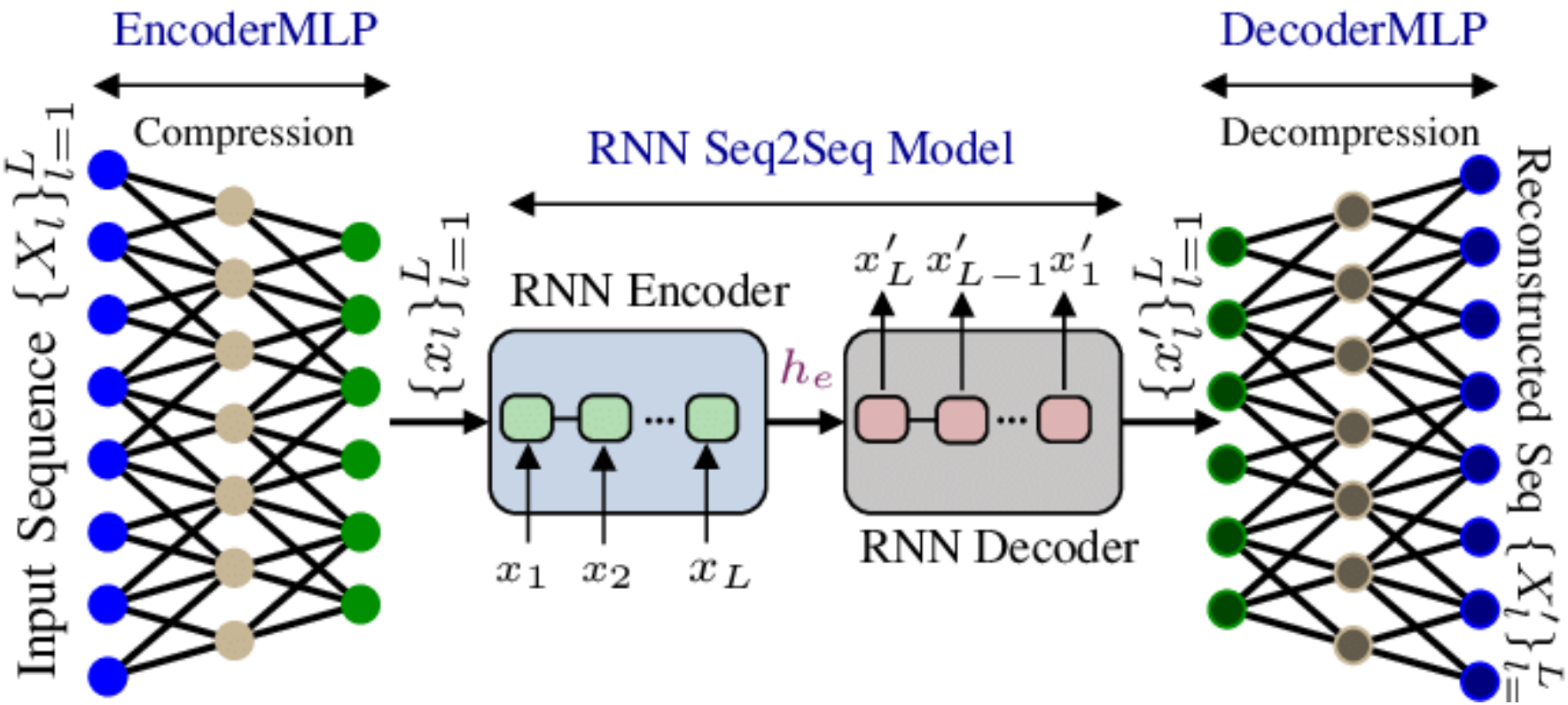


Figure 17-6. Unsupervised pretraining using autoencoders

Convolutional Autoencoders



Recurrent Autoencoders



https://www.researchgate.net/figure/A-depiction-of-a-Recurrent-Autoencoder-RAE-The-input-sequence-X-l-L-l1-is-first_fig4_339737675

Denoising Autoencoders: overcomplete autoencoder

❖ Stacked denoising autoencoders: Overcomplete autoencoder

- The noise can be pure Gaussian noise added to the inputs

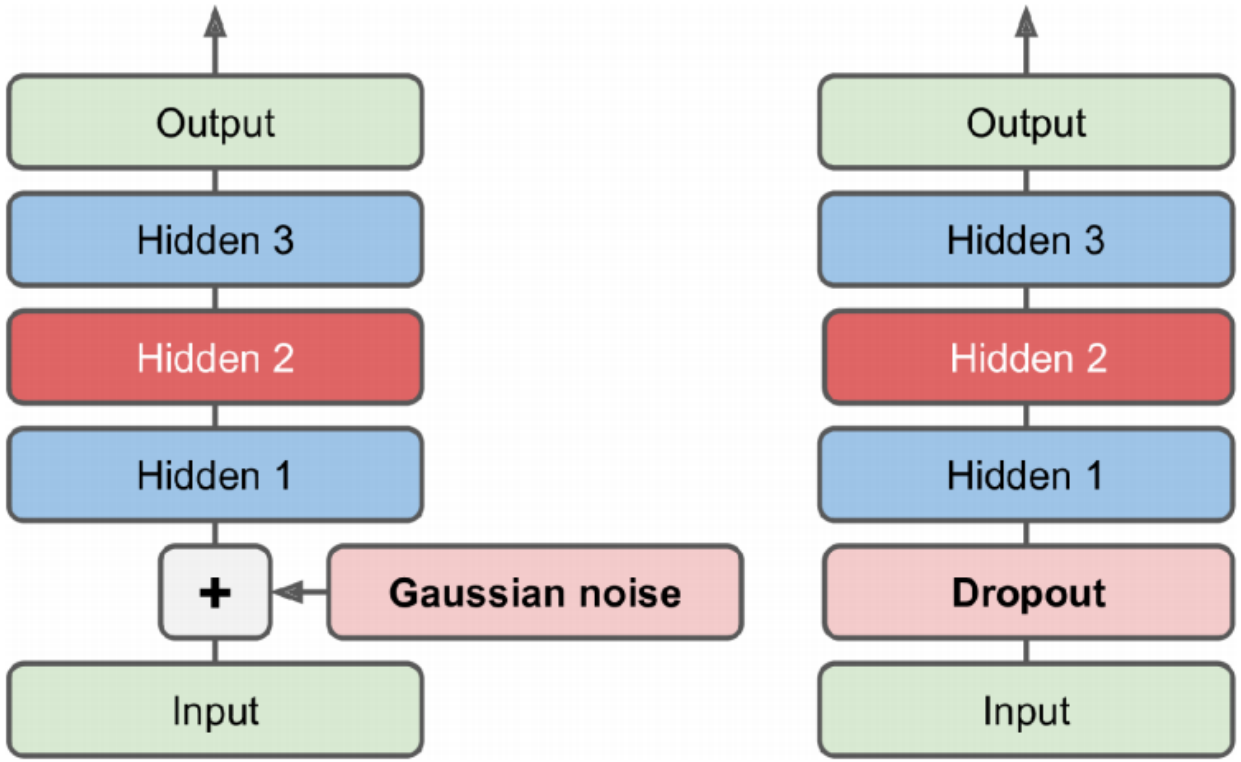
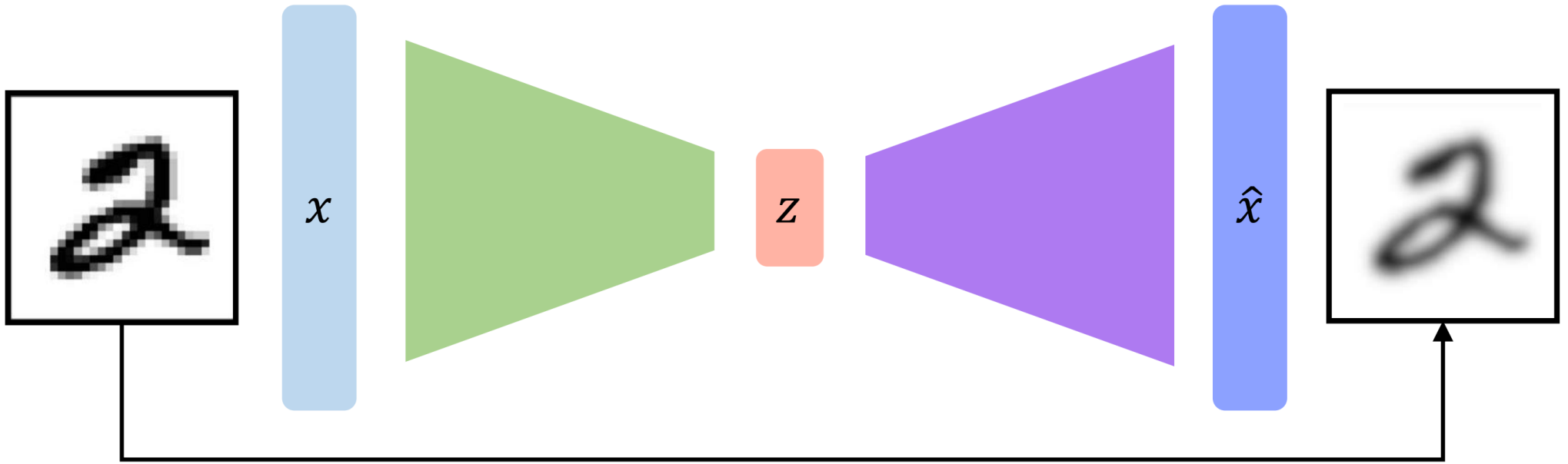


Figure 17-8. Denoising autoencoders, with Gaussian noise (left) or dropout (right)

Autoencoders: background

- How can we learn this latent space?
- Train the model to use these features to reconstruct the original data



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Loss function doesn't use any labels!!

❖ Dimensionality of latent space → reconstruction quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space

5D latent space

Ground Truth

Autoencoders for representation learning

Bottleneck hidden layer forces network to learn a compressed latent representation

Reconstruction loss forces the latent representation to capture (or encode) as much "information" about the data as possible

Autoencoding = **Auto**matically **encoding** data

Variational Autoencoders

❖ Variational autoencoders

- was introduced in 2013 by Diederik Kingma and Max Welling

✓ They are probabilistic autoencoders

- their outputs are partly determined by chance, even after training

✓ Most importantly, they are generative autoencoders,

- they can generate new instances that look like they were sampled from the training set
- Both these properties make them rather similar to RBMs,

✓ VAE perform variational Bayesian inference which is an efficient way to perform approximate Bayesian inference.

Variational Autoencoders

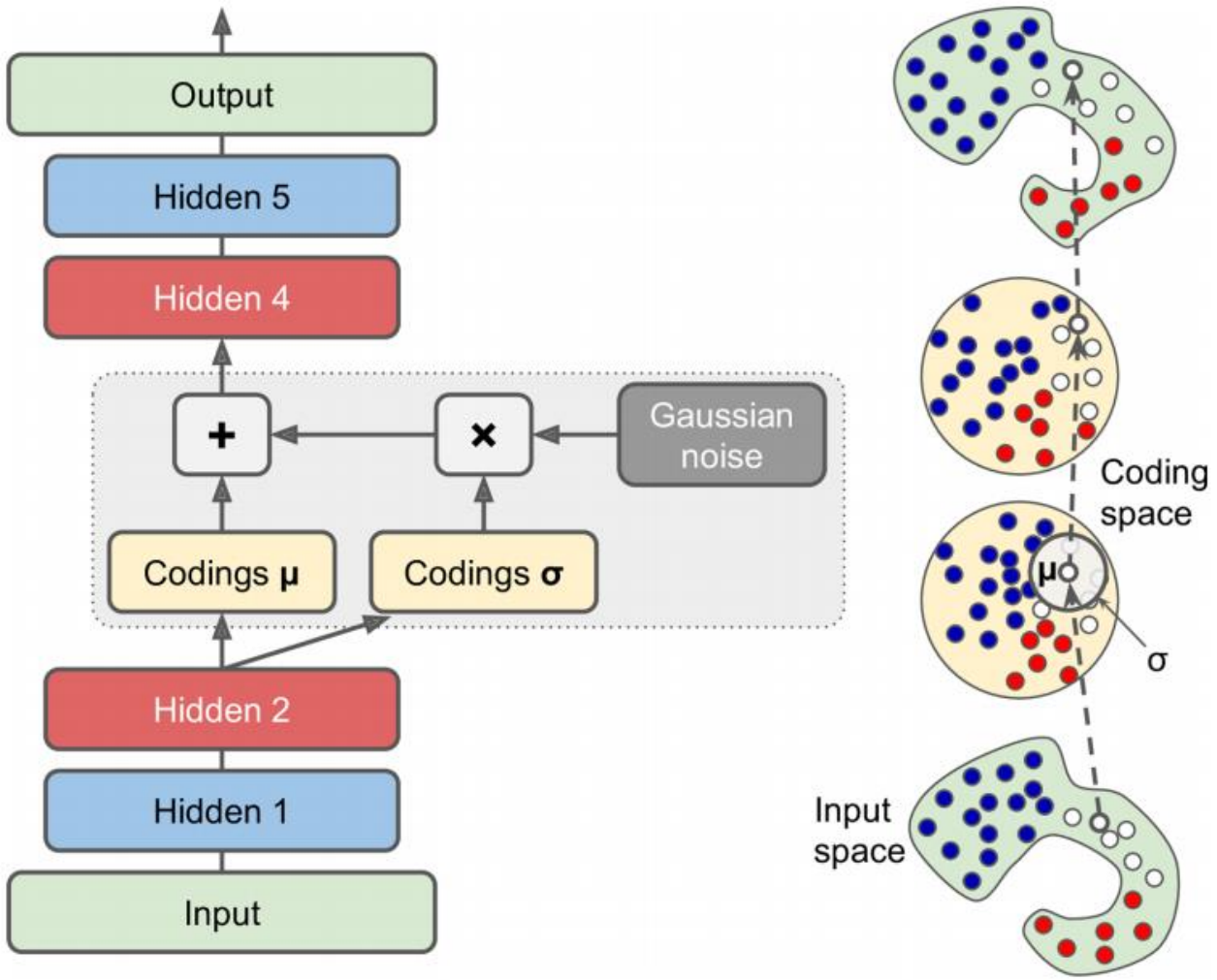
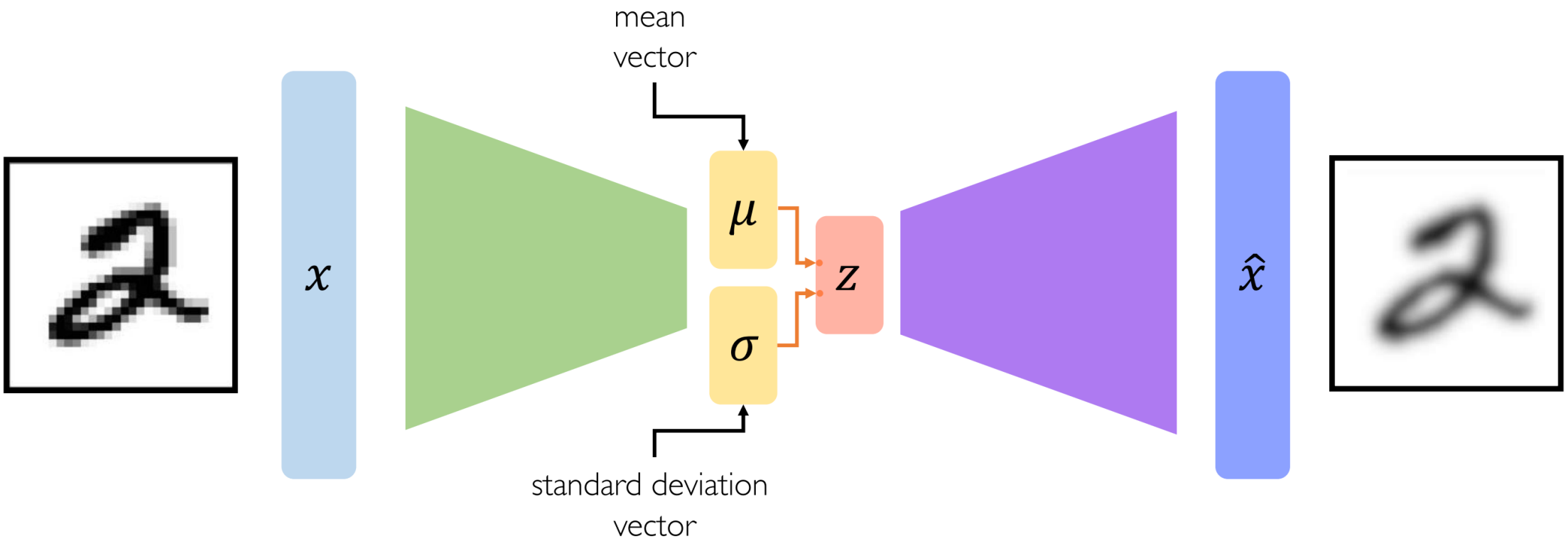


Figure 17-12. Variational autoencoder (left) and an instance going through it (right)

❖ VAEs: key difference with traditional autoencoder

- ✓Variational autoencoders are a probabilistic twist on autoencoders!
- ✓Sample from the mean and standard dev. to compute latent sample

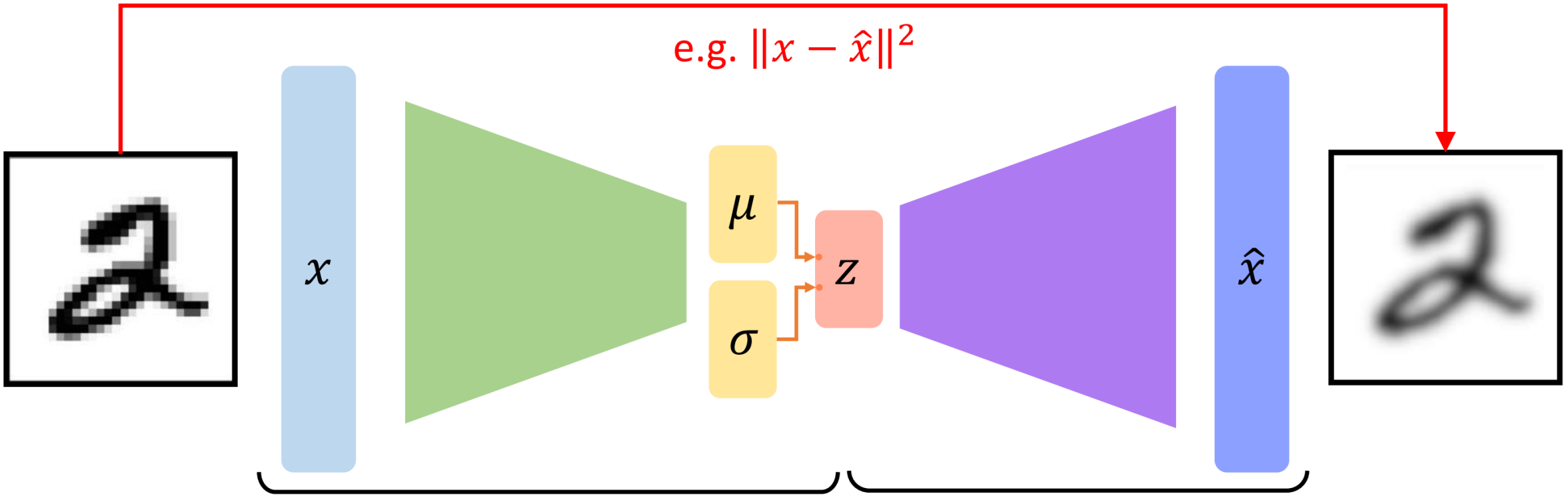


Cost function : Two parts

- The first is the usual reconstruction loss
  - that pushes the autoencoder to reproduce its inputs
  - we can use cross entropy for this
- The second is the latent loss
  - that pushes the autoencoder to have codings that look as though they were sampled from a simple Gaussian distribution: it is the KL divergence between the target distribution and the actual distribution of the codings.

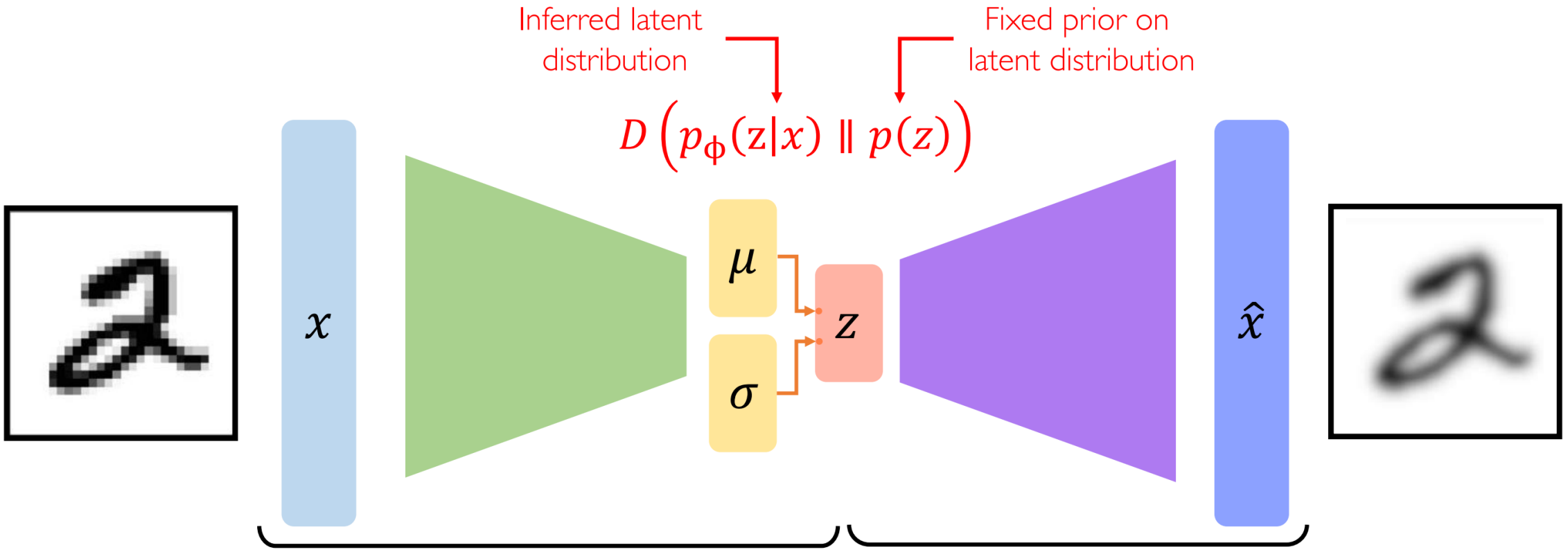
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{regularization term})$$

VAE optimization : Loss



$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

VAE optimization : Loss



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \boxed{\text{(regularization term)}}$$

❖ Variational autoencoder's latent loss

Equation 17-3. Variational autoencoder's latent loss

$$\mathscr{L} = -\frac{1}{2}\sum_{i=1}^{K} 1 + \log\left(\sigma_i^2\right) - \sigma_i^2 - \mu_i^2$$

In this equation, $\mathscr{L}$ is the latent loss, n is the codings' dimensionality, and μ_i and σ_i are the mean and standard deviation of the i^{th} component of the

Equation 17-4. Variational autoencoder's latent loss, rewritten using $\gamma = \log(\sigma^2)$

$$\mathscr{L} = -\frac{1}{2}\sum_{i=1}^{K} 1 + \gamma_i - \exp\left(\gamma_i\right) - \mu_i^2$$

Variational Autoencoders: Generating Data!

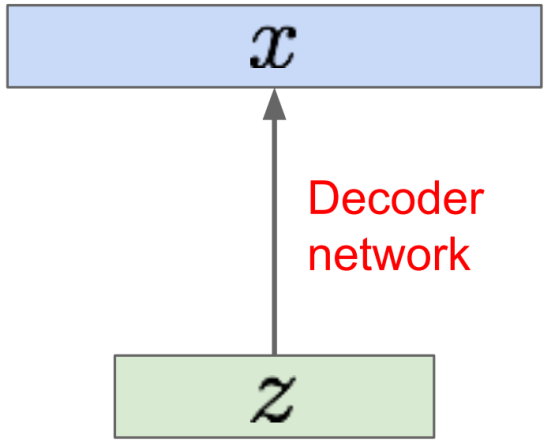
Our assumption about data generation process

Sample from true conditional

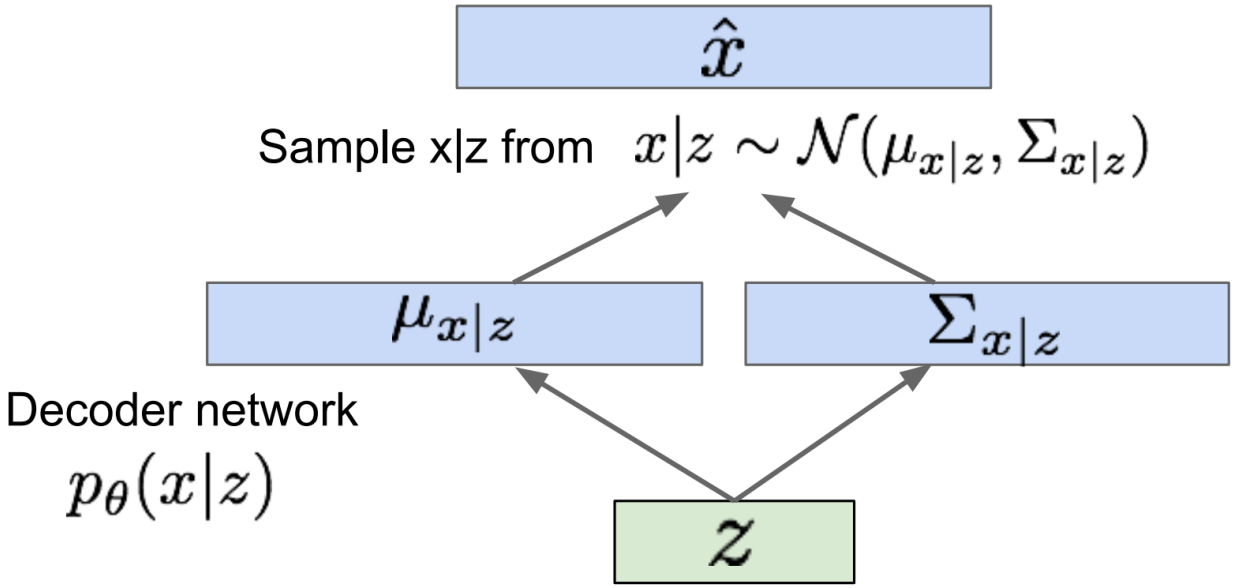
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$



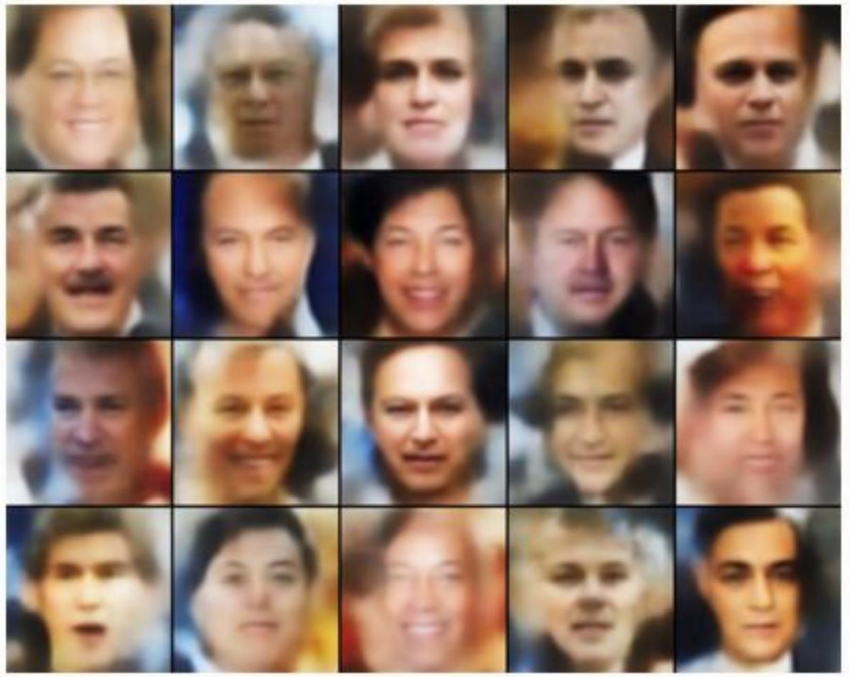
Now given a trained VAE:
use decoder network & sample z from prior!



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

32x32 CIFAR-10



Labeled Faces in the Wild

Figures copyright (L) Dirk Kingma et al. 2016; (R) Anders Larsen et al. 2017. Reproduced with permission.

Variational Autoencoders

- **Probabilistic spin to traditional autoencoders**
  - ✓ => allows generating data
- **Defines an intractable density**
  - ✓ => derive and optimize a (variational) lower bound
- **Pros:**
  - ✓ Principled approach to generative models
  - ✓ Interpretable latent space.
  - ✓ Allows inference of q(z|x), can be useful feature representation for other tasks
- **Cons:**
  - ✓ Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
  - ✓ Samples blurrier and lower quality compared to state-of-the-art (GANs)

Generative Adversarial Networks (GANs)

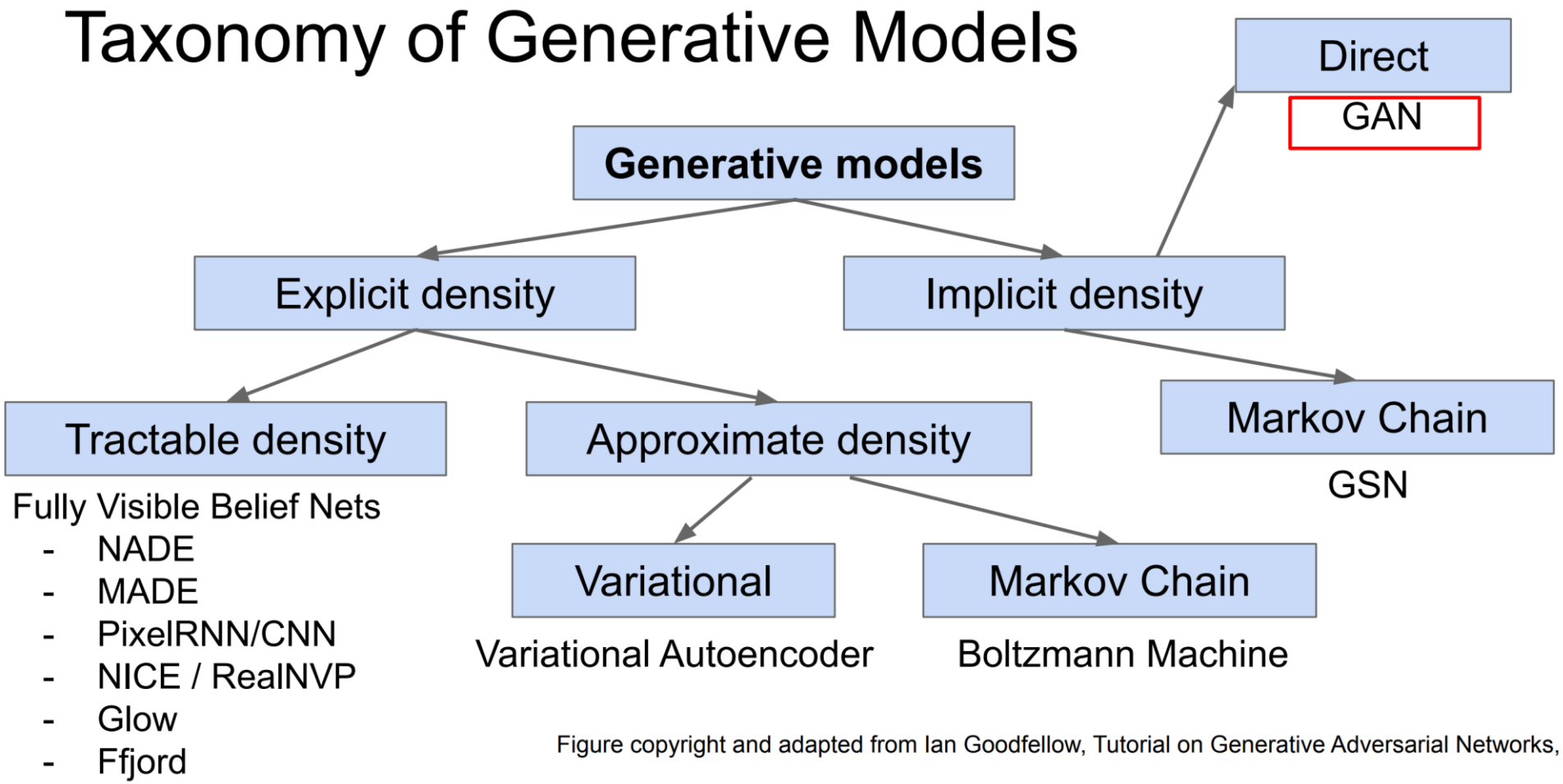
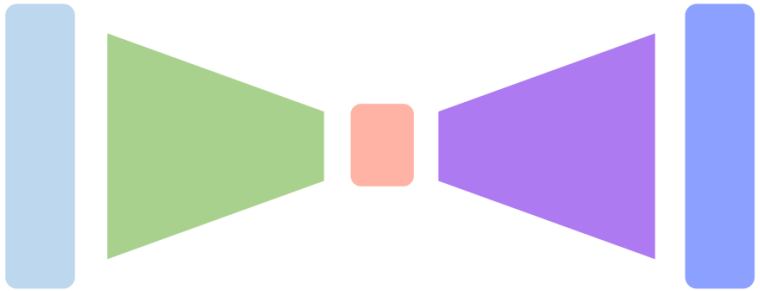


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Deep Generative Models

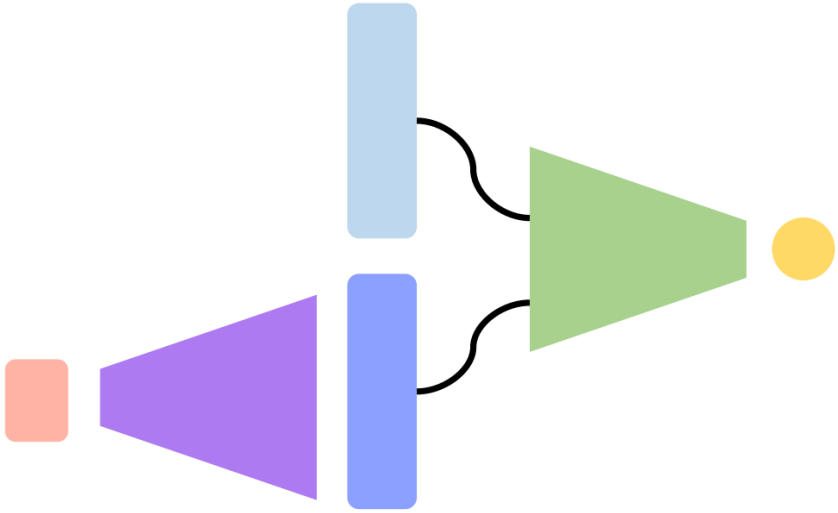
Autoencoders and Variational Autoencoders (VAEs)

Learn **lower-dimensional** latent space and **sample** to generate input reconstructions



Generative Adversarial Networks (GANs)

Competing **generator** and **discriminator** networks

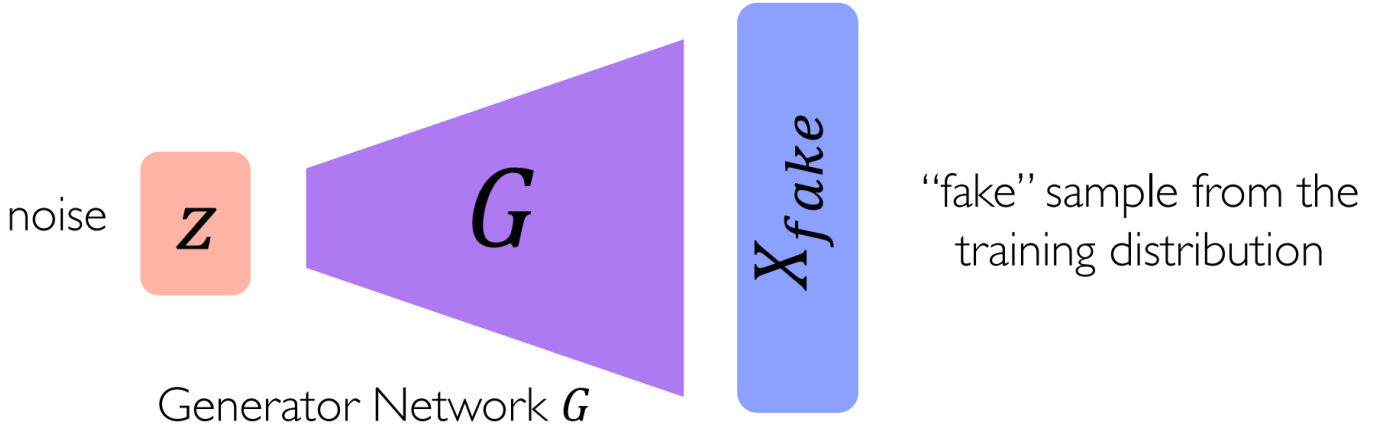


What if we just want to sample?

Idea: don't explicitly model density, and instead just sample to generate new instances.

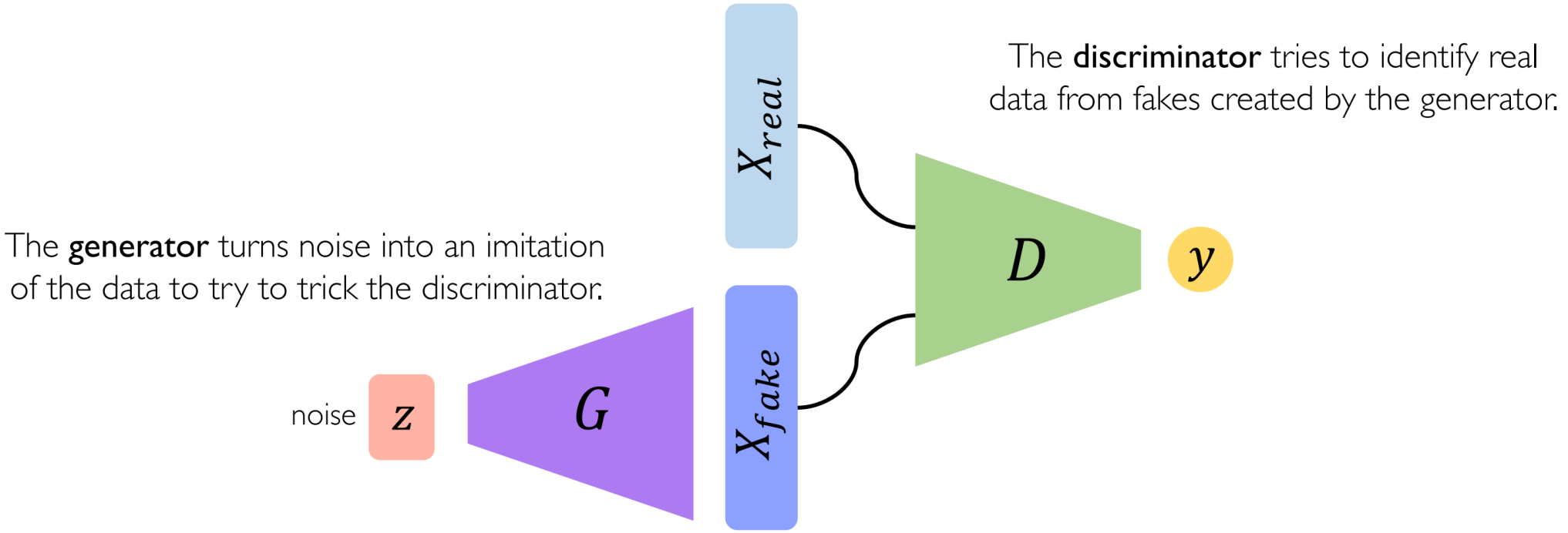
Problem: want to sample from complex distribution – can't do this directly!

Solution: sample from something simple (noise), learn a transformation to the training distribution.



Generative Adversarial Networks (GANs)

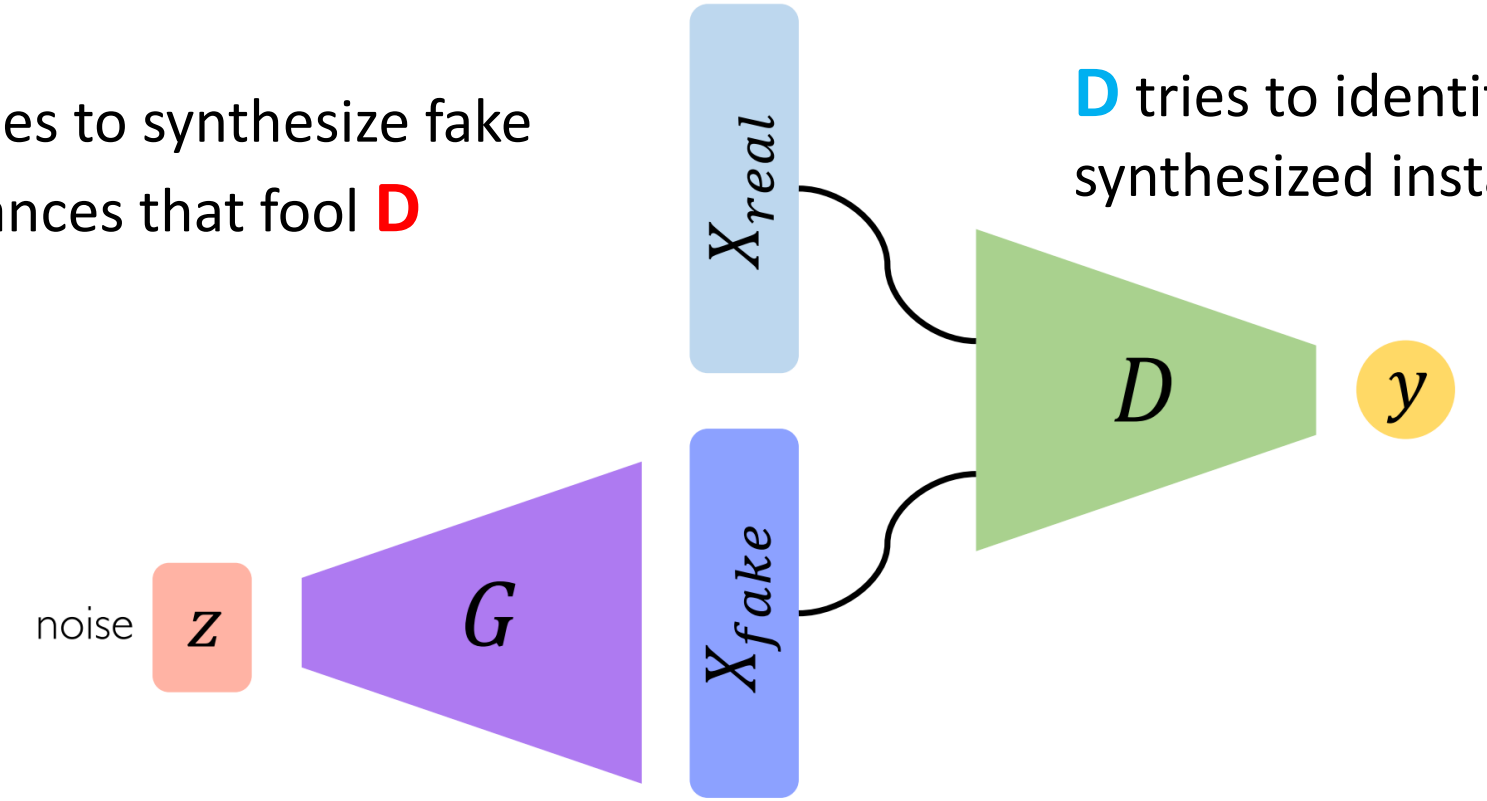
- GANs are a way to make a generative model by having two neural networks compete with each other.



Training GANs

G tries to synthesize fake Instances that fool **D**

D tries to identify the synthesized instances



Training: adversarial objectives for **D** and **G**
Global optimum: G reproduces the true data distribution

Training GANs

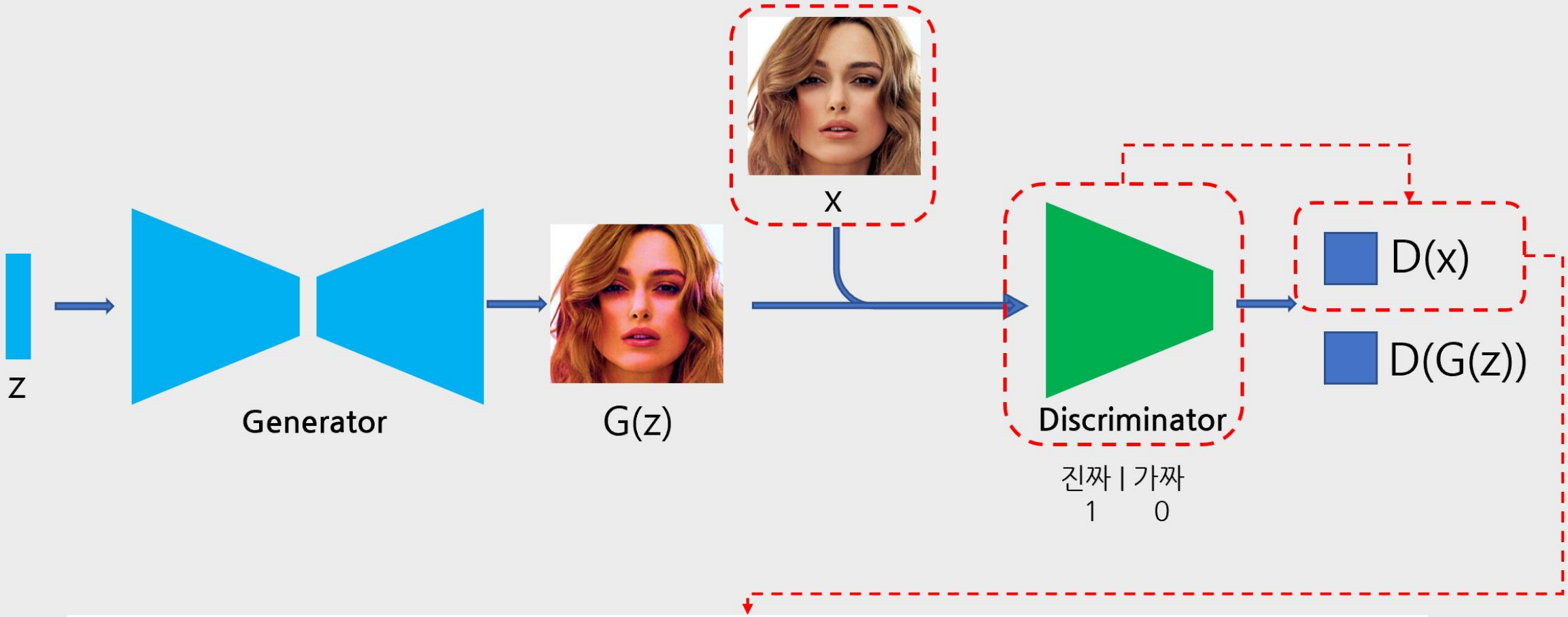
- Discriminator tries to identify real data from fakes created by the generator.
- Generator tries to create imitations of data to trick the discriminator.

Train GAN jointly via **minimax** game:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d} \left(G_{\theta_g}(z) \right) \right) \right]$$

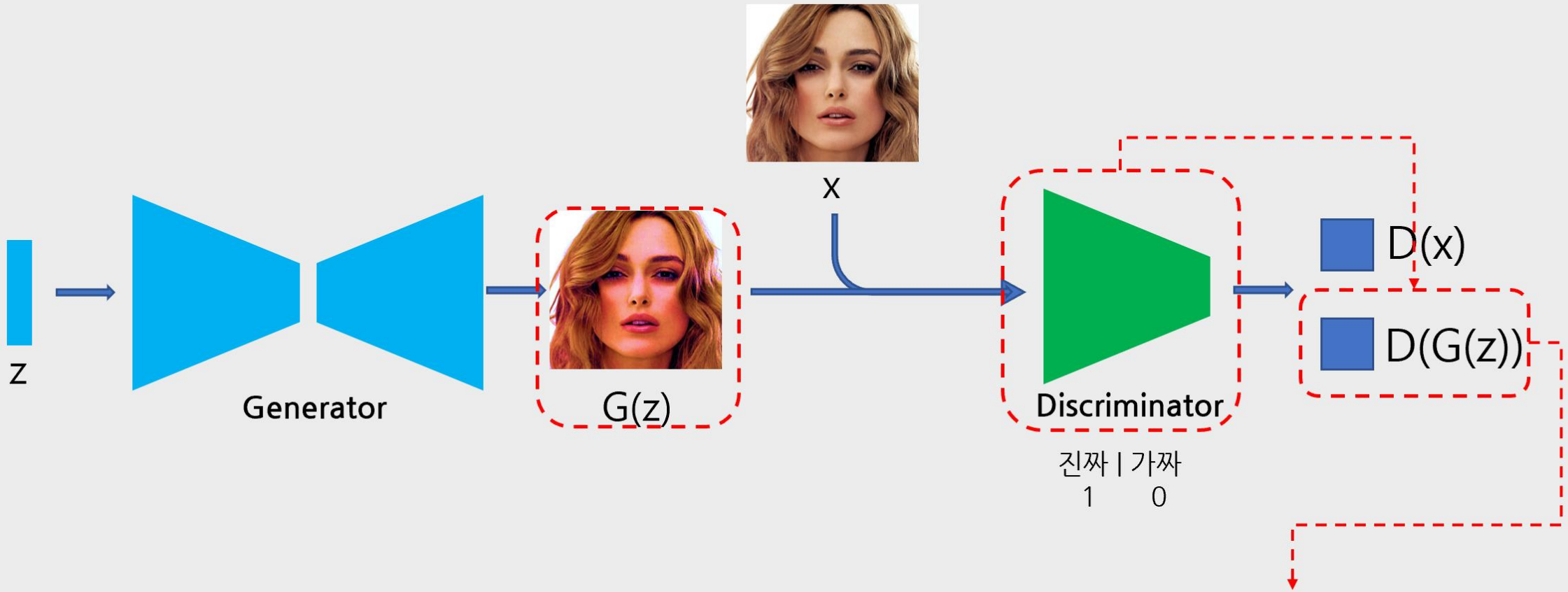
Discriminator wants to maximize objective s.t. $D(x)$ close to 1, $D(G(z))$ close to 0.
Generator wants to minimize objective s.t. $D(G(z))$ close to 1.

Loss function : Discriminator (1/2)



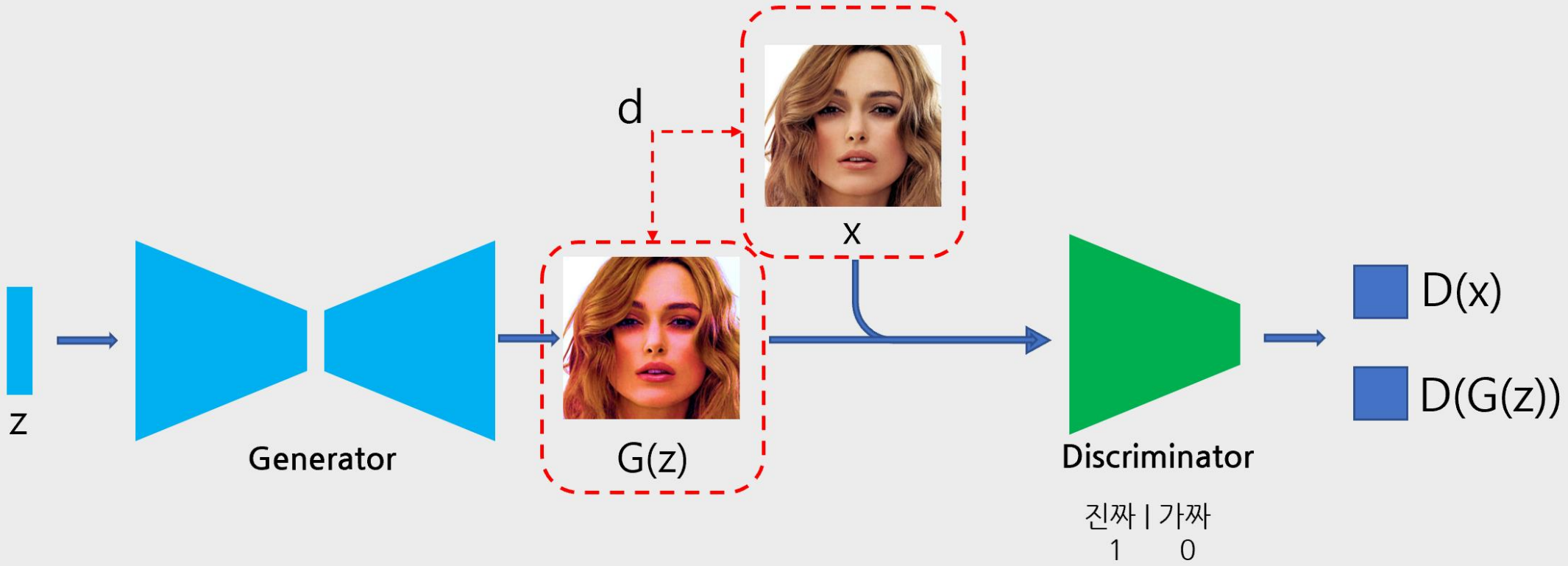
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_x(z)}[log(1 - D(G(z)))]$$

Loss function : Discriminator (2/2)



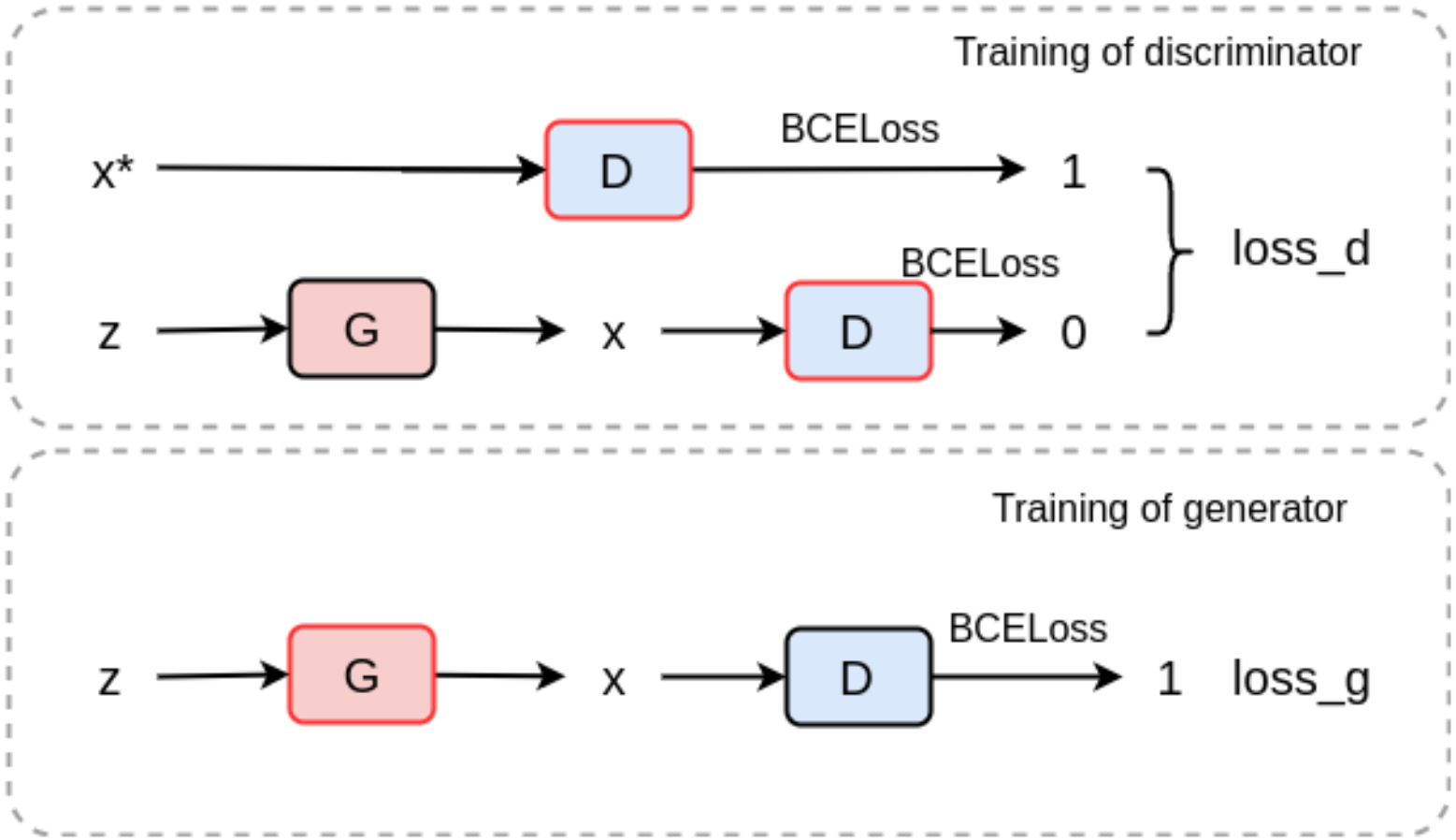
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_x(z)}[log(1-D(G(z)))]$$

Loss function : Generator



$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \arg\max_G log(D(G(z)))$$

Training process for GAN



Here,
x* denotes real data,
x denotes fake data,
and z denotes the latent vector

KISTI SINCE 1962

2022 TRUST KISTI

Korea Institute of Science and Technology Information