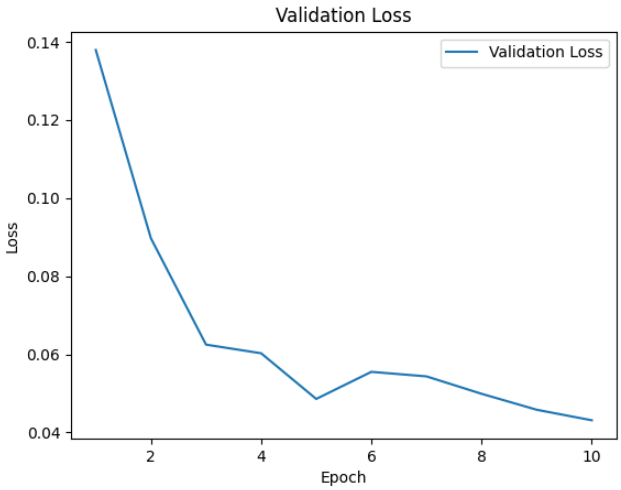
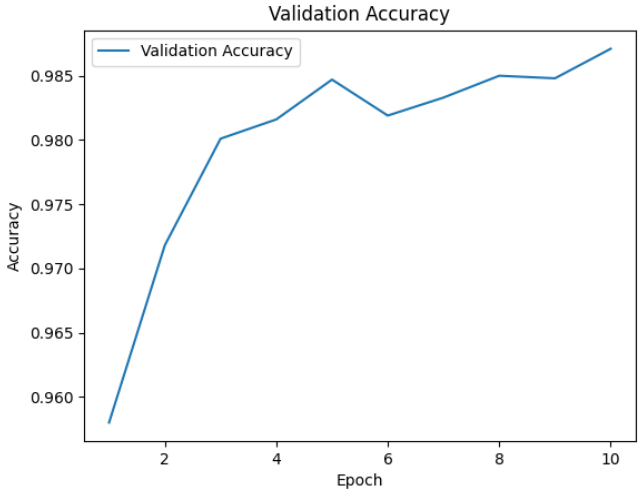
Advanced Machine Learning – Ex1

Sec. 2

Plots are also available in the sup_classification.ipynb notebook.

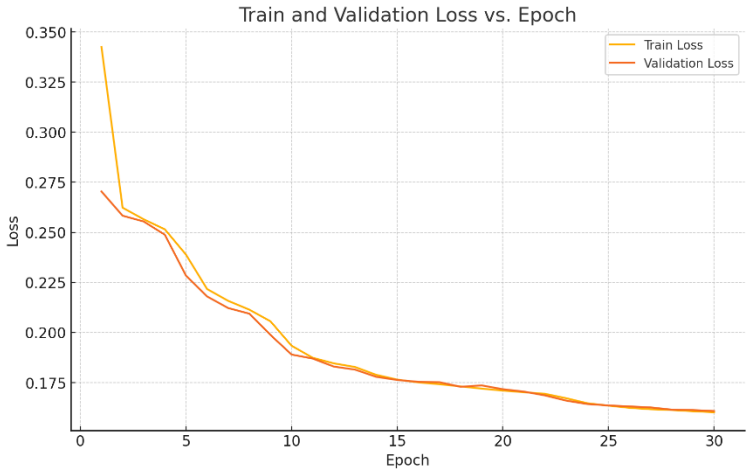
Epoch 10/10, Training Loss: 0.0133, Validation Loss: 0.0497, Validation Accuracy: 98.43%





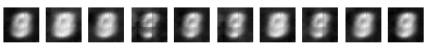
Sec. 3

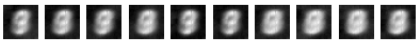
Q1: Amortized VAE.

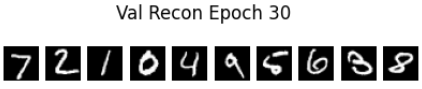


- Both train and validation losses steadily decrease and remain very close to each other throughout training.
- There is no divergence (i.e. val loss does not start rising while train loss keeps falling), which would signal overfitting.

Reconstructions for epochs 1,5,10,20,30. Train vs. Validation:



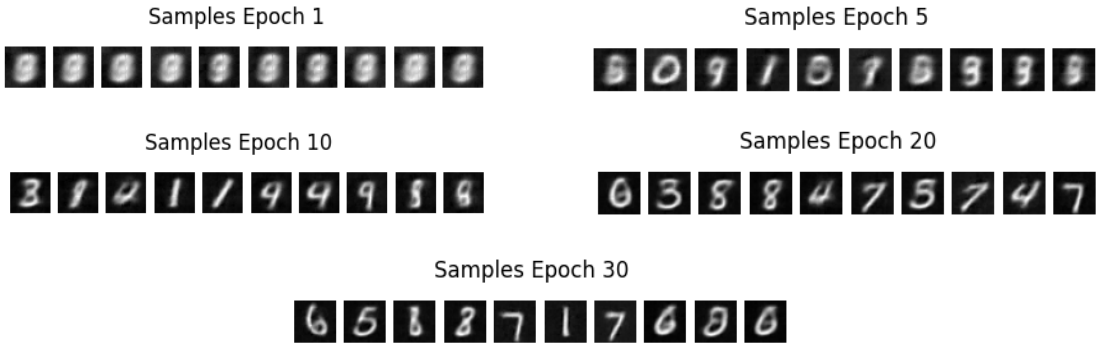




- Early epochs (1-5) produce very blurry reconstructions.
- By epoch 10, digit shapes become recognizable, and by 20-30 the reconstructions are sharp and clean.
- The validation reconstructions improve in lockstep with the training ones, showing similar quality gains.

No major overfitting is seen. The fact that validation loss tracks training loss so closely, coupled with parallel improvements in train and validation reconstructions, indicates the model is not overfitting. Instead, it's learning a good shared latent representation that generalizes well from epoch 1 through epoch 30.

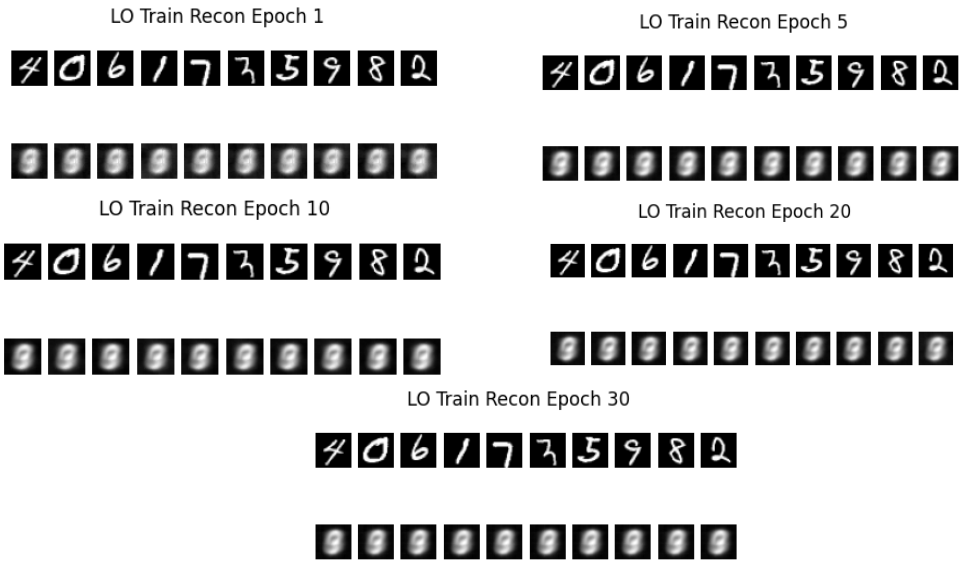
Q2: Sampling from a VAE.



The samples go from noise → rough strokes → clearly legible, diverse MNIST-style digits.

Q3: Latent Optimization.

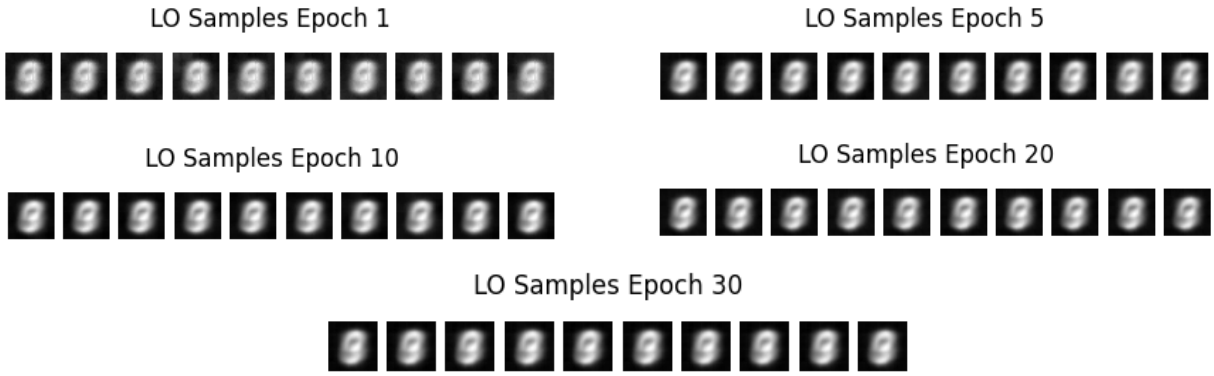
Reconstructions of 10 images (one from each class) from the training set:



The amortized-inference reconstructions from Q1 show each of the ten digits crisp, class-correct, and well-shaped by epoch 30. By contrast, the **LO** reconstructions are all the same blurry gray blob every time.

The amortized encoder learns a network that maps each image into its own region of latent space. Those q vectors preserve class-specific structure, so decoding faithfully recreates each digit.

The per-example latent-optimization vectors collapse. Without a shared encoder guiding them apart, every μ_i drifts to the single easiest "mode" to decode (a gray average of all digits). As a result, the LO method's q vectors carry no class information.



LO samples remain virtually identical at all epochs. The decoder outputs the same blurry "9"-like blob every time, with no digit diversity or improvement.

Amortized samples (Q2) progress to crisp, varied MNIST digits ("0–9") by epoch 30.

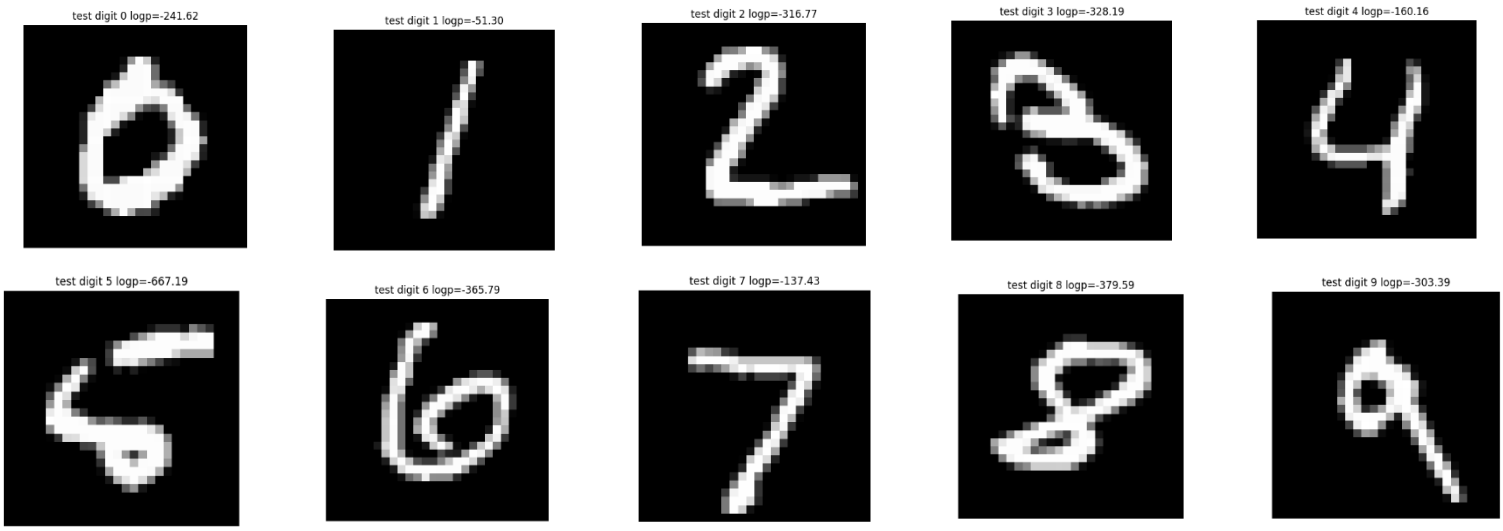
We can see that the initialization was not sufficient because the LO decoder only ever saw the optimized $u_i s$ (and **never** random latent draws during training), it never learned to map a standard normal $N(0, I)$ into the full multi-modal digit manifold.

All random z collapse to the one "mode" the decoder happens to reconstruct best (a generic blurry digit), so sampling yields no variety.

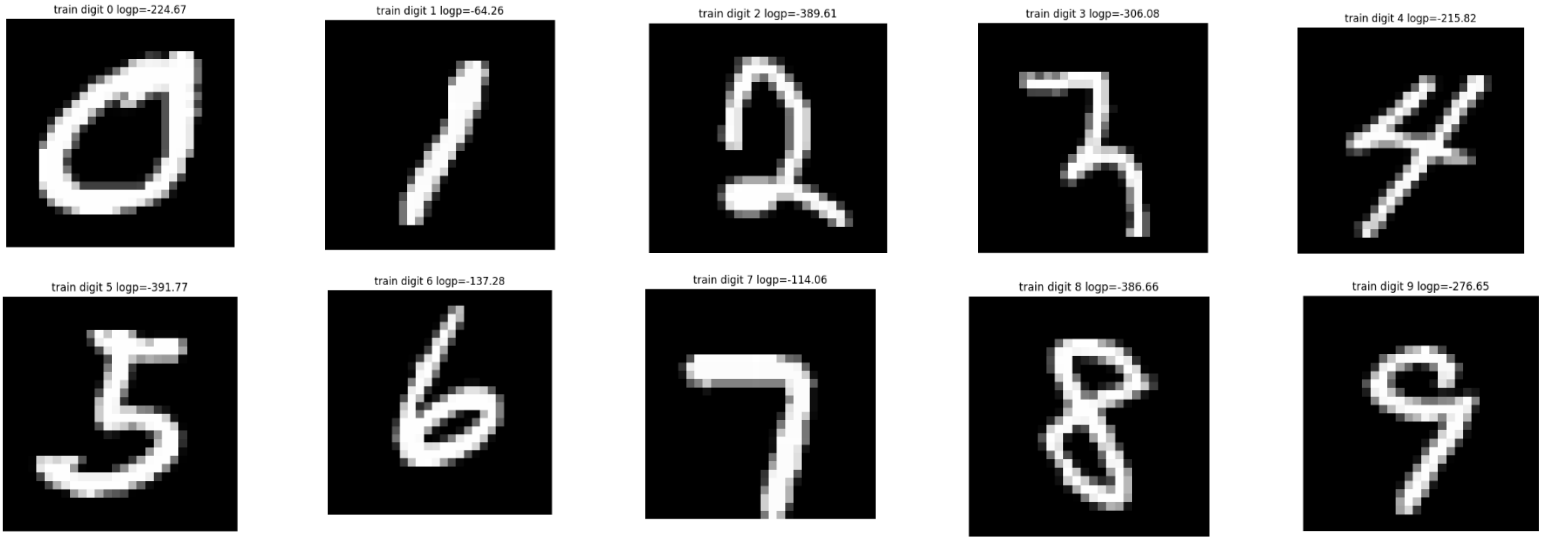
Thus, the LO initialization and training procedure **did not** establish a good prior distribution.

Q4: Computing the log-probability of an image.

<u>Test:</u>



Train:



Most likely digit: “1” has the highest mean log-prob (≈ –57.8), by a comfortable margin.

It makes sense because “1” is structurally the simplest (a single straight stroke), so the VAE can reconstruct it with very low reconstruction error and a modest KL term.

More complex digits (loops in “5”, “8”, curves in “2”) incur larger reconstruction penalties and thus lower ELBOs.

$$average\log p(train) = -250.7\ ;\ average\log p(test) = -295.1$$

Training-set images receive **higher** ELBOs (are assigned more probability) than test-set images. This is expected because the VAE’s parameters (encoder+decoder) were optimized to maximize ELBO on the training data, making those samples "easier" for the model to encode and reconstruct. Test images, being unseen, incur slightly higher reconstruction+KL loss and thus lower average log-prob.

4.1.1 Github Copilot & ChatGPT

I wrote the code using Cursor IDE, which supported me throughout the implementation of the exercise. Its built-in knowledge of PyTorch, matplotlib, and scikit-learn helped me develop the models and generate the plots for each section. When I encountered gaps in my understanding of certain topics, I turned to ChatGPT for explanations and to refine my answers in more formal and articulate language.