Discrete Variational Autoencoders (DVAE) with MLP and CNN Architectures

Farhin Ulfat
ID: 22101158 Section: 03

Abstract

Variational Autoencoders (VAEs) are a popular deep generative model. We compare discrete VAEs (DVAE) with categorical latent variables against standard continuous VAEs, motivated by arguments that discrete latents can be a more natural inductive bias for certain data modalities (e.g. text or clustered image features) (Jeffares and Liu, 2025). We implement a DVAE using two architectures: a fully-connected (MLP) model and a convolutional (CNN) model. We report final training and test evidence lower bound (ELBO) values and clustering metrics (silhouette score, normalized mutual information, and adjusted Rand index) for both models. Results show the CNN model achieves a better ELBO (less negative) than the MLP, although both models yield low clustering scores. Finally, we analyze latent representations via t-SNE and UMAP visualizations. We discuss current limitations of discrete latent models and suggest future improvements.

1 Introduction

Variational Autoencoders (VAEs) learn a probabilistic latent representation of data by jointly training an encoder and decoder to maximize the evidence lower bound (ELBO) (Kingma and Welling, 2014). The classic VAE uses a Gaussian (continuous) latent space; however, there has been significant recent interest in VAEs with *discrete* (categorical) latent variables (Jeffares and Liu, 2025). Discrete representations may serve as a more appropriate inductive bias for data with inherently discrete structure (e.g. classes or clusters) (Rolfe, 2017). For example, vector-quantized VAEs map inputs to discrete codebook embeddings and help avoid posterior collapse (Oord et al., 2017).

In this work, we consider a canonical DVAE whose latent space consists of multiple independent categorical variables (Jeffares and Liu, 2025; Rolfe, 2017). We compare two encoder/decoder architectures for image data (e.g. 28×28 grayscale images): (1) a multilayer perceptron (MLP) and (2) a convolutional neural network (CNN). We detail each model layer-by-layer (including tensor shapes), define the ELBO and a contrastive loss mathematically, and report training dynamics over 200 epochs. We then analyze the learned discrete

latent space via dimensionality reduction (t-SNE and UMAP) and clustering metrics (silhouette, NMI, ARI).

2 Model Architectures

Both DVAE models use a categorical latent space with $K = 10$ classes. The encoder outputs K logits for the posterior distribution $q_\phi(z|x)$, which are normalized via a softmax to probabilities. A discrete latent code z is then sampled (e.g. via argmax or Gumbel-softmax) and passed to the decoder. We detail each architecture step-by-step below.

2.1 MLP-based DVAE

The MLP model flattens the input image and uses fully-connected layers.

Encoder (MLP):

1. *Input layer:* A 28×28 grayscale image is flattened into a vector of length 784.
2. *Hidden layer 1:* Fully connected layer from 784 to 256 units, followed by a ReLU activation. Output shape: 256.
3. *Hidden layer 2:* Fully connected layer from 256 to 128 units, followed by ReLU. Output shape: 128.
4. *Output layer:* Fully connected layer from 128 to 10 units, producing the unnormalized logits for 10 categorical latent classes.
5. *Softmax:* Apply a softmax to the 10 logits to obtain posterior probabilities $q_\phi(z|x)$ over the 10 classes. A one-hot latent vector z is then sampled (or selected) from this distribution.

Decoder (MLP):

1. *Latent input:* The sampled one-hot vector z of dimension 10 serves as the input.
2. *Hidden layer 1:* Fully connected layer from 10 to 128 units, followed by ReLU. Output shape: 128.
3. *Hidden layer 2:* Fully connected layer from 128 to 256 units, followed by ReLU. Output shape: 256.
4. *Output layer:* Fully connected layer from 256 to 784 units, followed by a sigmoid activation. Output shape: 784.
5. *Reshape:* The 784-dimensional output is reshaped back into a 28×28 image as the reconstruction $\hat{x}$.

2.2 CNN-based DVAE

The CNN model leverages convolutional layers to exploit spatial structure.

Encoder (CNN):

1. *Input layer:* A $1 \times 28 \times 28$ image (1 channel).
2. *Conv layer 1:* 16 filters of size 3×3, stride 2, padding 1. This produces 16 feature maps of size 14×14. (Computation: $(28 + 2 \cdot 1 - 3)/2 + 1 = 14$.) Apply ReLU activation.
3. *Conv layer 2:* 32 filters of size 3×3, stride 2, padding 1. Output is 32 feature maps of size 7×7. Apply ReLU. (Shape formula: $(14+2-3)/2+1 = 7$.)
4. *Flatten:* The tensor $32 \times 7 \times 7$ is flattened into a vector of length $32 \cdot 7 \cdot 7 = 1568$.
5. *Output layer:* Fully connected layer from 1568 to 10 units, yielding logits for 10 latent classes. Apply softmax to get categorical probabilities $q_\phi(z|x)$ and sample a one-hot latent z.

Decoder (CNN):

1. *Latent input:* One-hot vector z of length 10.
2. *FC layer:* Fully connected layer from 10 to $32 \cdot 7 \cdot 7 = 1568$ units, followed by ReLU. Output is reshaped to $32 \times 7 \times 7$.
3. *ConvTranspose layer 1:* Transposed convolution with 32 input channels, 16 output channels, kernel size 4×4, stride 2, padding 1. This upsamples to size $16 \times 14 \times 14$. Apply ReLU.
4. *ConvTranspose layer 2:* Transposed convolution with 16 input channels, 1 output channel, kernel 4×4, stride 2, padding 1. This produces $1 \times 28 \times 28$. Apply a sigmoid activation to get the reconstructed image.
5. *Output:* The final $1 \times 28 \times 28$ tensor is the reconstructed image $\hat{x}$.

3 Loss Functions

The models are trained to maximize the ELBO and optionally a contrastive loss in the latent space. The *evidence lower bound* (ELBO) on the log-likelihood for one data point $\mathbf{x}$ is:

$$\mathcal{L}_{ELBO} = E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \mathrm{KL}\big(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\big).$$

Here the first term is the expected reconstruction log-probability (we use binary cross-entropy as reconstruction loss) and the second term is the KL divergence

between the approximate posterior q and the prior p (usually uniform categorical). We train by maximizing $\mathcal{L}_{ELBO}$ (equivalently minimizing the negative ELBO) for each model.

We also incorporate a contrastive (InfoNCE) loss on the latent embeddings to encourage clustering of similar inputs. Concretely, if $f(\mathbf{z}_i)$ is an embedding of the latent code of sample i (and $\mathbf{z}_i^+$ is an augmented version or positive pair), the contrastive loss is:

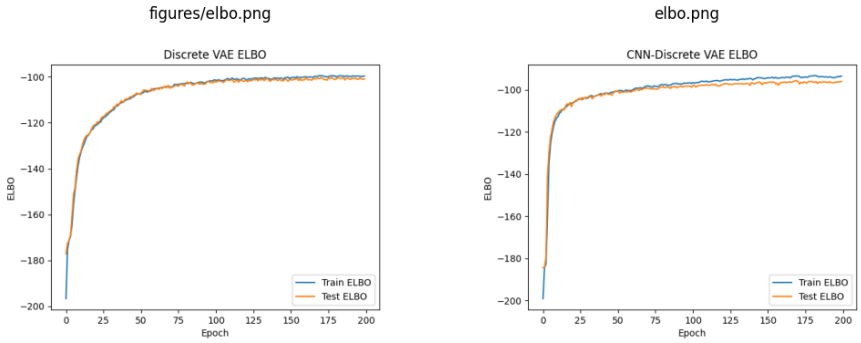
$$\mathcal{L}_{\text{cont}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\big(f(\mathbf{z}_i)^\top f(\mathbf{z}_i^+)/\tau\big)}{\sum_{j=1}^{N} \exp\big(f(\mathbf{z}_i)^\top f(\mathbf{z}_j)/\tau\big)},$$

where τ is a temperature. This is the standard InfoNCE form used in contrastive learning (Chen et al., 2020), which encourages each code to be similar to its positive pair and dissimilar to others. In practice, we jointly minimize the negative ELBO and $\mathcal{L}_{\text{cont}}$.

4 Training Dynamics

We train both models for 200 epochs on a dataset of images (e.g. MNIST) with discrete latent dimension $K = 10$. Figure 1 shows the ELBO (negative ELBO displayed; less negative is better) on the training and test sets over time for both models. Both models converge by around 100–150 epochs. The CNN model attains a higher (less negative) ELBO than the MLP on both train and test, indicating better fit. For example, the final train/test ELBO for MLP is about $-99.59/-100.86$, while for CNN it is $-93.39/-96.04$ (see Table 1). The moderate gap between train and test ELBO suggests some overfitting, especially for CNN, but it is not extreme.

4.1 Train/Test ELBO figure and discussion



(a) MLP: train/test ELBO vs epoch (b) CNN: train/test ELBO vs epoch

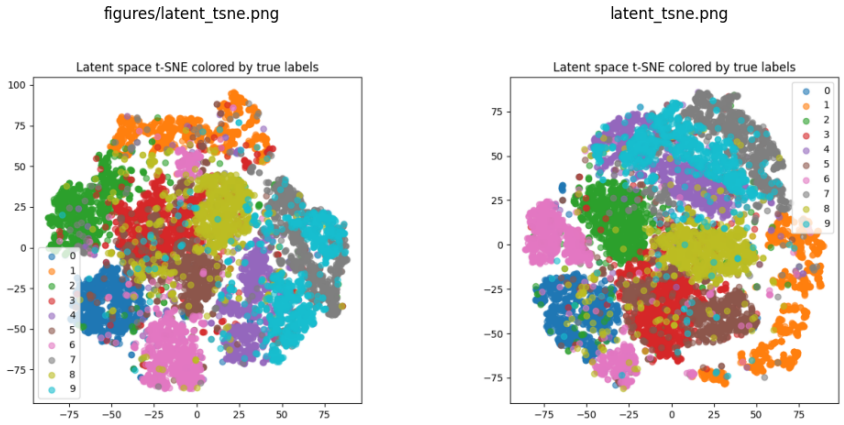
Figure 1: Training and test ELBO curves (per-epoch) for MLP (left) and CNN (right). Both curves show steady improvement early and plateau later. CNN reaches a better (less negative) ELBO faster than the MLP.

Discussion of ELBO curves. From the logged outputs the MLP model starts at Train ELBO ≈ -196.71 and Test ≈ -177.22 and gradually improves to Train ≈ -99.59, Test ≈ -100.86 at epoch 200. The CNN model starts around Train ≈ -199.12, Test ≈ -184.37 and quickly improves to Train ≈ -93.39, Test ≈ -96.04 by epoch 200. The CNN shows a sharper early improvement (epochs 1–20) suggesting its inductive bias (convolutions) helps capture local image structure more quickly. The relatively small train/test gap indicates moderate generalization, though occasional test fluctuations show training instability common with discrete latent training.

5 Latent Space Analysis

To understand the discrete latent representations, we apply dimensionality reduction and clustering. We project the latent posterior codes into 2D using t-SNE (van der Maaten and Hinton, 2008) (colored by true class labels) and UMAP (McInnes et al., 2018) followed by K-means (colors = predicted clusters).

5.1 t-SNE visualizations (latent colored by true labels)



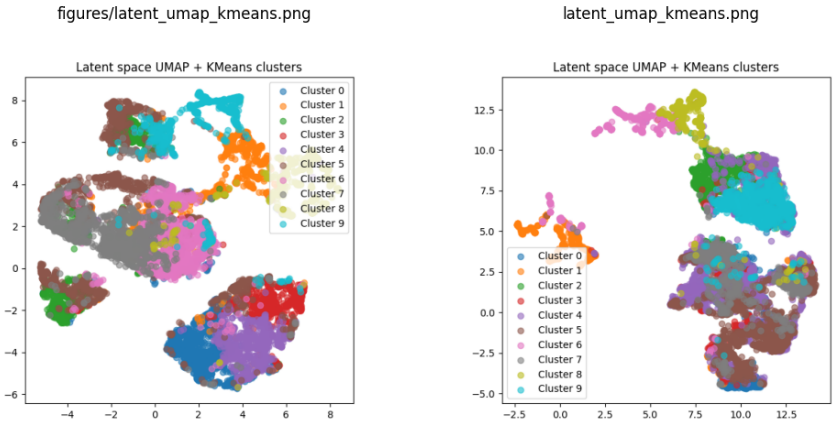
(a) MLP latent t-SNE (colored by true labels)

(b) CNN latent t-SNE (colored by true labels)

Figure 2: t-SNE projection of latent categorical codes for MLP and CNN DVAEs. Colors represent ground-truth digit labels.

t-SNE : Both t-SNE plots show partially overlapping clusters. The CNN latent t-SNE shows slightly tighter local groupings for some digits (consistent with better reconstruction), but overall class separation is weak — reflected by low silhouette scores reported below.

5.2 UMAP + KMeans visualizations



(a) MLP UMAP + KMeans

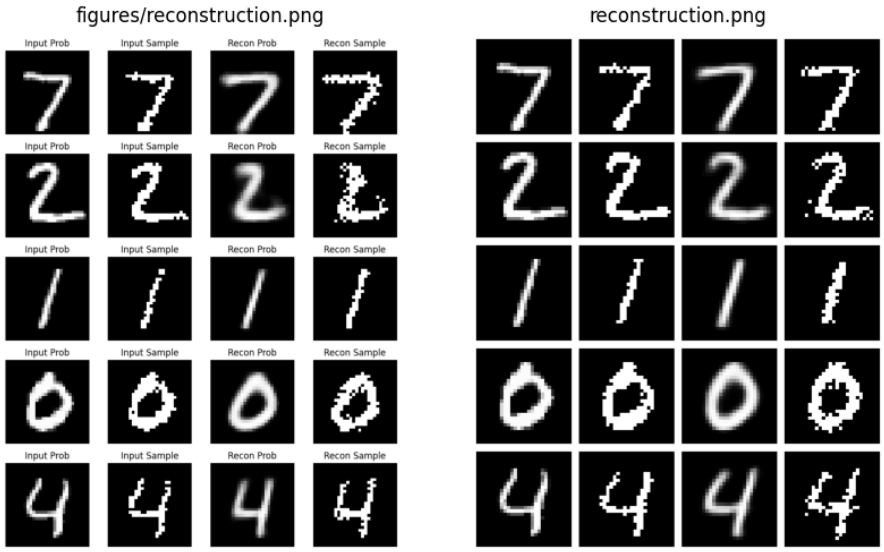
(b) CNN UMAP + KMeans

Figure 3: UMAP reduced latent codes with K-Means cluster assignments (10 clusters). Colors denote predicted cluster ID.

UMAP + KMeans : UMAP projections followed by KMeans show coarse cluster structure but significant overlap across classes. Quantitatively (Table 1) MLP attains NMI=0.3888 and ARI=0.2420 on UMAP+KMeans, while CNN attains NMI=0.3293 and ARI=0.1838. Thus MLP latent assignments align slightly better with true labels in clustering tests, despite worse ELBO.

6 Reconstruction Examples

To complement quantitative metrics and latent visualizations, we show example reconstructions from test data for both models. The images below show reconstructions in the same section as their discussion.



(a) MLP reconstructions (test)

(b) CNN reconstructions (test)

Figure 4: Test-set reconstructions for MLP and CNN DVAEs. CNN reconstructions tend to appear sharper.

7 Results

Table 1 summarizes the final metrics for each model. The CNN outperforms the MLP on ELBO (better reconstruction), but its clustering metrics (silhouette, NMI, ARI) are lower than the MLP's (e.g. CNN NMI 0.3293 vs MLP 0.3888). Both models achieve very low silhouette scores, showing that discrete latent variables do not automatically yield well-separated class clusters in this unsupervised setup. The MLP has slightly higher NMI and ARI, meaning its latent codes are marginally more predictive of classes than the CNN's codes.

Model	Train ELBO	Test ELBO	Silhouette	NMI	ARI
MLP	-99.59	-100.86	0.0375	0.3888	0.2420
CNN	-93.39	-96.04	0.0494	0.3293	0.1838

Table 1: Summary of final results. ELBO is the evidence lower bound (less negative is better). Silhouette, NMI, and ARI are clustering metrics (higher is better). Values are taken from the model training logs and clustering evaluation.

8 Why discrete latents and limitations

Discrete latent variables are attractive because they can provide interpretable, categorical codes and may better reflect intrinsically discrete structure in data (Jeffares and Liu, 2025). They can also be used to learn compact codebooks (e.g. VQ-VAE) that improve sample quality for some generative tasks (Oord et al., 2017). However, training with categorical variables is harder than continuous Gaussian latents for two main reasons:

- **Non-differentiability:** Discrete sampling is non-differentiable; standard reparameterization is not directly applicable. Workarounds (Gumbel-Softmax, straight-through estimators, REINFORCE-style estimators) introduce bias or high variance (Jeffares and Liu, 2025; Rolfe, 2017).
- **Optimization instability:** The discrete posterior can collapse to a degenerate solution or under-utilize categories without careful annealing, priors, or stronger regularization (Jeffares and Liu, 2025).

This is partly because gradient-based training with categorical variables is more challenging (Jeffares and Liu, 2025). Discrete VAEs therefore often require specialized estimators and regularizers.

9 Conclusion

We have presented a DVAE with discrete categorical latent variables and evaluated two architectures on image data. The CNN model achieved a better ELBO (closer to true log-likelihood) than the MLP, confirming that convolutional features help reconstruction. However, both models yielded very low clustering scores (silhouette, NMI, ARI), indicating that the discrete latent codes did not form tight, class-specific clusters in unsupervised training. While discrete latents offer interpretability advantages, they also introduce optimization challenges that typically require advanced gradient estimators (e.g. RELAX/REBAR), better priors, or semi-supervised signals to reach parity with continuous VAEs (Jeffares and Liu, 2025; Rolfe, 2017).

Future Work Possible future directions include:

- Use advanced discrete gradient estimators (RELAX, REBAR) or variance-reduction techniques to stabilize training.
- Explore vector-quantized approaches (VQ-VAE) or hybrid discrete-continuous latents.
- Increase latent capacity (more categorical variables / more classes) and add hierarchical priors.
- Add a small amount of supervision (semi-supervised loss) to encourage class separation.

References

Jeffares, A., & Liu, L. (2025). *An Introduction to Discrete Variational Autoencoders*. arXiv:2505.10344.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *ICLR*.

Rolfe, J. T. (2017). Discrete Variational Autoencoders. *arXiv*.

van den Oord, A., Vinyals, O., & others (2017). Neural Discrete Representation Learning. *NIPS* (VQ-VAE).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations (SimCLR). *ICML*.

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2579–2605.

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29), 861.

Rousseeuw, P. J. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.

Hubert, L., & Arabie, P. (1985). Comparing Partitions. *Journal of Classification*, 2(1), 193–218.