# Variational Autoencoders on MNIST

## ELBO Derivation, KL Annealing, and CVAE Extension

Alexis Le Trung, Khatir Youyou, Tidjani Adam Kandine,
Yahya Ahachim, Aniss Outaleb, Quentin Wurtlin

Generative AI and Diffusion Models

January 18, 2026

# Project Objectives

1. **Derive the ELBO** from first principles with closed-form KL for Gaussians
2. **Implement a Convolutional VAE** on MNIST with 2D latent space
3. **Track reconstruction vs KL** separately, implement KL annealing
4. **Visualize**: latent space, latent traversals, interpolations
5. **Extend to CVAE** for controlled digit generation

# ELBO Derivation

Starting from intractable marginal likelihood, we derived:

## Evidence Lower Bound

$$\log p(x) \geq \mathcal{L} = \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction (BCE)}} - \underbrace{\mathrm{KL}(q_\phi(z|x) \| p(z))}_{\text{Regularization}}$$

**Closed-form KL** for Gaussian encoder $q = \mathcal{N}(\mu, \sigma^2)$ and prior $p = \mathcal{N}(0, I)$:

$$\mathrm{KL}(q\|p) = \frac{1}{2} \sum_{j=1}^{d} \left(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1\right)$$

**Gradients:** $\frac{\partial \mathrm{KL}}{\partial \mu_j} = \mu_j, \quad \frac{\partial \mathrm{KL}}{\partial \sigma_j} = \sigma_j - \frac{1}{\sigma_j}$

# Architecture & Training Setup

**Convolutional VAE:**

- Encoder: Conv layers $\rightarrow$ 2D latent $(\mu, \log \sigma^2)$
- Decoder: Linear $\rightarrow$ ConvTranspose layers
- Reparameterization: $z = \mu + \sigma \odot \epsilon$

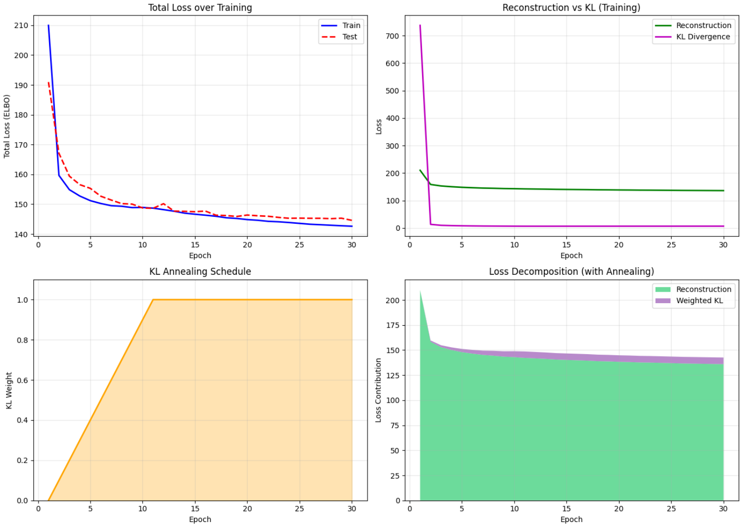
**KL Annealing:**

$$\beta(t) = \min\left(1, \frac{t}{10}\right)$$

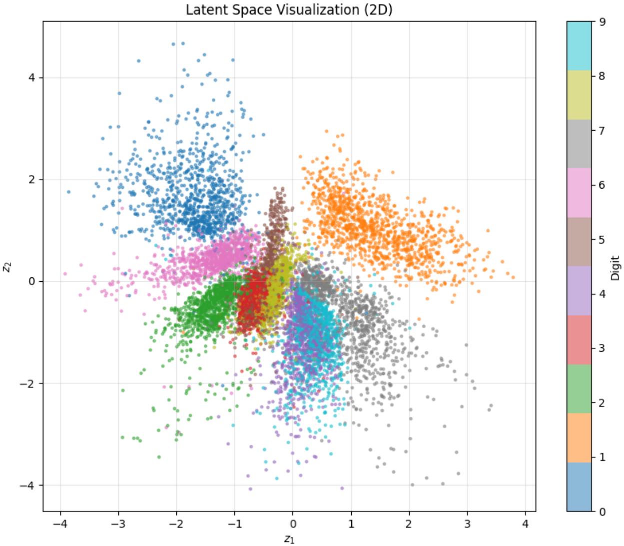
Prevents posterior collapse

| **Parameter** | **Value** |
|---|---|
| Latent dim | 2 |
| Batch size | 128 |
| Learning rate | $10^{-3}$ |
| Optimizer | Adam |
| Epochs | 30 |
| KL warmup | 10 epochs |

# VAE Training Curves



# Latent Space Visualization

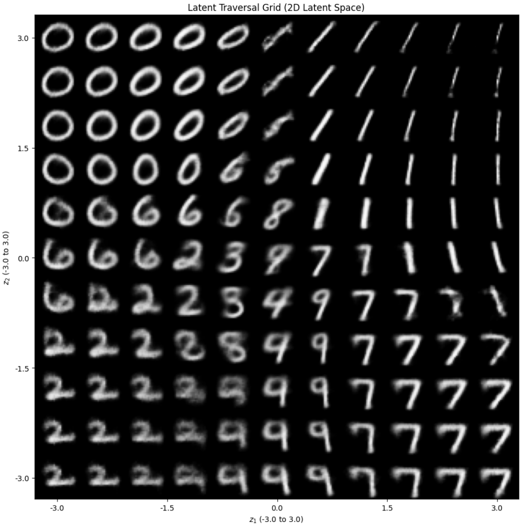



**Observations:**

- Clear clustering by digit class
- Similar digits nearby (4/9, 3/8, 1/7)
- Smooth, continuous manifold
- Matches $\mathcal{N}(0, I)$ prior

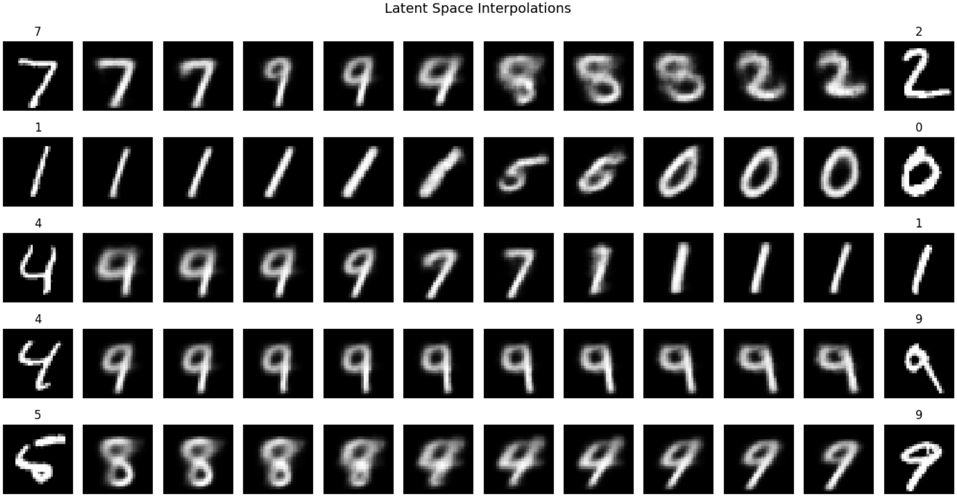
**Key insight:**
Organization emerges from reconstruction alone — no labels used!

# Latent Traversal Grid




**Traversing $z_1$ vs $z_2$:** $z_1$ (horizontal) controls slant/rotation; $z_2$ (vertical) controls thickness/scale.

# Latent Interpolations




Linear interpolation between encoded digit pairs. Smooth transitions validate the continuous, well-structured latent space.

# CVAE Extension: Conditional Generation

**Modification:** Condition encoder and decoder on class label $c$

$$\mathcal{L}_{\mathsf{CVAE}} = \mathbb{E}_{q_\phi(z|x,c)}[\log p_\theta(x|z,c)] - \mathrm{KL}(q_\phi(z|x,c) \| p(z))$$

**Implementation:**

- One-hot encode label $\rightarrow$ broadcast to $10 \times 28 \times 28$
- Concatenate with image as extra channels
- Class-independent prior: $p(z) = \mathcal{N}(0, I)$

**Result:** Latent space encodes *style*, label provides *class identity*

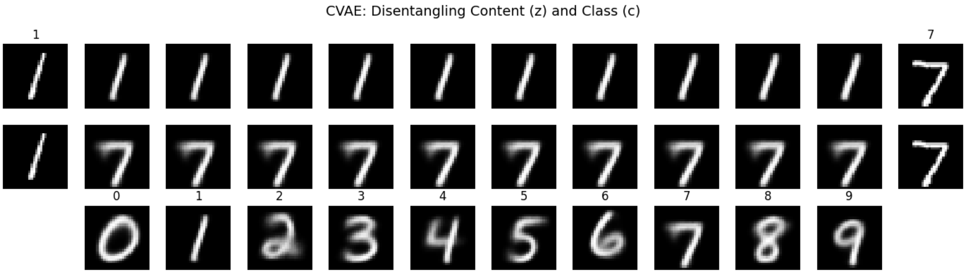
# VAE vs CVAE: Quantitative Comparison

| Model | Recon. Loss | KL | Total Loss |
|---|---|---|---|
| VAE | 138.00 | 6.59 | 144.59 |
| CVAE | 124.36 | 4.80 | 129.16 |
| **Improvement** | **9.9%** | **27.2%** | **10.7%** |

**Why CVAE performs better:**

- Decoder doesn't need to infer class from $z \Rightarrow$ simpler task
- Latent space focuses purely on style variations
- More compact representation (lower KL)

# CVAE: Disentanglement Demonstration




**Row 1-2:** Same $z$ interpolation with different class labels (1 vs 7). **Row 3:** Fixed $z$, varying class 0-9 $\Rightarrow$ style transfer across all digits.

# Key Findings

1. **ELBO provides principled training objective**
   - Reconstruction + regularization trade-off
   - Closed-form KL enables efficient optimization
2. **KL annealing is essential**
   - Without it: posterior collapse, KL $\to 0$
   - 10-epoch warmup gave balanced training
3. **VAE learns meaningful structure unsupervised**
   - Class clustering, smooth interpolations
4. **CVAE achieves explicit disentanglement**
   - 9.9% better reconstruction, controlled generation

# Thank You!

Questions?

Code available in accompanying Jupyter notebook