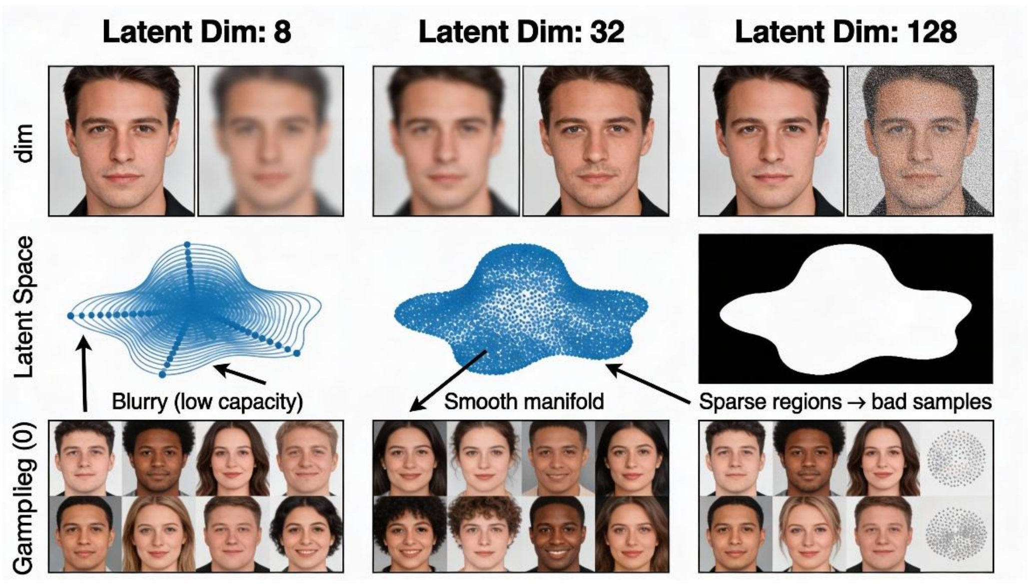
1. Latent Dimension Choice

a. How does the size of the latent space affect reconstruction quality and generative diversity?

- **Reconstruction Quality:** A larger latent dimension increases the capacity of the information bottleneck, allowing the VAE to preserve more high-frequency details (sharpness) and reconstruct the input more accurately. A smaller dimension forces higher compression, often resulting in blurry reconstructions as fine details are discarded.
- **Generative Diversity**: A smaller latent dimension forces the model to learn meaningful, continuous data manifolds (better disentanglement), often leading to smoother interpolations. Too large a dimension can lead to overfitting, where the latent space becomes sparse (empty gaps between points). This hurts generative quality because sampling from "empty" regions in the prior p(z) yields noisy or unrealistic images.



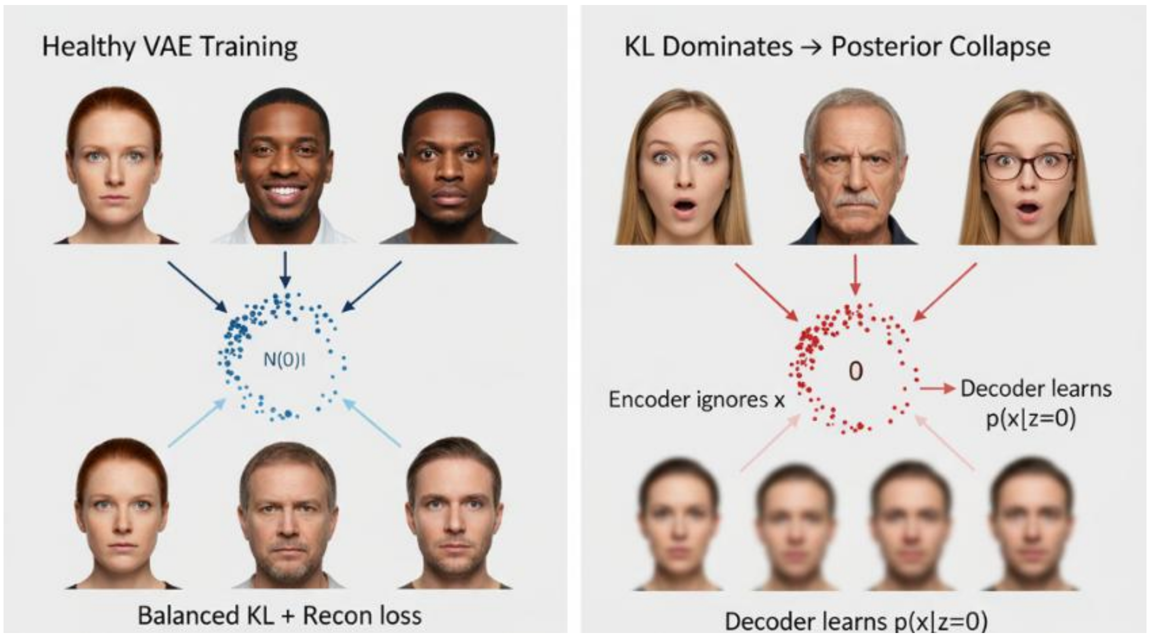
b. What practical signs indicate that your latent dimension is too small or too large?

- **Too Small:** The reconstruction loss remains high; generated images are unrecognizable or excessively blurry; the model fails to capture distinct features (e.g., all faces look the same).
- **Too Large:** The reconstruction is near-perfect (pixel-for-pixel), but random samples from the prior look like static noise or do not resemble faces. This suggests the model is acting like a standard Autoencoder, mapping inputs to scattered points rather than a dense distribution.

2. KL vs. Reconstruction Trade-off

a. During training, what does it mean if the KL term dominates the loss?

If the KL divergence term is too strong (or the reconstruction term is too weak), the Encoder is forced to map every **input (x)** to the exact **prior distribution N (0, I)** regardless of the image content. The Decoder effectively receives pure noise and learns to ignore the **latent code (z)**, outputting the "average" image of the dataset for every input. This phenomenon is known as **Posterior Collapse**.



b. How would you adjust your training strategy to address this?

- **KL Annealing:** Start training with the KL weight Beta = 0(treating it like a standard Autoencoder to learn good reconstructions first) and linearly increase Beta to 1 over a number of epochs.
- **Beta-VAE:** Treat Beta as a hyperparameter and tune it (e.g., set Beta < 1) to balance the trade-off.
- **KL Thresholding:** modifying the loss function to not penalize the KL term until it exceeds a certain target value.

3. Data Preprocessing Impact

a. Why is image alignment and cropping critical for CelebA when training VAEs?

VAEs are not translation-invariant. If the eyes are at pixel (50, 50) in one image and (60, 60) in another, the model must learn to predict "eyes everywhere," which wastes capacity. Alignment ensures that features (such as eyes, nose, and mouth) appear in predictable locations, allowing the model to focus on learning semantic variations (like glasses and hair color) rather than their spatial position.

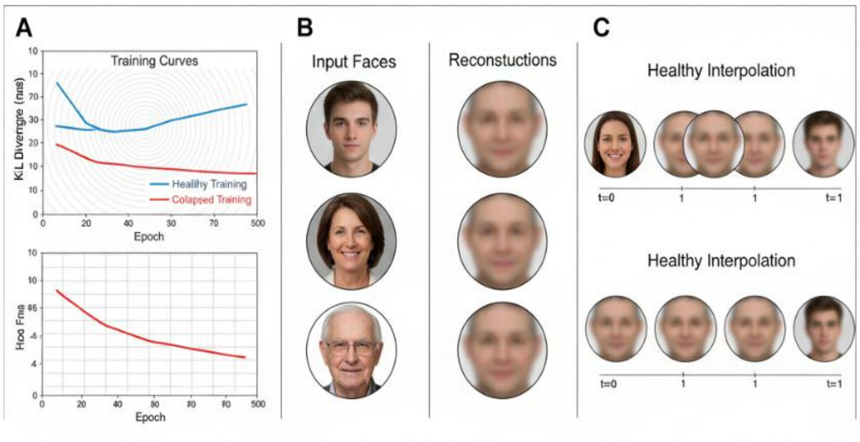
b. What could happen if you skip normalization or resizing?

- **Skipping Normalization:** If pixels are (0, 255) instead of (0, 1) or (-1, 1), gradients can explode, causing training instability. Furthermore, the activation functions (like Tanh or Sigmoid) and the loss function assumptions (MSE vs. BCE) require specific input ranges to function correctly.
- **Skipping Resizing:** processing full-resolution images (e.g., 1024x1024) increases computational cost exponentially, likely causing Out-Of-Memory (OOM) errors and drastically slowing down training without adding semantic value for a basic VAE.

4. Training Stability

What symptoms indicate posterior collapse during training?

1. **KL Loss = approx. 0:** The KL term drops effectively to zero very early in training.
2. **Identical Outputs:** The decoder produces the same image (usually a blurry, average face) regardless of the input image or latent vector supplied.
3. **Latent Code Ignored:** Interpolating between two z vectors results in no change in the output image.



Symptom 1: KL Loss ≤ 0 Symptom 2: Identical Outputs Colapom 3: Latent Code Ignored

5. Latent Space Manipulation

How can you estimate directions in latent space that correspond to semantic attributes (e.g., “Smiling”)?

You perform vector arithmetic using the labelled data.

1. Compute the average latent vector of all images labelled "Smiling": $\mu_{smiling}$
2. Compute the average latent vector of all images labelled "Not Smiling": $\mu_{neutral}$
3. The attribute vector is the difference:

   $$d_{smile} = \mu_{smiling} - \mu_{neutral}$$

4. To add a smile to a new face z, you calculate

   $$z_{new} = z + \alpha \cdot d_{smile}$$