# Variational Autoencoders: Theory, Implementation, and Conditional Extensions on MNIST

Alexis Le Trung, Khatir Youyou, Tidjani Adam Kandine,
Yahya Ahachim, Aniss Outaleb, Quentin Wurtlin
*Course: Generative AI and Diffusion Models*

January 18, 2026

**Abstract**

This report presents a comprehensive study of Variational Autoencoders (VAEs), encompassing both theoretical foundations and practical implementation. We derive the Evidence Lower Bound (ELBO) from first principles, implement a convolutional VAE architecture for the MNIST dataset, and investigate training dynamics through KL divergence annealing. We extend the standard VAE to a Conditional VAE (CVAE) that enables controlled generation by conditioning on class labels. Our experiments demonstrate that CVAEs achieve superior reconstruction quality (124.36 vs 138.00) while maintaining lower KL divergence (4.80 vs 6.59), confirming the benefits of incorporating label information. We provide extensive visualizations including latent space embeddings, latent traversals, and interpolations to illustrate the learned representations.

## Contents

# 1 Introduction

Generative modeling represents one of the fundamental challenges in machine learning: learning to model the underlying probability distribution of data such that we can generate new, plausible samples. Among the various approaches to generative modeling, Variational Autoencoders (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014] stand out for their elegant combination of deep learning with principled probabilistic inference.

The core idea behind VAEs is to learn a low-dimensional latent representation of data that captures its essential structure. Unlike traditional autoencoders that learn deterministic mappings, VAEs learn probabilistic encodings, enabling them to generate diverse samples by sampling from the learned latent distribution. This probabilistic formulation is grounded in variational Bayesian inference, providing a theoretically principled framework for balancing reconstruction fidelity against regularization of the latent space.

## 1.1 Objectives

This project aims to:

1. **Derive the ELBO**: Present a complete mathematical derivation of the Evidence Lower Bound from first principles, establishing the theoretical foundation for VAE training.
2. **Implement a Convolutional VAE**: Build and train a convolutional VAE architecture on the MNIST handwritten digit dataset.
3. **Investigate Training Dynamics**: Track reconstruction and KL divergence losses separately, implementing KL annealing to prevent posterior collapse.
4. **Visualize Latent Representations**: Produce latent space visualizations, latent traversals, and interpolations to understand the learned representations.
5. **Extend to Conditional VAE**: Implement a CVAE that conditions generation on class labels, enabling controlled digit generation.

## 1.2 Organization

Section 2 presents the theoretical foundations, including the complete ELBO derivation. Section 3 describes the model architectures for both VAE and CVAE. Section 4 details the training methodology including KL annealing. Section 5 presents experimental results with extensive visualizations. Section 6 discusses findings and limitations. Section 7 concludes with future directions.

# 2 Theoretical Foundations

## 2.1 Problem Setup

Consider a dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$ of i.i.d. samples from an unknown data distribution $p_{\text{data}}(x)$. We assume each observation $x$ is generated by a latent variable $z \in \mathbb{R}^d$ according to the generative process:

$$z \sim p(z) = \mathcal{N}(0, I) \tag{1}$$

$$x \sim p_\theta(x|z) \tag{2}$$

where $p_\theta(x|z)$ is the decoder network parameterized by $\theta$.

Our goal is to maximize the marginal log-likelihood of the data:

$$\log p_\theta(x) = \log \int p_\theta(x|z)p(z)\,dz \tag{3}$$

This integral is intractable for complex decoder networks, motivating the variational approach.

## 2.2 Derivation of the Evidence Lower Bound (ELBO)

**Theorem 1** (Evidence Lower Bound)**.** *For any distribution $q_\phi(z|x)$ over latent variables, the log marginal likelihood satisfies:*

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)\|p(z)) = \mathcal{L}(\theta, \phi; x) \tag{4}$$

*Proof.* We begin by introducing an arbitrary distribution $q_\phi(z|x)$ (the encoder) and applying Jensen's inequality:

**Step 1: Importance sampling formulation**

$$\log p_\theta(x) = \log \int p_\theta(x|z)p(z)\,dz \tag{5}$$

$$= \log \int q_\phi(z|x)\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\,dz \tag{6}$$

$$= \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] \tag{7}$$

**Step 2: Apply Jensen's inequality**

Since log is concave, Jensen's inequality gives:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] \tag{8}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(z)}{q_\phi(z|x)}\right] \tag{9}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)\|p(z)) \tag{10}$$

This completes the derivation of the ELBO. □

## 2.3 Alternative Derivation via KL Decomposition

An illuminating alternative derivation reveals the gap between the ELBO and the true log-likelihood:

$$\log p_\theta(x) = \log p_\theta(x) \int q_\phi(z|x)\,dz \tag{11}$$

$$= \int q_\phi(z|x) \log p_\theta(x)\,dz \tag{12}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] \tag{13}$$

Using Bayes' rule $p_\theta(z|x) = p_\theta(x|z)p(z)/p_\theta(x)$:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)}\right] \tag{14}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \tag{15}$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)\|p(z))}_{\text{ELBO}} + \underbrace{\mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))}_{\geq 0} \tag{16}$$

This shows that the gap equals $\mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))$, measuring how well the encoder approximates the true posterior.

### 2.4 Closed-Form KL Divergence for Gaussians

We parameterize the encoder as $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x)))$. With standard Gaussian prior $p(z) = \mathcal{N}(0, I)$, the KL divergence has a closed form:

**Theorem 2** (Gaussian KL Divergence)**.** *For $q = \mathcal{N}(\mu, diag(\sigma^2))$ and $p = \mathcal{N}(0, I)$ in $d$ dimensions:*

$$\mathrm{KL}(q\|p) = \frac{1}{2}\sum_{j=1}^{d}\left(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1\right) \tag{17}$$

*Proof.* By definition:

$$\begin{aligned}
\mathrm{KL}(q\|p) &= \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(z)] && (18)\\
&= -\frac{1}{2}\sum_j\left(1 + \log\sigma_j^2\right) + \frac{1}{2}\mathbb{E}_q\left[\sum_j z_j^2\right] && (19)\\
&= -\frac{1}{2}\sum_j\left(1 + \log\sigma_j^2\right) + \frac{1}{2}\sum_j(\mu_j^2 + \sigma_j^2) && (20)\\
&= \frac{1}{2}\sum_j\left(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1\right) && (21)
\end{aligned}$$

□

### 2.5 Gradients for Optimization

For gradient-based optimization, we need derivatives of the KL divergence:

$$\frac{\partial D_{KL}}{\partial \mu_j} = \mu_j \tag{22}$$

$$\frac{\partial D_{KL}}{\partial \sigma_j} = \sigma_j - \frac{1}{\sigma_j} = \frac{\sigma_j^2 - 1}{\sigma_j} \tag{23}$$

These gradients reveal important dynamics: the KL term penalizes large mean values (pulling $\mu \to 0$) and deviations from unit variance (pulling $\sigma \to 1$).

### 2.6 The Reparameterization Trick

A key innovation of VAEs is the reparameterization trick [Kingma and Welling, 2014], which enables backpropagation through stochastic sampling.

**Definition 1** (Reparameterization Trick)**.** *Instead of sampling $z \sim \mathcal{N}(\mu, \sigma^2)$ directly, we write:*

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{24}$$

This transforms a stochastic node into a deterministic function of parameters plus independent noise, enabling gradient flow:

$$\frac{\partial z}{\partial \mu} = I \tag{25}$$

$$\frac{\partial z}{\partial \sigma} = \mathrm{diag}(\epsilon) \tag{26}$$

The expectation $\mathbb{E}_{q_\phi(z|x)}[f(z)]$ becomes $\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(\mu + \sigma \odot \epsilon)]$, and we can use Monte Carlo estimation with a single sample during training.

# 3 Model Architecture

## 3.1 Convolutional VAE Architecture

We implement a convolutional VAE suitable for the 28×28 grayscale MNIST images. The architecture follows an encoder-decoder structure with a 2-dimensional latent space for visualization purposes.

### 3.1.1 Encoder Network

The encoder maps input images to latent distribution parameters:

Table 1: Encoder Architecture

| Layer | Output Shape | Kernel | Stride |
|---|---|---|---|
| Input | $1 \times 28 \times 28$ | – | – |
| Conv2d + ReLU | $32 \times 14 \times 14$ | $4 \times 4$ | 2 |
| Conv2d + ReLU | $64 \times 7 \times 7$ | $4 \times 4$ | 2 |
| Conv2d + ReLU | $128 \times 4 \times 4$ | $3 \times 3$ | 2 |
| Flatten | 2048 | – | – |
| Linear $\rightarrow \mu$ | 2 | – | – |
| Linear $\rightarrow \log \sigma^2$ | 2 | – | – |

### 3.1.2 Decoder Network

The decoder maps latent vectors back to image space:

Table 2: Decoder Architecture

| Layer | Output Shape | Kernel | Stride |
|---|---|---|---|
| Input | 2 | – | – |
| Linear + ReLU | 2048 | – | – |
| Reshape | $128 \times 4 \times 4$ | – | – |
| ConvTranspose2d + ReLU | $64 \times 7 \times 7$ | $3 \times 3$ | 2 |
| ConvTranspose2d + ReLU | $32 \times 14 \times 14$ | $4 \times 4$ | 2 |
| ConvTranspose2d + Sigmoid | $1 \times 28 \times 28$ | $4 \times 4$ | 2 |

## 3.2 Conditional VAE Architecture

The CVAE extends the VAE by conditioning both encoder and decoder on class labels $c \in \{0, 1, \ldots, 9\}$.

### 3.2.1 Label Conditioning Mechanism

We incorporate labels through spatial broadcasting:

1. Convert label $c$ to one-hot vector $\mathbf{c} \in \mathbb{R}^{10}$
2. Broadcast to spatial dimensions: $\mathbf{C} \in \mathbb{R}^{10 \times H \times W}$
3. Concatenate with input: encoder receives $(1 + 10) \times 28 \times 28$

4. Decoder also receives label embedding concatenated with $z$

The modified generative model becomes:

$$z \sim p(z) = \mathcal{N}(0, I) \tag{27}$$
$$x \sim p_\theta(x|z, c) \tag{28}$$

And the CVAE objective:

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{q_\phi(z|x,c)}[\log p_\theta(x|z, c)] - \text{KL}(q_\phi(z|x, c)\|p(z)) \tag{29}$$

Note that we use a class-independent prior $p(z) = \mathcal{N}(0, I)$, which encourages the latent space to encode style variations independent of class identity.

# 4 Training Methodology

## 4.1 Loss Function

The VAE loss function combines reconstruction and regularization terms:

$$\mathcal{L}(\theta, \phi; x) = -\mathcal{L} = \underbrace{\text{BCE}(x, \hat{x})}_{\text{Reconstruction}} + \underbrace{\beta \cdot \text{KL}(q_\phi(z|x)\|p(z))}_{\text{Regularization}} \tag{30}$$

where BCE denotes binary cross-entropy (appropriate for normalized pixel values), and $\beta$ is a weighting factor.

## 4.2 KL Annealing

A common challenge in VAE training is **posterior collapse** [Bowman et al., 2016], where the model ignores the latent variable and the encoder produces $q_\phi(z|x) \approx p(z)$ regardless of $x$.

To address this, we implement KL annealing (also called $\beta$-scheduling):

$$\beta(t) = \min\left(1, \frac{t}{T_{\text{warmup}}}\right) \tag{31}$$

where $t$ is the current epoch and $T_{\text{warmup}}$ is the warmup period. This allows the model to first learn good reconstructions before gradually enforcing the prior constraint.

## 4.3 Training Configuration

Table 3: Training Hyperparameters

| Parameter | VAE | CVAE |
|---|---|---|
| Latent dimension | 2 | 2 |
| Batch size | 128 | 128 |
| Learning rate | $10^{-3}$ | $10^{-3}$ |
| Optimizer | Adam | Adam |
| Epochs | 30 | 30 |
| KL warmup epochs | 10 | 10 |

### 4.4 Monitoring Training

We track three metrics during training:

1. **Reconstruction Loss**: BCE between input and reconstruction
2. **KL Divergence**: Regularization term (before $\beta$ weighting)
3. **Reconstruction/KL Ratio**: Indicator of balance; values 10-50 suggest healthy training

## 5 Experiments and Results

### 5.1 Dataset

We use the MNIST dataset [LeCun et al., 1998] consisting of 60,000 training and 10,000 test images of handwritten digits (0-9). Images are 28×28 grayscale, normalized to $[0, 1]$.

### 5.2 VAE Training Results

#### 5.2.1 Training Dynamics

Figure 1 shows the training curves for the standard VAE. Key observations:

- **Reconstruction loss** decreases steadily from  160 to  138, indicating improving reconstruction quality.
- **KL divergence** shows the characteristic "bump" during warmup (epochs 1-10) as $\beta$ increases, then stabilizes around 6.6.
- **Recon/KL ratio** of  20 indicates balanced training without posterior collapse.

#### 5.2.2 Reconstruction Quality

Visual inspection of reconstructions shows that the VAE captures the essential structure of digits while smoothing fine details—a characteristic of VAE reconstructions due to the variational regularization.

#### 5.2.3 Latent Space Visualization

With a 2D latent space, we can directly visualize the learned representation (Figure 2). The latent space shows clear clustering by digit class, with semantically similar digits (e.g., 4 and 9, 3 and 8) positioned nearby. This organization emerges purely from the reconstruction objective without any label supervision.

#### 5.2.4 Latent Traversals

By fixing one latent dimension and varying the other across $[-3, 3]$, we observe smooth morphological changes (Figure 3):

- $z_1$ (horizontal) primarily controls digit slant and rotation
- $z_2$ (vertical) influences stroke thickness and overall scale

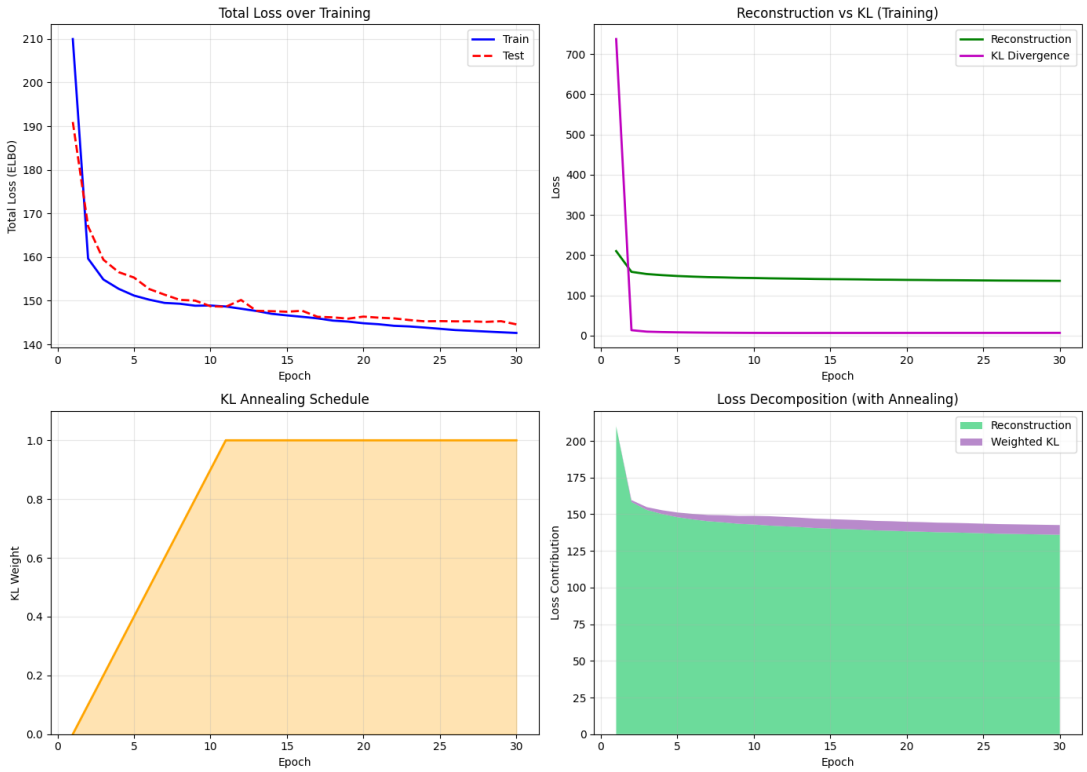


Figure 1: VAE training curves showing reconstruction loss (left), KL divergence (center), and their ratio (right) over 30 epochs. The KL annealing warmup (epochs 1-10) is visible in the KL curve.

#### 5.2.5 Latent Interpolations

Linear interpolation between encoded digits produces semantically meaningful transitions (Figure 4). For example, interpolating from a "3" to an "8" shows intermediate forms that resemble plausible hybrid digits.

### 5.3 CVAE Training Results

#### 5.3.1 Training Dynamics

The CVAE achieves notably better metrics:

- **Final reconstruction loss**: 124.36 (vs 138.00 for VAE)
- **Final KL divergence**: 4.80 (vs 6.59 for VAE)
- **Final total loss**: 129.16 (vs 144.59 for VAE)

The improvement is expected: by providing class labels, the decoder doesn't need to infer digit identity from $z$, allowing the latent space to focus purely on within-class variations.

#### 5.3.2 Conditional Generation

The CVAE enables controlled generation: given any latent vector $z$ and class label $c$, we can generate a digit of class $c$ with style determined by $z$. Figure **??** shows 10 samples for each digit class, demonstrating consistent class identity with natural style variation.

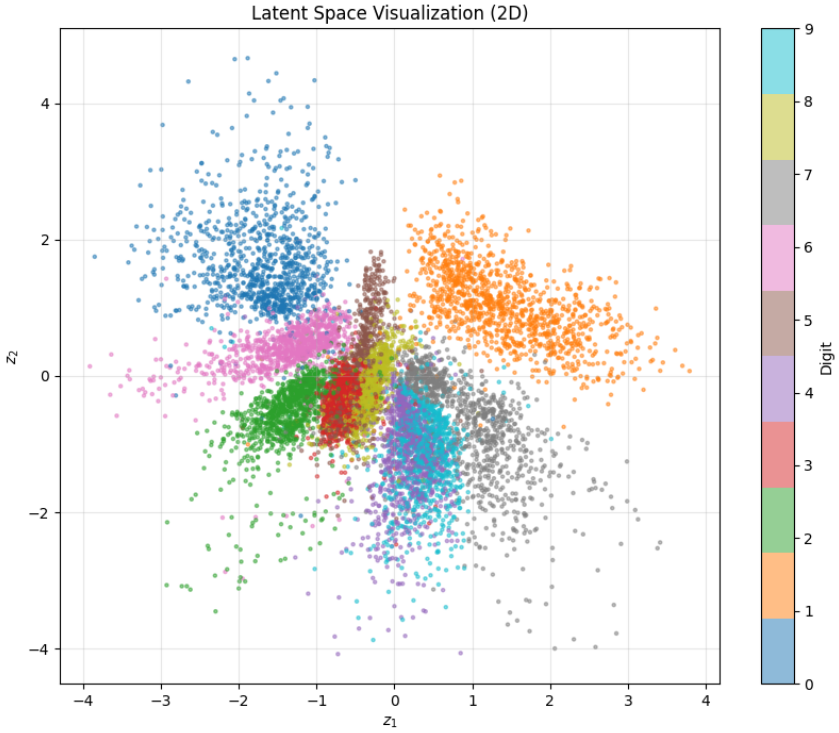


Figure 2: Visualization of the 2D latent space for 5000 MNIST test samples, colored by digit class. Note the natural clustering by class without any label supervision.

#### 5.3.3 Latent Space Structure

Unlike the VAE's clustered latent space, the CVAE's latent space shows **mixed class distributions**. This is the desired behavior with a class-independent prior: the latent space encodes class-agnostic features (style, stroke characteristics) while class information is provided separately.

#### 5.3.4 Disentanglement Demonstration

A key advantage of CVAEs is explicit disentanglement of class identity from other factors of variation. We demonstrate this in Figure 5 with three experiments:

1. **Fixed $z$, varying class**: The same latent vector generates different digits when conditioned on different labels, with consistent style across classes.
2. **Style transfer**: A latent vector extracted from one digit's style can be applied to generate any other digit class with that style.
3. **Class-conditional traversals**: Latent traversals for a fixed class show morphological variations (slant, thickness) while maintaining class identity perfectly.

### 5.4 Quantitative Comparison

The CVAE shows consistent improvements across all metrics. The larger relative improvement in KL divergence (27.2%) suggests that conditioning allows the encoder to learn a more compact representation.

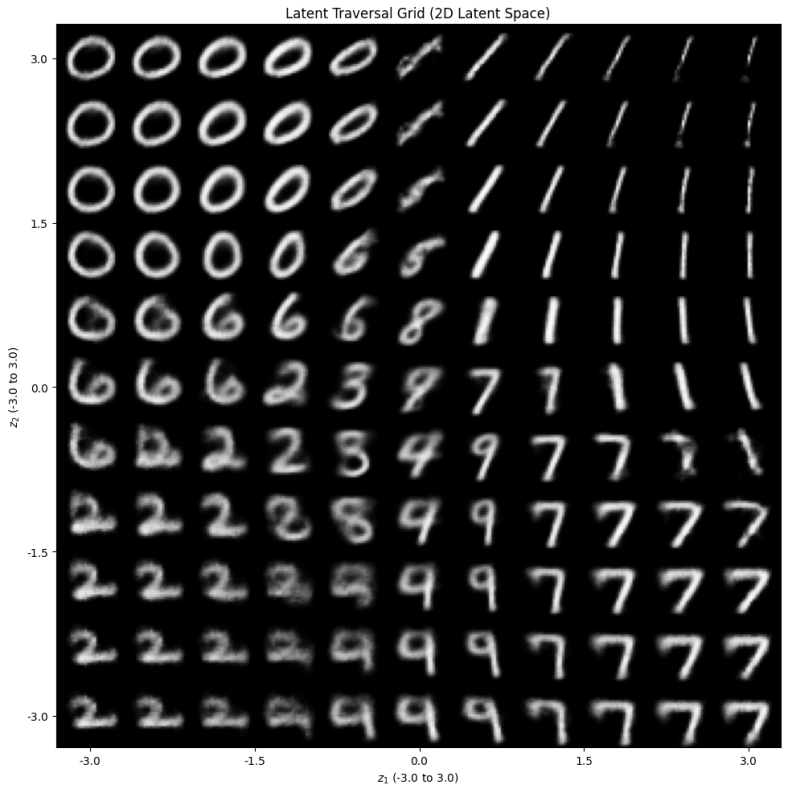


Figure 3: Latent space traversal grid. Each row corresponds to a fixed $z_2$ value, each column to a fixed $z_1$ value, ranging from -3 to +3.

Table 4: Final Training Metrics Comparison

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

# 6 Discussion

## 6.1 Key Findings

1. **KL annealing is essential**: Without gradual warmup, early training showed signs of posterior collapse with KL divergence dropping to near zero.

2. **2D latent space enables visualization but limits capacity**: While useful for visualization, a 2D latent space constrains the model's representational power. Production VAEs typically use 64-256 dimensions.

3. **CVAE achieves superior reconstruction through task decomposition**: By separating "what digit" (label) from "how it looks" (latent), the CVAE can allocate all latent capacity to style variations.

4. **Latent space structure reflects conditioning**: VAE latent spaces naturally cluster by class; CVAE latent spaces mix classes because labels provide class information externally.

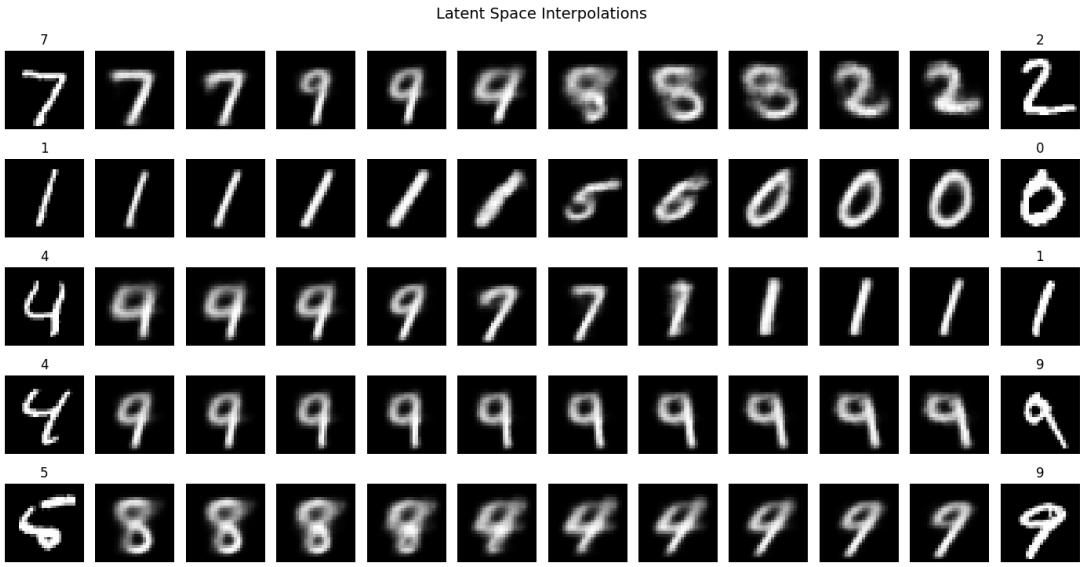


Figure 4: Linear interpolations in latent space between pairs of encoded digits. Each row shows 10 steps from the source digit (left) to the target digit (right).

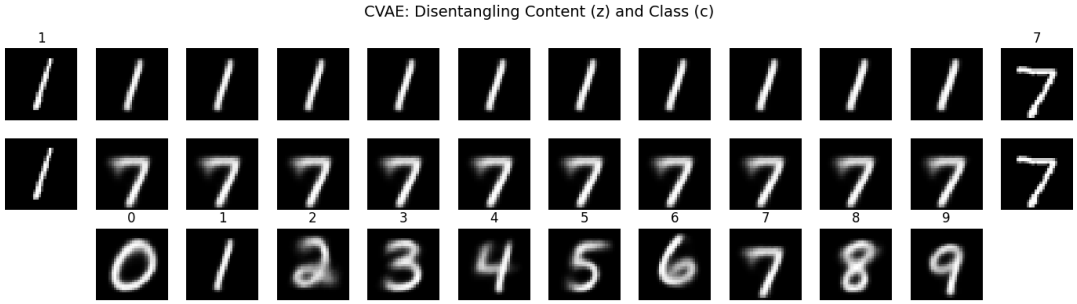


Figure 5: CVAE disentanglement demonstration. Top rows: same latent interpolation with different class labels (1 vs 7). Bottom row: fixed latent vector $z$ applied to all digit classes 0-9, showing style transfer.

### 6.2 Limitations

1. **Reconstruction blurriness**: VAE reconstructions are characteristically blurry compared to input images, a known limitation of pixel-wise reconstruction losses.

2. **Limited latent dimensions**: Our 2D space cannot capture all factors of variation in handwritten digits.

3. **Binary assumption**: Using BCE loss assumes binary pixel values, whereas real-valued assumptions (Gaussian decoder) might be more appropriate.

4. **Architectural choices**: We did not extensively tune architecture or hyperparameters; better results are achievable with deeper networks or different configurations.

### 6.3 Relation to Other Work

Our implementation follows the original VAE formulation [Kingma and Welling, 2014]. The CVAE extension follows Sohn et al. [2015]. KL annealing was proposed by Bowman et al. [2016] for sequence VAEs but applies broadly.

Recent advances include:

- $\beta$**-VAE** [Higgins et al., 2017]: Uses $\beta > 1$ for better disentanglement
- **VQ-VAE** [van den Oord et al., 2017]: Discrete latent spaces
- **Hierarchical VAEs** [Vahdat and Kautz, 2020]: Multi-scale latent hierarchies

# 7 Conclusion

This project presented a comprehensive study of Variational Autoencoders, from theoretical derivation to practical implementation. We derived the ELBO from first principles, implemented convolutional architectures for both VAE and CVAE, and demonstrated the importance of KL annealing for stable training.

Our experiments on MNIST showed that:

1. Standard VAEs learn semantically meaningful latent spaces that cluster by digit class
2. Latent traversals reveal interpretable factors of variation
3. CVAEs achieve better reconstruction (9.9% improvement) by decomposing class identity from style
4. CVAEs enable explicit control over generated digit class while preserving style variations

## 7.1 Future Directions

Several extensions could improve upon this work:

1. **Higher-dimensional latent spaces**: Increase latent dimension and use t-SNE/UMAP for visualization
2. $\beta$**-VAE exploration**: Systematically vary $\beta$ to study disentanglement-reconstruction tradeoff
3. **Alternative architectures**: Explore ResNet encoders/decoders or attention mechanisms
4. **Different datasets**: Apply to more complex datasets (CIFAR-10, CelebA)
5. **Quantitative disentanglement metrics**: Implement metrics like DCI or $\beta$-VAE metric

# Acknowledgments

This work was completed as part of the Generative AI and Diffusion Models course. We thank the course instructors for guidance and the PyTorch team for their excellent deep learning framework.

# References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 10–21, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. $\beta$-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, 2020.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

# A Implementation Details

## A.1 PyTorch Code: VAE Loss Function

```
def vae_loss(x, x_recon, mu, logvar, beta=1.0):
    # Reconstruction: Binary Cross Entropy
    recon_loss = F.binary_cross_entropy(
        x_recon, x, reduction='sum') / x.size(0)

    # KL Divergence: Closed form for Gaussian
    kl_loss = -0.5 * torch.sum(
        1 + logvar - mu.pow(2) - logvar.exp()
    ) / x.size(0)

    total_loss = recon_loss + beta * kl_loss
    return total_loss, recon_loss, kl_loss
```

## A.2 Reparameterization Implementation

```
def reparameterize(self, mu, logvar):
    std = torch.exp(0.5 * logvar)
    eps = torch.randn_like(std)
    return mu + eps * std
```