Autoencoder

Notations

$\mathbf{x}$	One data sample from the dataset, $\mathbf{x} \in \mathcal{D}$.
$\mathbf{x}'$	The reconstructed version of $\mathbf{x}$.
$\tilde{\mathbf{x}}$	The corrupted version of $\mathbf{x}$.
$\mathbf{z}$	The compressed code learned in the bottleneck layer.
$a_j^{(l)}$	The activation function for the j-th neuron in the l-th hidden layer.
$g_\phi(.)$	The **encoding** function parameterized by ϕ.
$f_\theta(.)$	The **decoding** function parameterized by θ.
$q_\phi(\mathbf{z}\|\mathbf{x})$	Estimated posterior probability function, also known as **probabilistic encoder**.
$p_\theta(\mathbf{x}\|\mathbf{z})$	Likelihood of generating true data sample given the latent code, also known as **probabilistic decoder**.

Autoencoder

Large files

- When we have large files, we often **compress** them to save space. Later, we **decompress** them when we need the original contents.

Step	File System Analogy
Compress	ZIP the file
Store zipped	Keep only .zip file
Decompress	Unzip back to original

High-dimensional data

- Many datasets (e.g., images, audio, text) are:
  - **High-dimensional**: e.g., a 128×128 RGB image = 49,152 features
  - **Redundant**: nearby pixels or time steps often carry similar information
  - **Noisy**: contain uninformative or irrelevant variations (e.g., lighting, background)

- Compressing helps focus on **what really matters**.

Representation Learning

- In autoencoders:
  - The **encoder** transforms input x into a **lower-dimensional latent code** z
  - The goal is to **preserve important information**, not just reduce size

- This process is not arbitrary:
  - The encoder **learns to discard irrelevant variations**
  - It extracts the **most informative and useful features** to reconstruct x

What Happens During Compression (Encoder)

- We learn a function g_ϕ which maps

$$x \in \mathbb{R}^D \quad \longrightarrow \quad z \in \mathbb{R}^d \quad \text{with } d \ll D$$

- This transformation:
  - Reduces dimension
  - Filters out noise
  - Forces the model to find **structure** in the data

Why it this useful?

- **Dimensionality Reduction**
  - Visualize data (e.g., 2D t-SNE)
  - Make downstream models faster and simpler
- **Feature Extraction**
  - Latent code z can be reused for classification, clustering, etc.
- **Robustness**
  - Small, informative codes resist overfitting
  - Encourages generalization

Latent Code Alone Is Not Enough

- After the encoder compresses the input x into a latent representation z, we want to:
  - **Check whether this code really captures the important parts** of the original input
  - **Reconstruct** the input data to verify information retention
- A decoder gives the latent code a **“reality check”** — can we get back what we started with?

What Does the Decoder Actually Do?

- The decoder is a **learned function**:

$$f_\theta(z) \rightarrow x'$$

- It tries to **map the compact latent vector** z back to the **original data space**, reconstructing as accurately as possible:

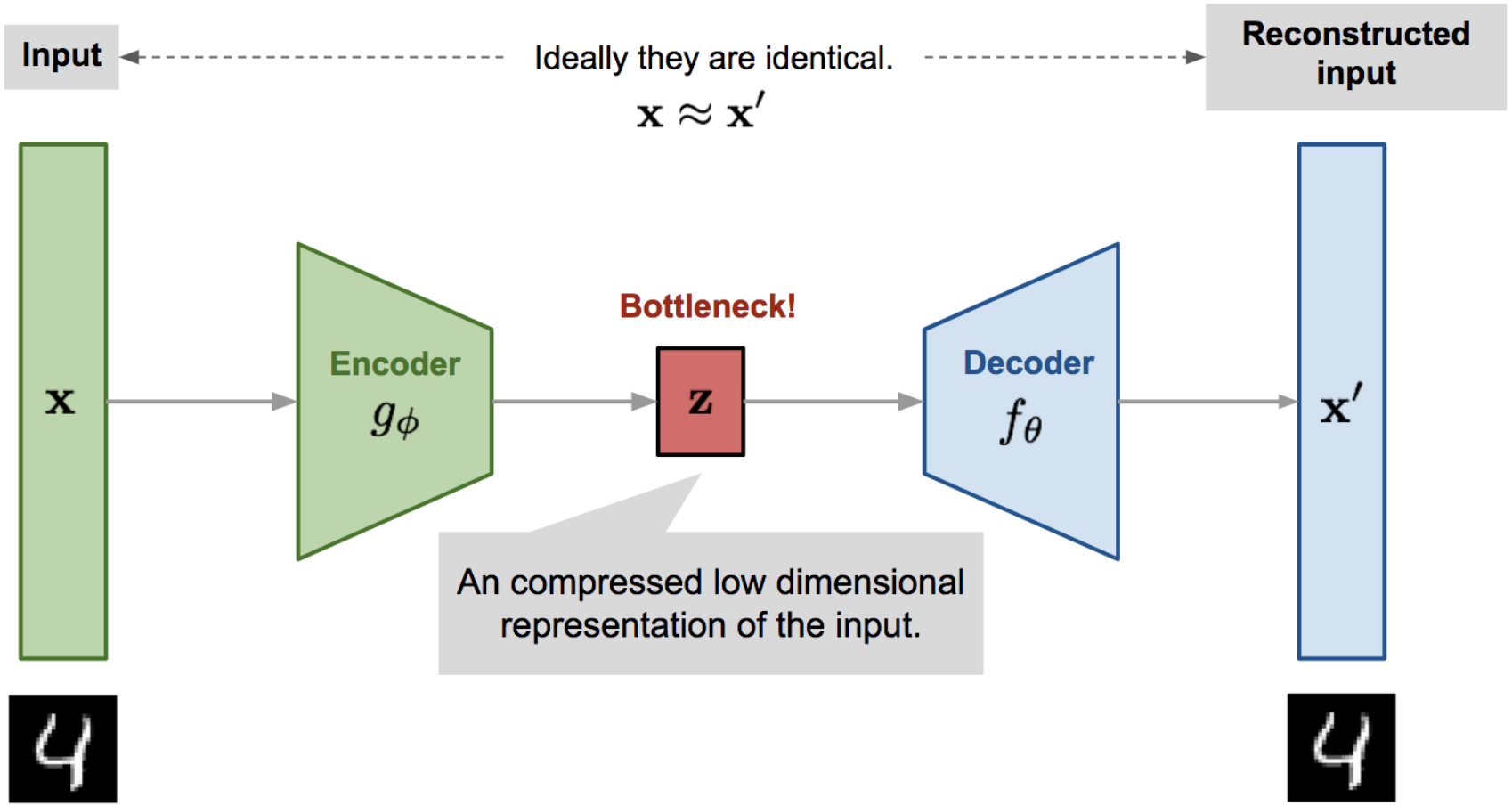
$$x' \approx x$$

- This trains the encoder to:
  - Learn **compressions that are reversible**
  - Avoid trivial or lossy encodings

Analogy

Step	File System Analogy	Autoencoder Analogy
Compress	ZIP the file	**Encoder** maps input x → latent z
Store zipped	Keep only .zip file	Store the compact latent features
Decompress	Unzip back to original	**Decoder** reconstructs \hat{x} from z

Architecture



Objective function

- We want to recover x’ as much as possible, that is minimize the difference between x’ and x

$$\|x - x'\|$$

- There are various metrics to quantify the difference between two vectors, such as cross entropy when the activation function is sigmoid, or as simple as MSE loss:

$$L_{\mathrm{AE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2$$

Denoising Autoencoder (DAE)

Limitations of AE

- **Trivial Identity Mapping**
  - If the **encoder and decoder have too much capacity**, the network might simply learn to **copy** the input to the output without extracting useful structure.
  - Especially true when the latent dimension is large or the model is overparameterized.

- **Sensitive to Noise**
  - A small perturbation in the input can lead to a large difference in the reconstruction.
  - The model doesn’t learn robustness by default.

Core idea

- If the model learns to recover the **original input** from a **corrupted version**, it must have captured **deeper structure** in the data

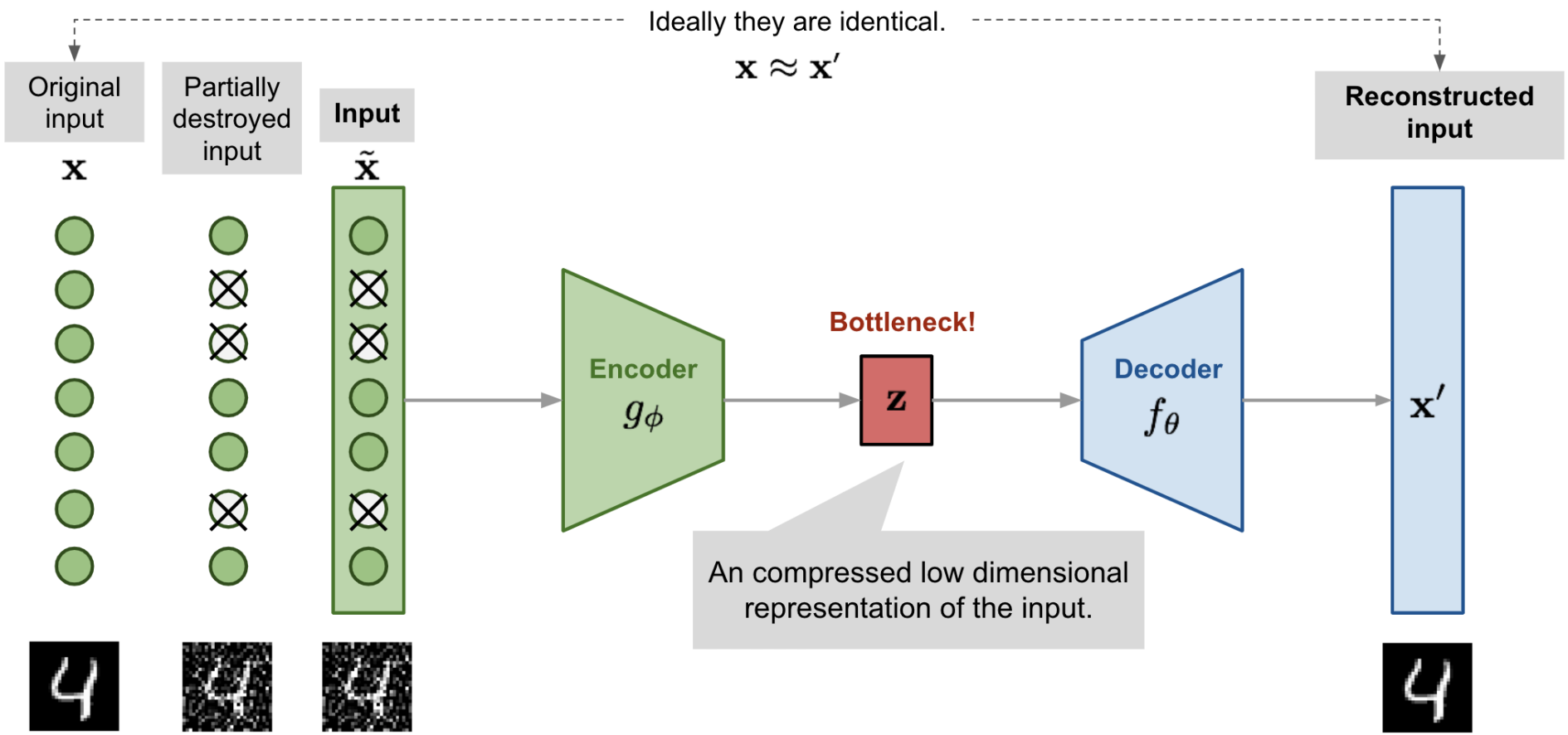
- We will corrupt the input (add noise)

$$x \rightarrow \tilde{\tilde{x}}$$

Types of Noise in DAE

- Gaussian noise: $\tilde{x} = x + \varepsilon$
- Masking noise: randomly zero out features
- Salt-and-pepper noise: random 0/1 flips

Architecture



Objective function

- Similar to AE, we can use MSE as the loss function

$$L_{\mathrm{DAE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\tilde{\mathbf{x}}^{(i)})))^2$$

Variational Autoencoder

Core idea

- **Latent space becomes probabilistic**
  - Instead of mapping input x to a point z, we map to a **distribution**

$$q_\phi(\mathbf{z} \mid \mathbf{x})$$

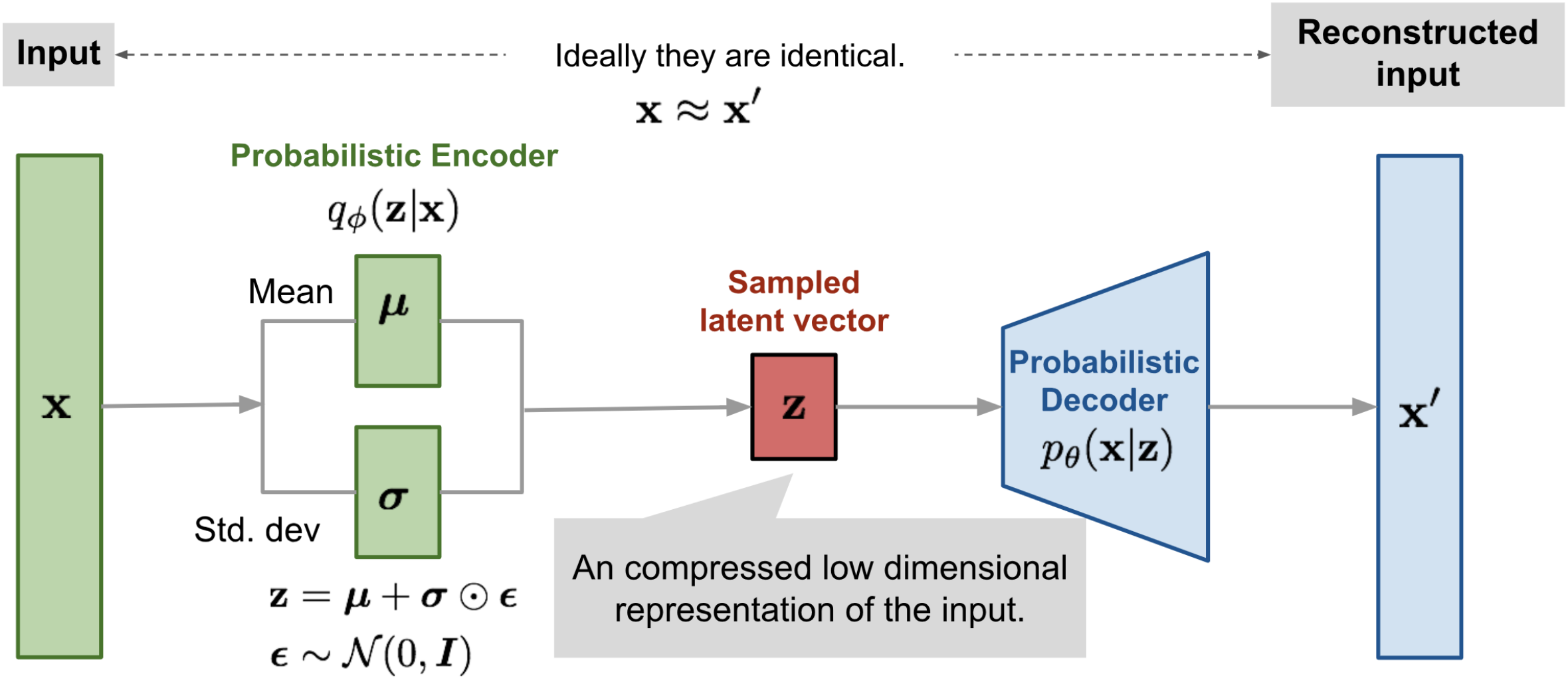
typically a Gaussian.

- **Add a prior over latent space**
  - Encourage all latent codes to stay close to a prior

$$p(\mathbf{z}) = \mathcal{N}(0, I)$$

  - so we can **sample** from it at test time.

Architecture



Comparison: AE vs. DAE vs. VAE

- AE: deterministic, no noise, not generative
- DAE: robust to input noise, still deterministic
- VAE: probabilistic, structured latent space, generative

GAN vs. VAE

GAN: Adversarial training

VAE: maximize variational lower bound