Autoencoders and Latent Representations

Unsupervised learning of compressed data representations

Autoencoder Architecture

Input
Original Data

→

Encoder
Maps to latent space

→

Latent Space
Low-dimensional representation

→

Decoder
Reconstructs input

→

Output
Reconstructed Data

⚙️ Bottleneck forces learning of essential features

📚 Historical Context

1980s: Hinton and Rumelhart pioneered dimensionality reduction using backpropagation

2006: Hinton's Deep Belief Networks marked the dawn of the deep learning era

2013: Kingma and Welling proposed the Variational Autoencoder (VAE)

Present: Evolved into a core technology for generative models, anomaly detection, and representation learning

🔬 Structural Details

Encoder: $x \rightarrow z = f_{enc}(x; \theta_{enc})$

- Compresses input dimension n to latent dimension d ($d \ll n$)
- Composed of multiple neural network layers (e.g., FC layers, CNNs)

Decoder: z → x̂ = f_dec(z; θ_dec)
• Reconstructs original input from latent representation z
• Can be symmetric or independent from encoder architecture

Loss Function: $L = ||x - \hat{x}||^2$ (reconstruction error)

🧮 Vector Operation Example

Simple Example: 784-dimensional MNIST image → 2-dimensional latent space

```
Input vector: x ∈ ℝ^784 (28×28 image flattened)
e.g., x = [0.1, 0.0, 0.8, ..., 0.3]

Encoder operations:
h₁ = ReLU(W₁ x + b₁ ) # W₁ ∈ ℝ^(128×784)
h₂ = ReLU(W₂ h₁ + b₂ ) # W₂ ∈ ℝ^(64×128)
z = W₃ h₂ + b₃ # W₃ ∈ ℝ^(2×64), z ∈ ℝ^2
→ z = [1.2, -0.5] # 2D latent vector

Decoder operations:
h₃ = ReLU(W₄ z + b₄ ) # W₄ ∈ ℝ^(64×2)
h₄ = ReLU(W₅h₃ + b₅) # W₅ ∈ ℝ^(128×64)
x̂= σ(W₆h₄ + b₆) # W₆ ∈ ℝ^(784×128)
→ x̂= [0.09, 0.02, 0.79, ..., 0.31]

Loss: MSE = (1/784) Σ(xᵢ - x̂ᵢ)² = 0.003
```

🎲 Variational Autoencoders (VAE)

Learn probabilistic latent distributions for generation and sampling
• Models latent space as probability distribution in the form $z \sim N(\mu, \sigma^2)$
• Learns regularized latent space with additional KL divergence

Dimensionality Reduction

Denoising

Generation