Variational Auto Encoders (VAE)

Karthik Inbasekar

16-10-2025

MoonMath.ai

Autoencoders

- An auto encoder is a pair of neural networks designed to learn an identity function without supervision.

Hinton & Salakhutdinov 2006

- Purpose: Compress data, and recover, learn efficient compression of data
  - Encoder network: High dimension input $x \in \mathbb{R}^n$ to latent code output $\mathbf{z} \in \mathbb{R}^k$, $k < n$

$$\mathbf{z} = g_\phi(\mathbf{x})$$

  - Decoder network: Recover high dim input from the low dim latent code

$$\mathbf{x}' = f_\theta g_\phi(\mathbf{x})$$

- Reconstruction loss

$$L(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (x_i - f_\theta g_\phi(x_i))^2$$

Video Autoencoders

- Input dimension: $\mathbf{x} \in \mathbb{R}^{B \times C \times T \times H \times W}$, Output dimension: $\mathbf{z} \in \mathbb{R}^{B \times C' \times T' \times H' \times W'}$
- Compression ratio: $C_t \times C_f \times C_f$ where $C_t = T/T'$, $C_f = H/H' = W/W'$
- Eg: LTX : has a compression 8 x 32 x 32 pixels per token (patchify size)

$3 \times 64 \times 512 \times 512$

Pixel space volume: 50331648

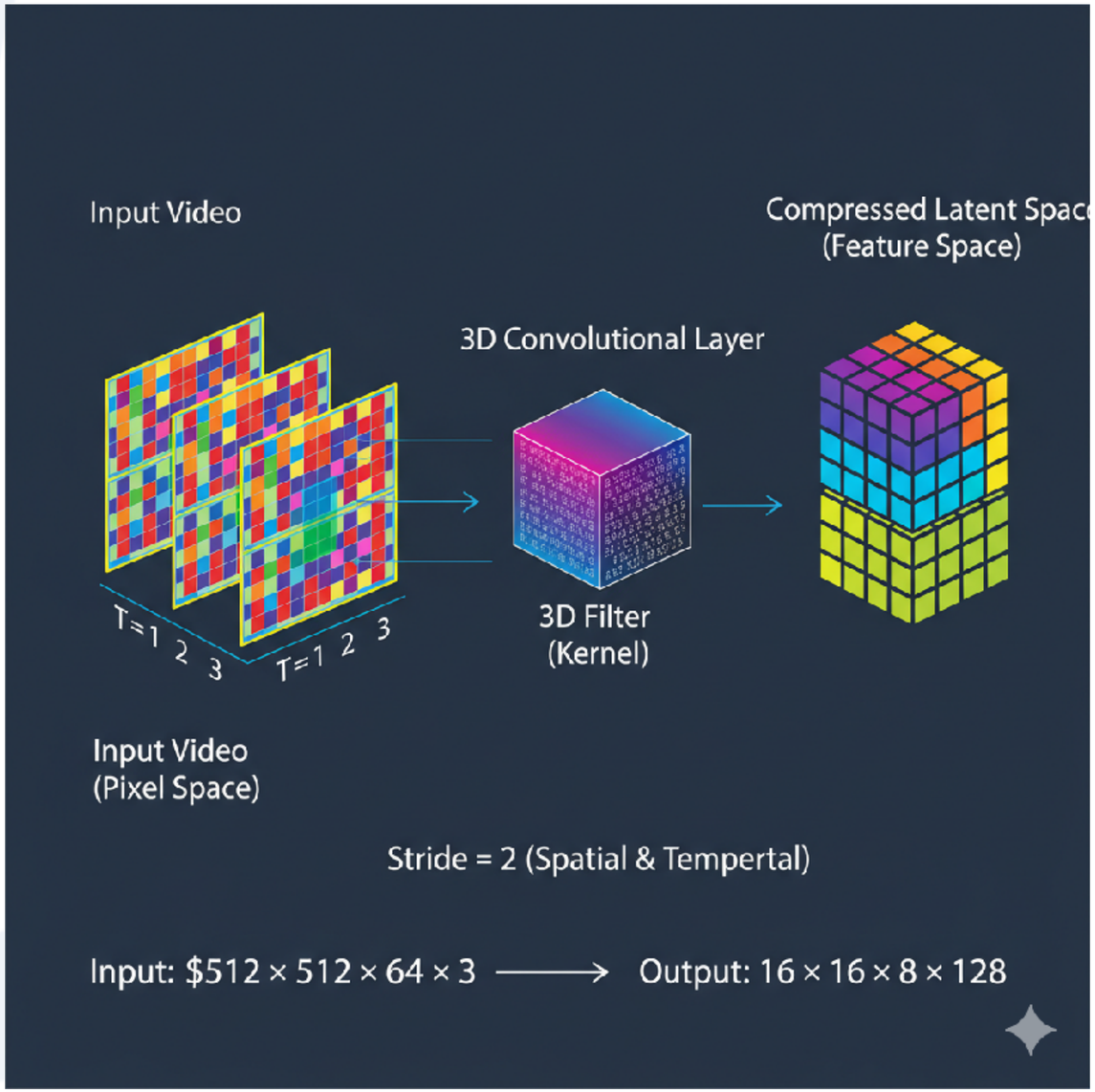
$128 \times 8 \times 16 \times 16$

Latent space volume: 262144

Compression ratio 1:192

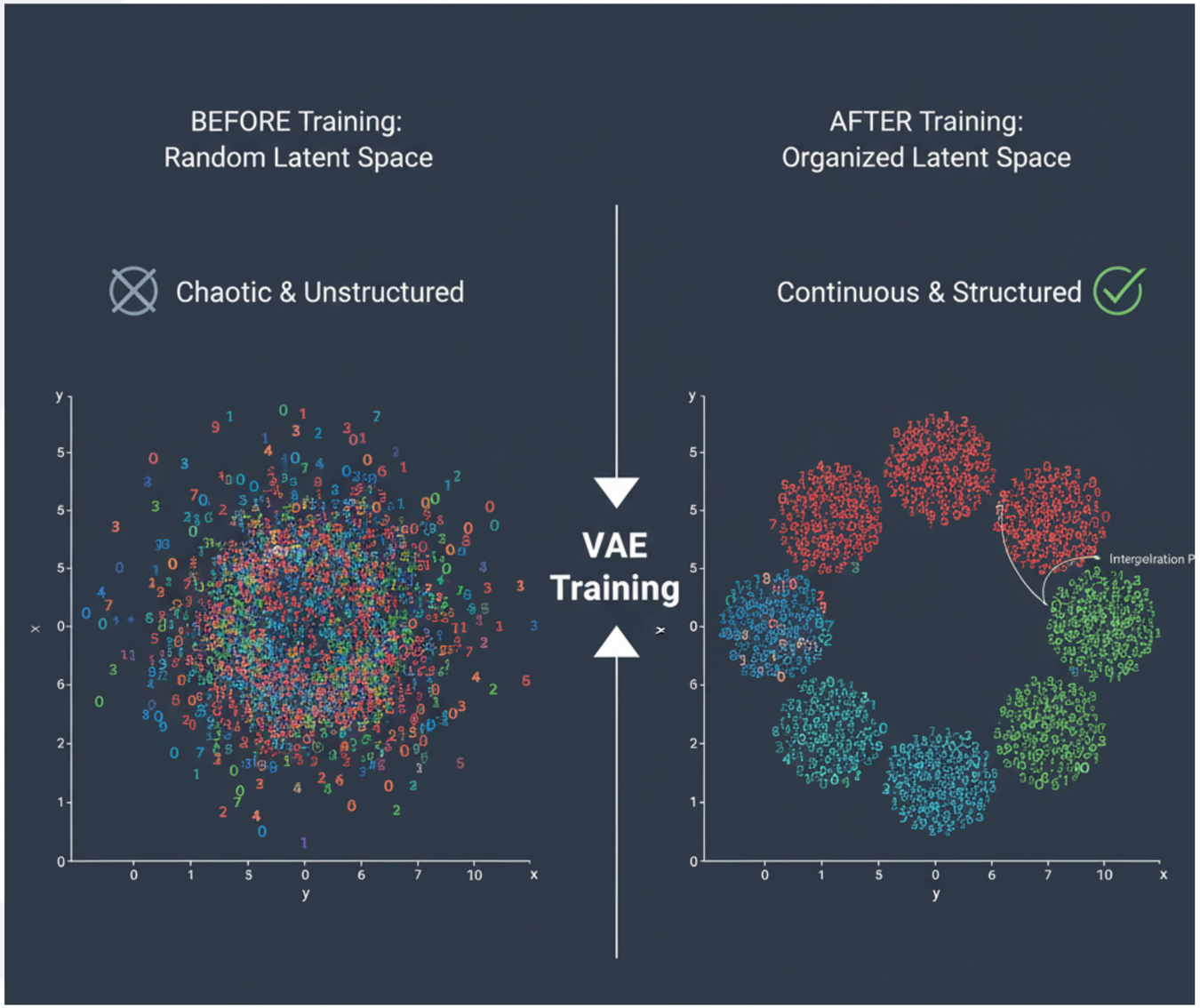
Video Autoencoders

- Initial layer: CNN to say 32 channels
- Progressive expansion: Subsequent layers of encoder have more channels till desired number (128 say for the final layer) is met
- Increasing num channels: trade spatial temporal resolution to feature depth
- Allows retention of abstract features even with aggressive compression.
- Suspicion: In AR this exact idea contributes to the increased error accumulation. LTX AR very bad.

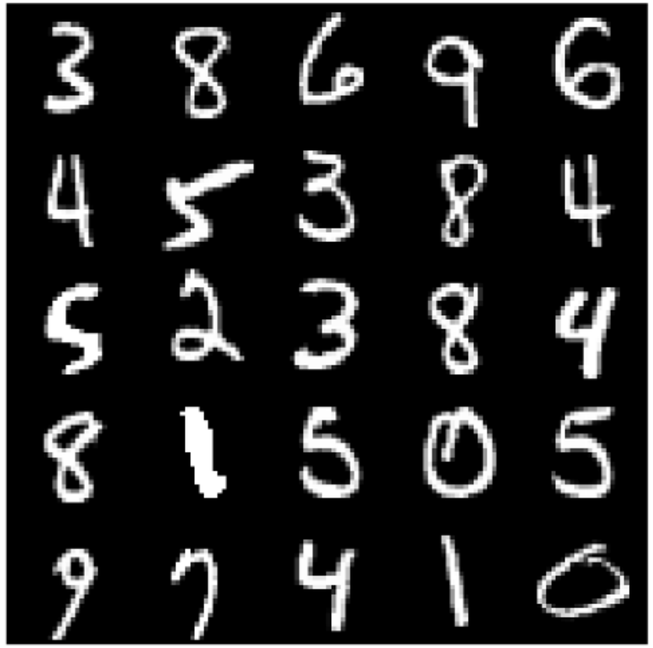


Data organization in latent space

- Every data (image/video) is a feature vector in the space $\mathbf{z} \in \mathbb{R}^{T' \times C' \times H' \times W'}$
- If we have a bunch of data, and we encoded them into latent space, it will look like a disorganized mess
- As the training progresses, data of similar structure (latent vector) gets mapped to similar regions in latent space forming clusters.
- However, in an ordinary auto encoder, there is no smooth interpolation across data clusters. Hint: reconstruction loss (MSE)

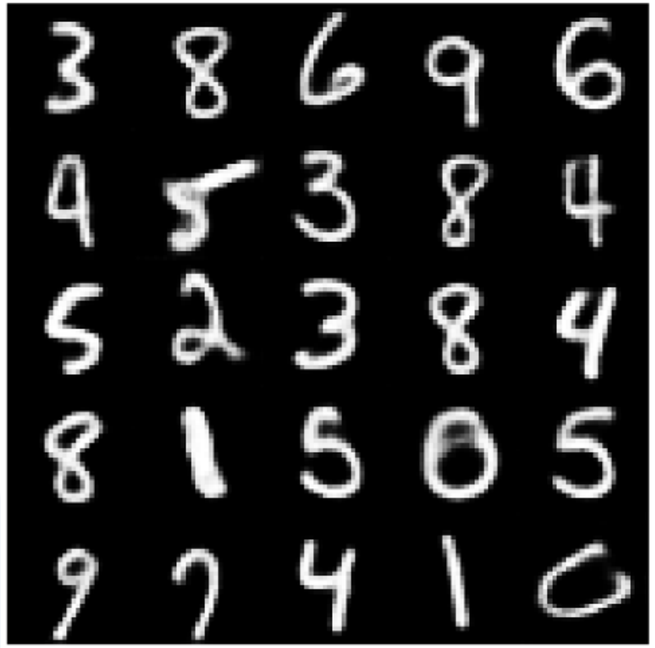


Autoencoders : Eg(MNIST example)



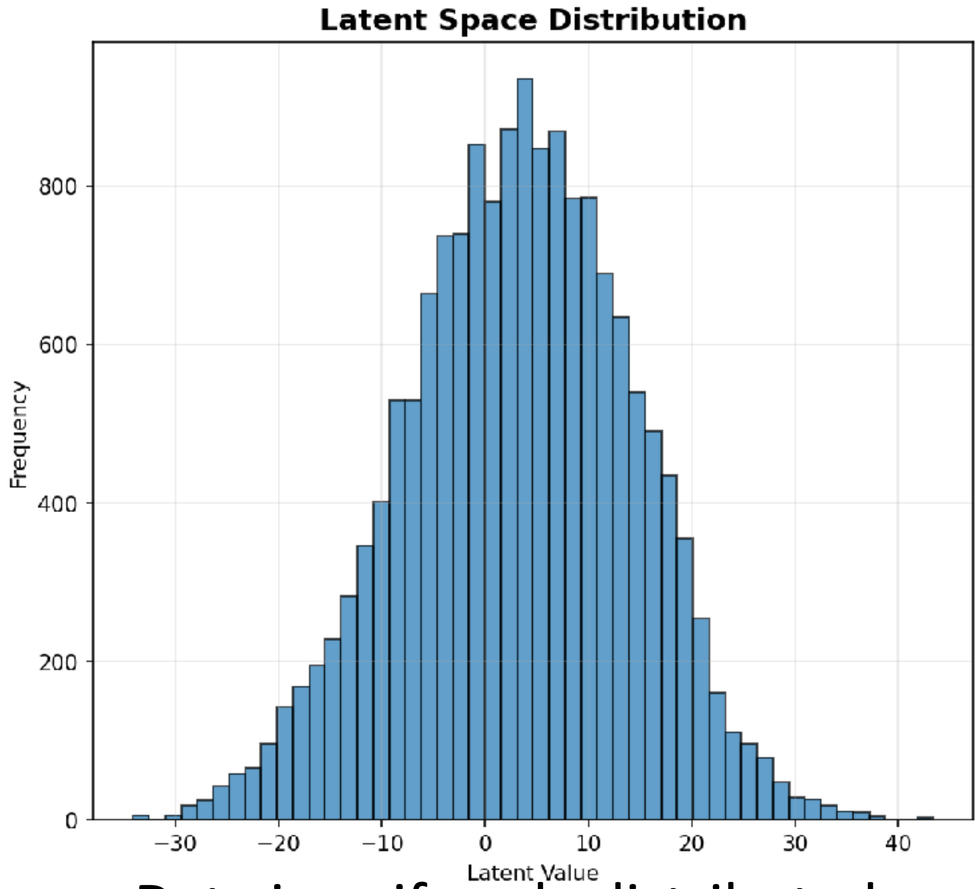
Original image set

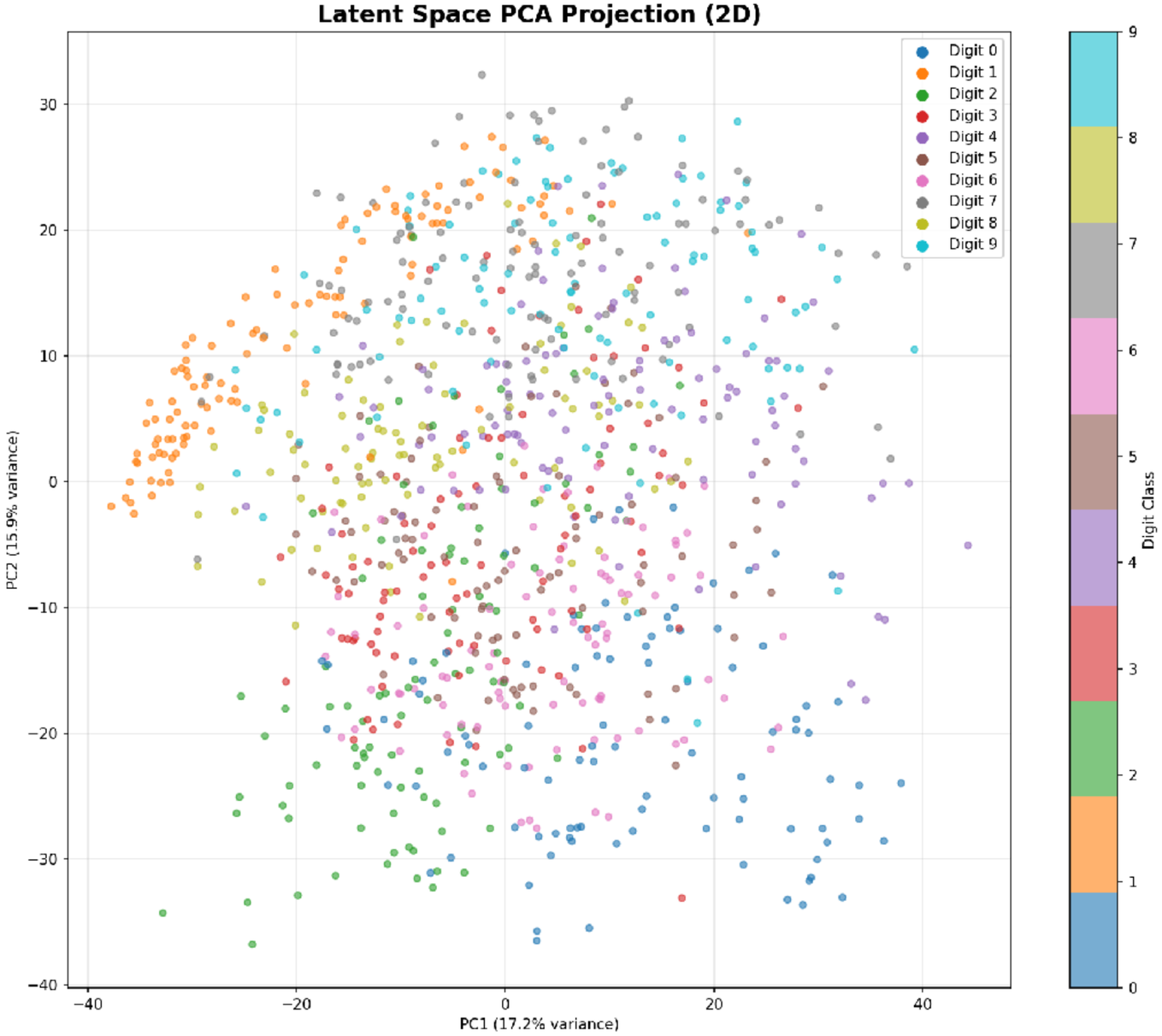
```
📊 MODEL ARCHITECTURE:
--------------------------------------------------
Encoder parameters:          394,896
Decoder parameters:          339,009
Total trainable parameters: 733,905
Estimated model size:         2.80 MB

🔢 LATENT SPACE:
--------------------------------------------------
Input dimensions:            (1, 28, 28) = 784 values
Latent dimensions:           16

📉 COMPRESSION:
--------------------------------------------------
Compression ratio:           49.00x
Compression percentage:      97.96%
Information retention:       2.04%

🔍 ANALYZING LATENT SPACE...
--------------------------------------------------
Encoded 1000 images
Latent codes shape:          (1000, 16)
Latent mean:                 3.1545
Latent std:                  10.9917
Latent min:                  -34.0851
Latent max:                  43.3592
```



Reconstructed image

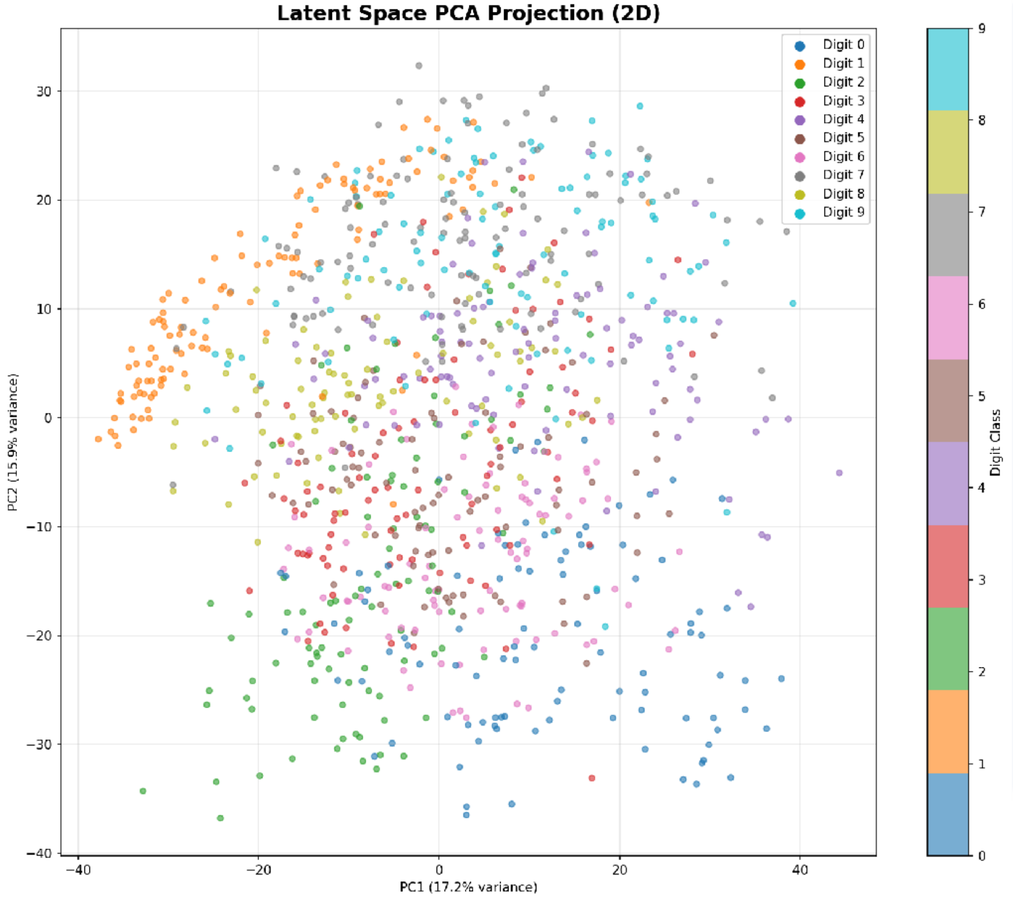
Autoencoders : Latent space organization

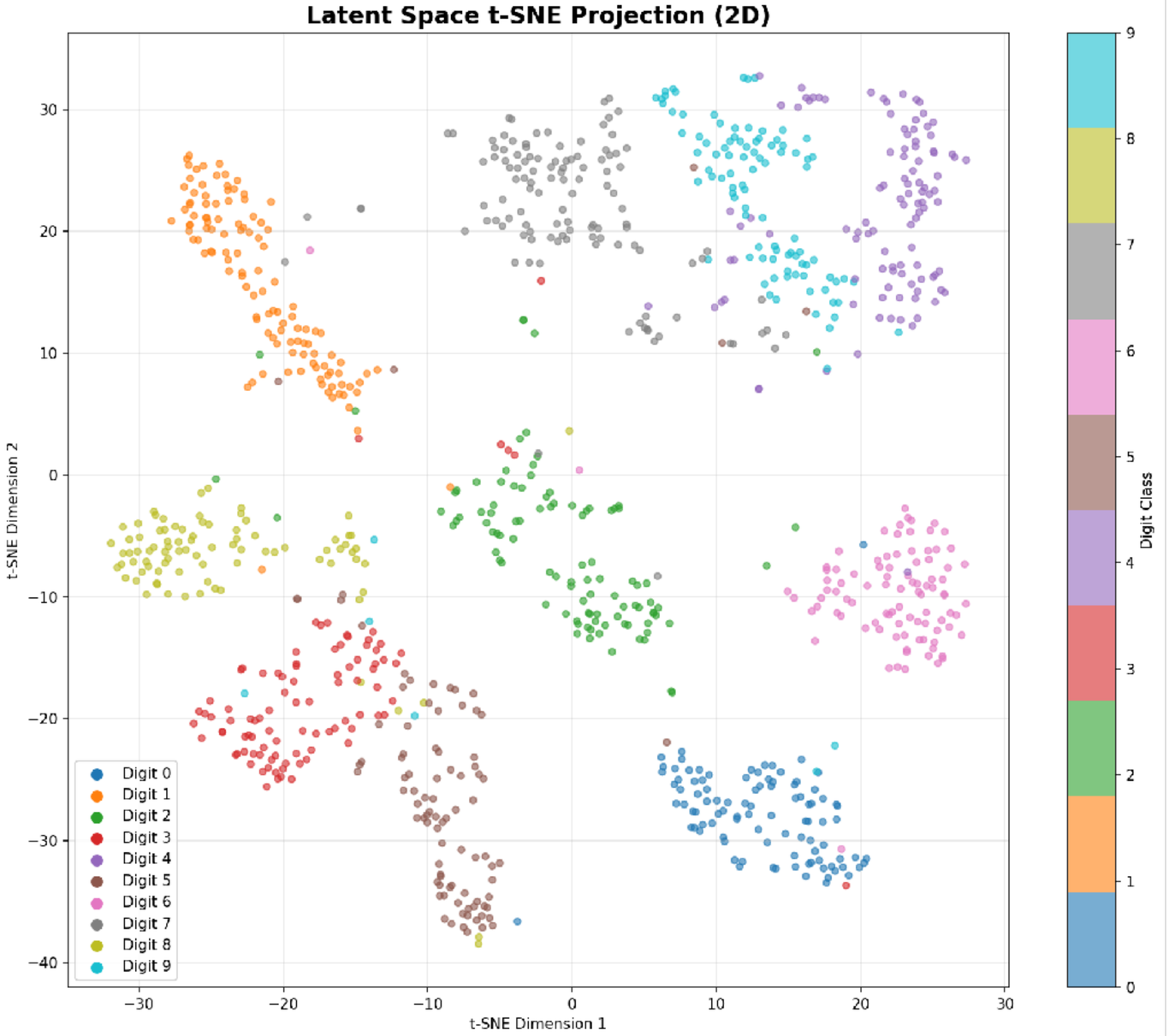




- Data is uniformly distributed, and has a clustering (not so visible in the simple PCA projection)

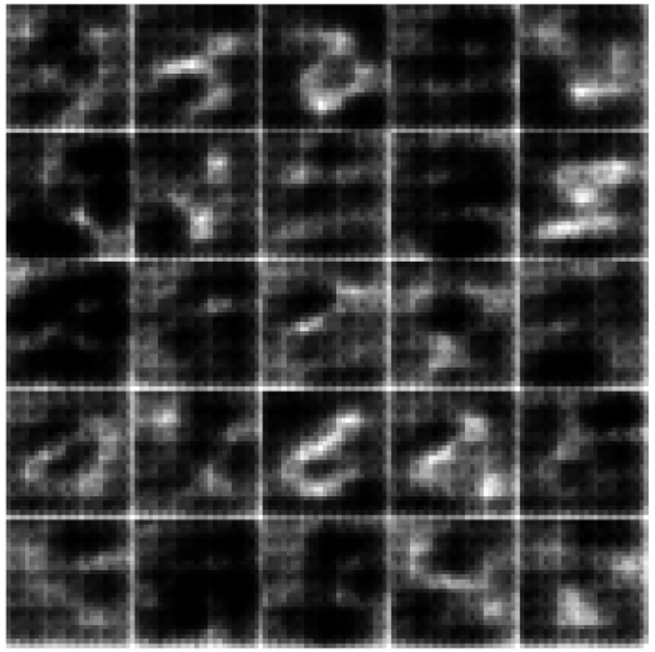
Autoencoders : Latent space organization



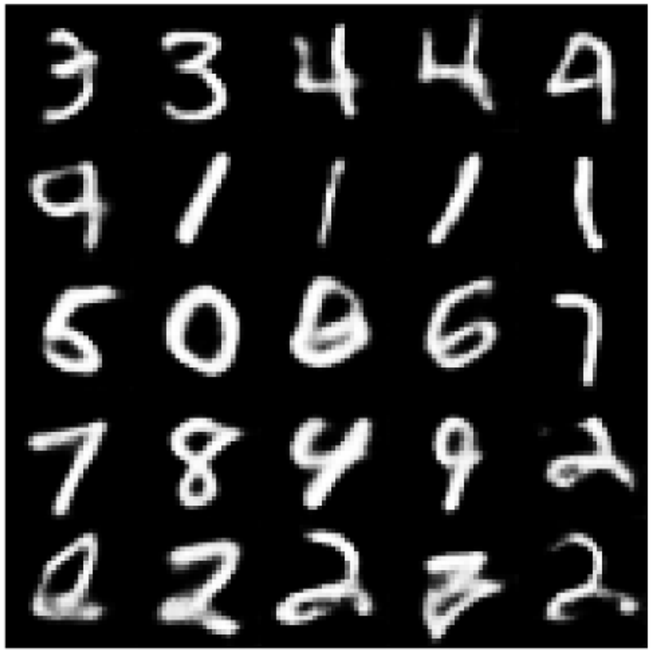


- In the t-SNE (t distributed stochastic neighbor embedding) . Roughly : dimensionality reduction technique that preserves distance in high dim

Autoencoders : Sampling issues



Using a random sampler

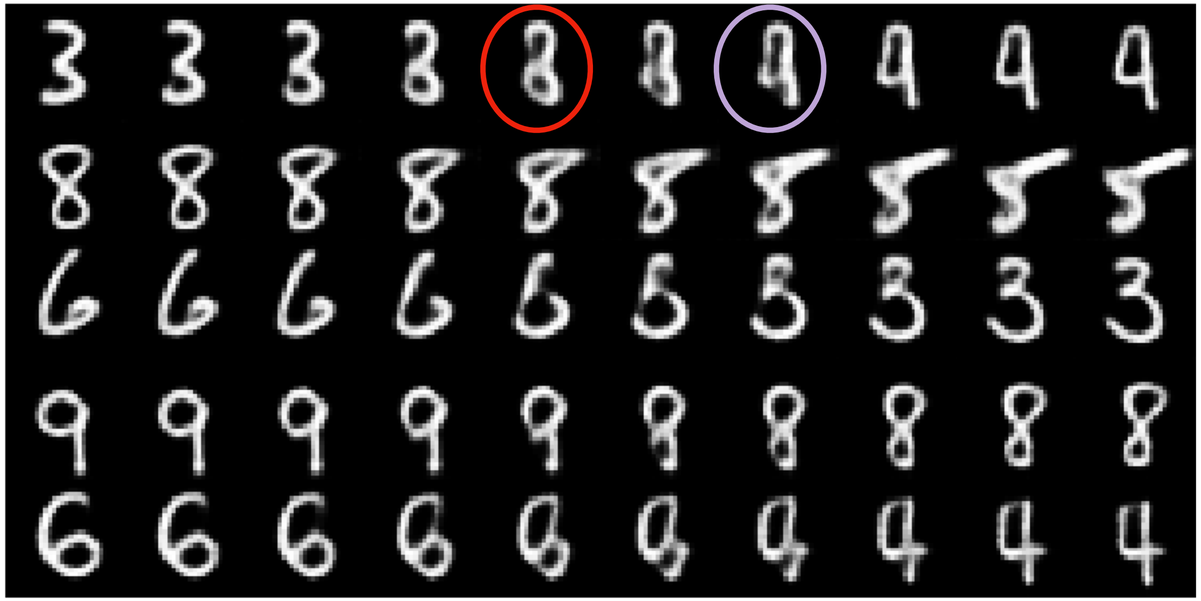


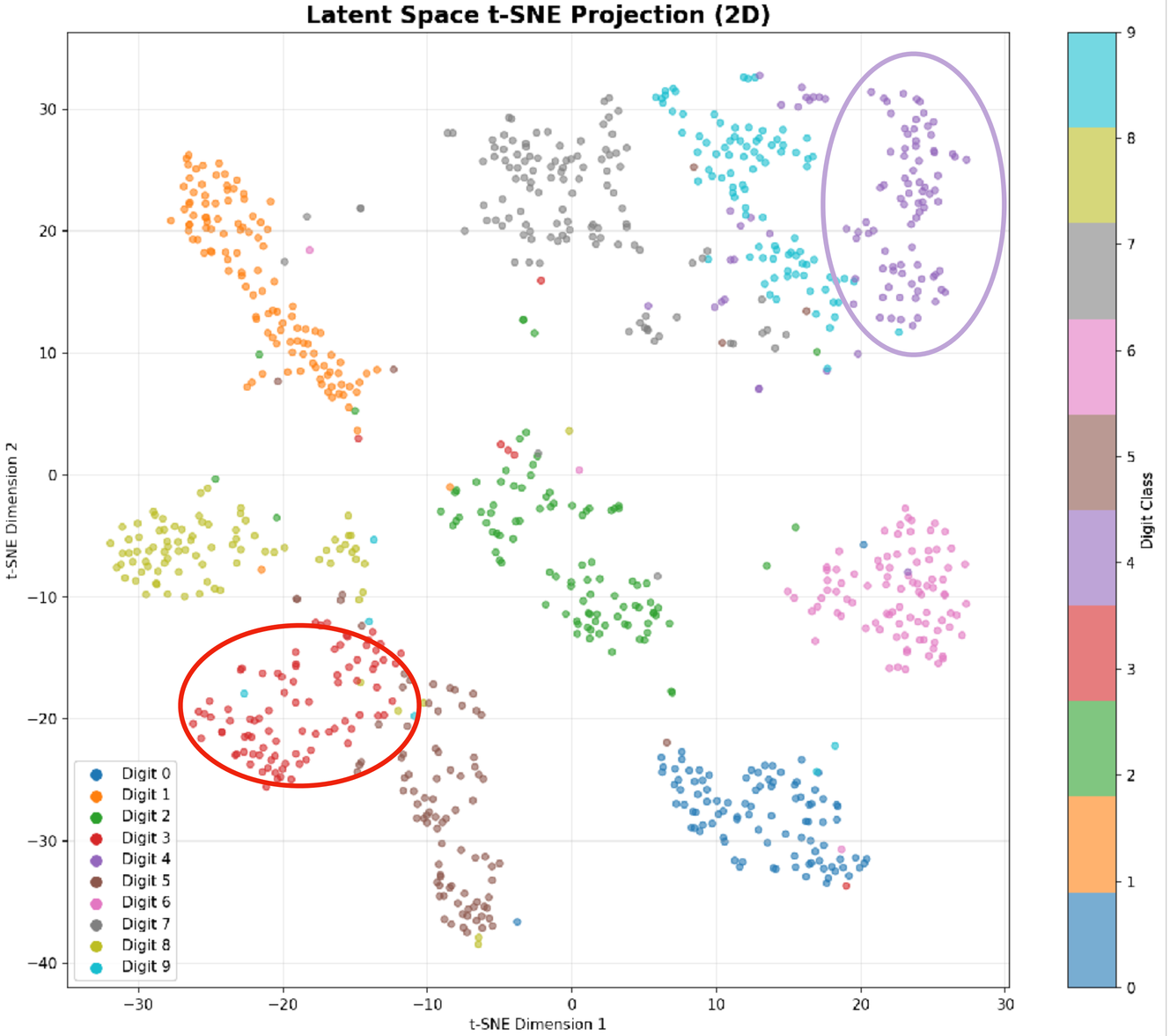
Using a GMM sampler

Random sampling is really worse, as model doesn’t know what to do with dead space in latent space

Gaussian mixture model sampler samples from regions where valid latents exist.

Autoencoders : Interpolation issues





- Interpolation is unsmooth, and gives unpredictable results

Variational Auto Encoders (VAE)

- VAEs resolve the interpolation issue by mapping data into distributions

Kingma & Welling 2014

- The relationship between input data $\mathbf{x}$ and latent variable $\mathbf{z}$ are expressed through three parametric distributions
  - Prior : $p(\mathbf{z})$, easy (definition, eg: choose $\mathbf{z} \sim \mathcal{N}(0,\mathbf{I})$, unconditioned on observation
  - (Decoding) Likelihood : $p_\theta(\mathbf{x}\,|\,\mathbf{z})$ easy , eg: multivar gaussian $p_\theta(\mathbf{x}\,|\,\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu(\mathbf{z}), \sigma^2(\mathbf{z})\mathbf{I})$
  - (Encoding) Posterior : $p_\theta(\mathbf{z}\,|\,\mathbf{x})$

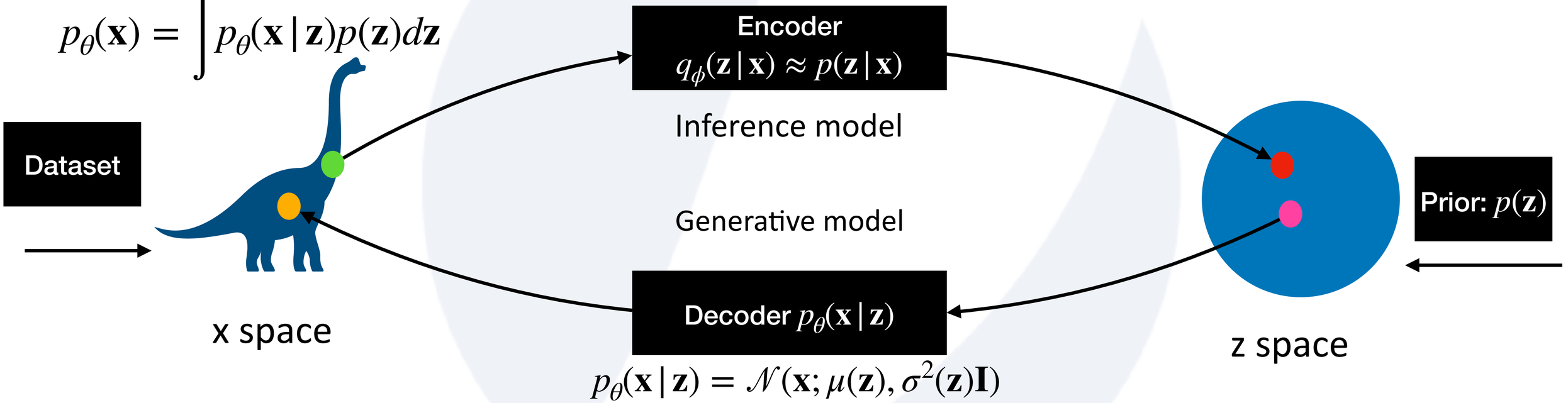
$$p_\theta(\mathbf{z}\,|\,\mathbf{x}) = \frac{p_\theta(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})}{\int p_\theta(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})d\mathbf{z}}$$ Intractable integral

- Goal: Find an approximation $q_\phi(\mathbf{z}\,|\,\mathbf{x}) \approx p_\theta(\mathbf{z}\,|\,\mathbf{x})$
- A VAE learns a stochastic map between an observed x space, with some complicated distribution to latent space with a simple distribution (eg: $z \sim \mathcal{N}(0,I)$)

Variational Auto Encoders (VAE)

- A VAE learns a stochastic map between an observed x space, with some complicated distribution to latent space with a simple distribution (eg: $z \sim \mathcal{N}(0,I)$)

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})d\mathbf{z}$$



- generative model: learns a joint distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{x}\,|\,\mathbf{z})$
  inference model approximates the intractable Posterior $p_\theta(\mathbf{z}\,|\,\mathbf{x}) \approx q_\phi(\mathbf{z}\,|\,\mathbf{x})$
- To find a loss function that finds the best approximation to the posterior.

Loss function: Evidence Lower Bound (ELBO)

$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{x}) \int q_\phi(\mathbf{z}\,|\,\mathbf{x}) d\mathbf{z} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})]$$

Evidence

Marginal Likelihood

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

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

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

ELBO KL divergence

- KL divergence is always ≥ 0, is zero when $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ is the true posterior distribution

Loss function: Evidence Lower Bound (ELBO)

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

$$\mathscr{L}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - D_{KL}\left(q_\phi(\mathbf{z}\,|\,\mathbf{x}) \,\middle|\middle|\, p_\theta(\mathbf{z}\,|\,\mathbf{x})\right)$$

$$\leq \log p_\theta(\mathbf{x}) \qquad D_{KL} \geq 0$$

- KL divergence measures:
  - divergence of approx posterior to true posterior
  - The better the approximation to true posterior, tighter the bound, smaller the gap
- Optimizing $\mathscr{L}_{\theta,\phi}(\mathbf{x})$ w.r.t θ, ϕ
  - Improves marginal likelihood (generation) $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})d\mathbf{z} = \dfrac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}\,|\,\mathbf{x})}$
  - Improves approximation for q (inference) $p_\theta(\mathbf{z}\,|\,\mathbf{x}) \approx q_\phi(\mathbf{z}\,|\,\mathbf{x})$
  - Maximizing ELBO is equivalent to minimizing KL divergence

Optimizing ELBO with stochastic gradient descent

$$\mathscr{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right]$$

- Optimizing w.r.t θ

$$\nabla_\theta \mathscr{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})\right]$$

$$\sim \frac{1}{L}\sum_{i=1}^{L} \nabla_\theta \log p_\theta(\mathbf{z},\mathbf{x})$$

MC simulation over L samples

- Optimizing w.r.t ϕ

$$\nabla_\phi \mathscr{L}_{\theta,\phi}(\mathbf{x}) \neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\phi \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right]$$

ELBO: Reparametrization trick

- Optimizing w.r.t ϕ $$\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathbf{x}) = \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right]$$

- Expectation involves sampling random $\mathbf{z} \sim q_\phi(\mathbf{z}\,|\,\mathbf{x})$, this sampling is a stochastic process and cannot back-propagate the gradient

- Change variables in the integral, s.t $\epsilon \sim p(\epsilon)$ is random, independent of ϕ, $\mathbf{x}$

$$\mathbf{z} = g(\epsilon, \phi, \mathbf{x})$$

- Choose g such that expectation and gradient commute (eg linear function)

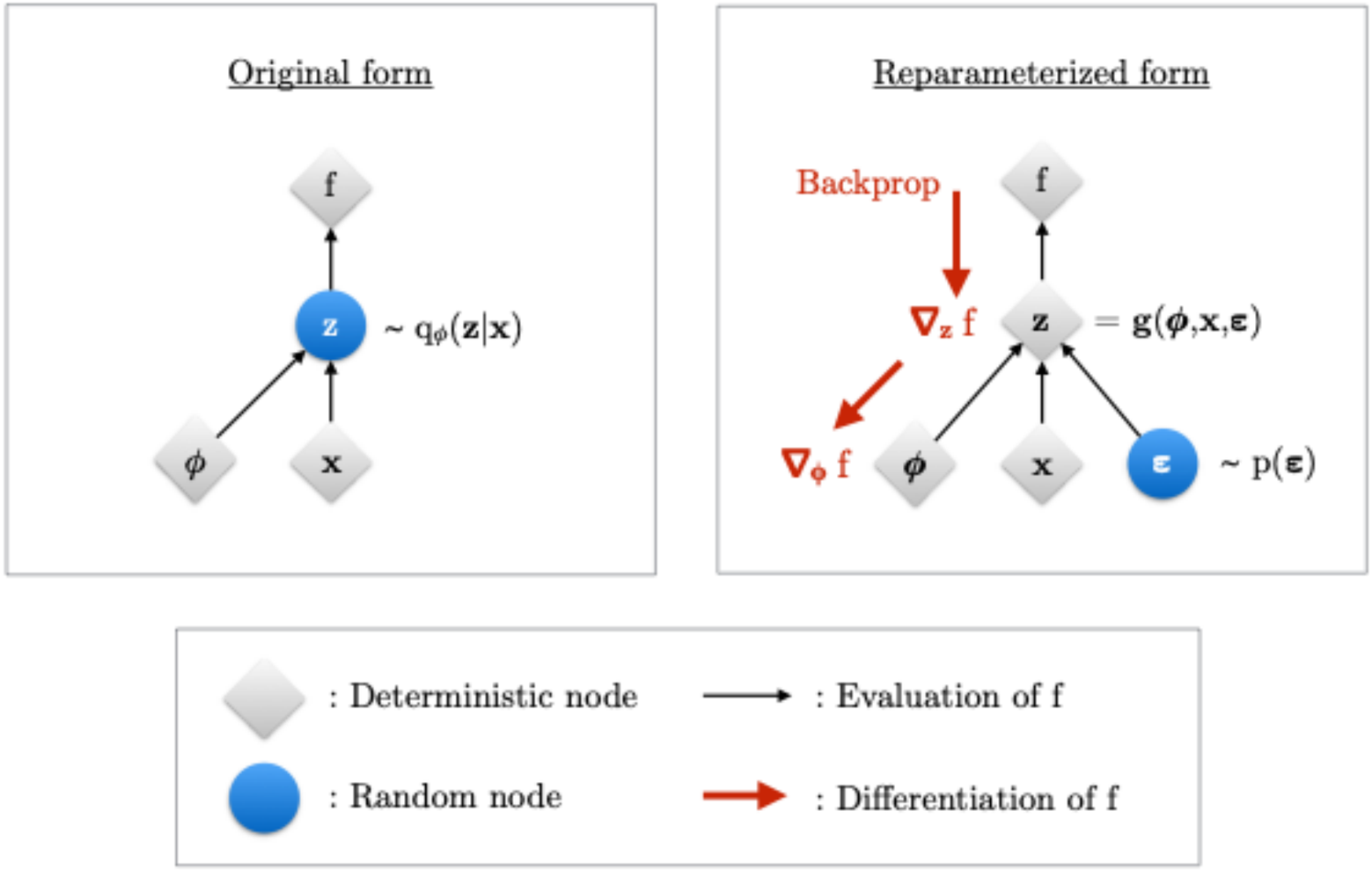
$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[f(\mathbf{z})]$$

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(\mathbf{z})]$$

$$\simeq \frac{1}{L}\sum_{i=1}^{L} \nabla_\phi f(\mathbf{z})$$

MC simulation over L samples

ELBO: Reparametrization trick



- Reparametrization $\mathbf{z} = g(\epsilon, \phi, \mathbf{x})$

$$\mathscr{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right]$$

$$\tilde{\mathscr{L}}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{p(\epsilon)}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right]$$

- So reparametrization, allows us to sample from $\epsilon \sim p(\epsilon)$, and deterministically transform it to get the sample $\mathbf{z} = g(\epsilon, \phi, \mathbf{x})$

$$\nabla_\phi \mathscr{L}_{\phi,\theta}(\mathbf{x}) \simeq \nabla_\phi \tilde{\mathscr{L}}_{\phi,\theta}(\mathbf{x})$$

$$\simeq \frac{1}{L}\sum_{i=1}^{L} \nabla_\phi \tilde{\mathscr{L}}_{\phi,\theta}(\mathbf{x})$$

ELBO: Reparametrization trick

- Common choice for encoder function (Gaussian Encoder)

$$q_\phi(\mathbf{z}\,|\,\mathbf{x}) = \prod_i q_\phi(z_i\,|\,\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)$$

$$\mathbf{z} = g(\epsilon, \phi, \mathbf{x}) = \mu(\mathbf{x}, \phi) + \sigma(\mathbf{x}, \phi) \circ \epsilon$$

$$\epsilon \sim \mathcal{N}(0, \mathbf{I})$$

- The PDF upon change of variables

$$q_\phi(\mathbf{z}\,|\,\mathbf{x}) = \frac{p(\epsilon)}{det\left(\frac{\delta g}{\delta \epsilon}\right)} = \frac{p(\epsilon)}{\prod_i \sigma_i}$$

- Posterior density

$$\log q_\phi(\mathbf{z}\,|\,\mathbf{x}) = \sum_i \left(\mathcal{N}(\epsilon_i; 0, 1) - \log \sigma_i\right)$$

ELBO: Reparametrization trick

- Go back to the relation between ELBO and marginal likelihood

$$\mathscr{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\,|\,\mathbf{x})}\right] = \log p_\theta(\mathbf{x}) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}\,|\,\mathbf{x})}{p_\theta(\mathbf{z}\,|\,\mathbf{x})}\right]$$

- In gaussian parametrization

$$\tilde{\mathscr{L}}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{\mathcal{N}(0,\mathbf{I})}\left[\log p_\theta(\mathbf{x}\,|\,\mu(\mathbf{x},\phi) + \sigma(\mathbf{x},\phi)\circ\epsilon)\right] - \frac{1}{2}\sum_i \left(1 + \log\sigma_i^2 - \mu_i^2 - \sigma_i^2\right)$$

- Construct an optimizer for θ, ϕ and produce better estimates of μ and σ that minimize the loss function

ELBO visualization of training objective

- Construct an optimizer for θ, ϕ and produce better estimates of mu and sigma that minimize the loss function

- The encoders posterior approximation is “pulled” towards the standard normal prior

Reparametrization trick and the forward process

- Reparametrization trick $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

$$\mathbf{z} = g(\epsilon, \phi, \mathbf{x}) = \mu(\mathbf{x}, \phi) + \sigma(\mathbf{x}, \phi) \circ \epsilon$$

- Diffusion process: A markov chain, with gaussian noise at each step

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t$$

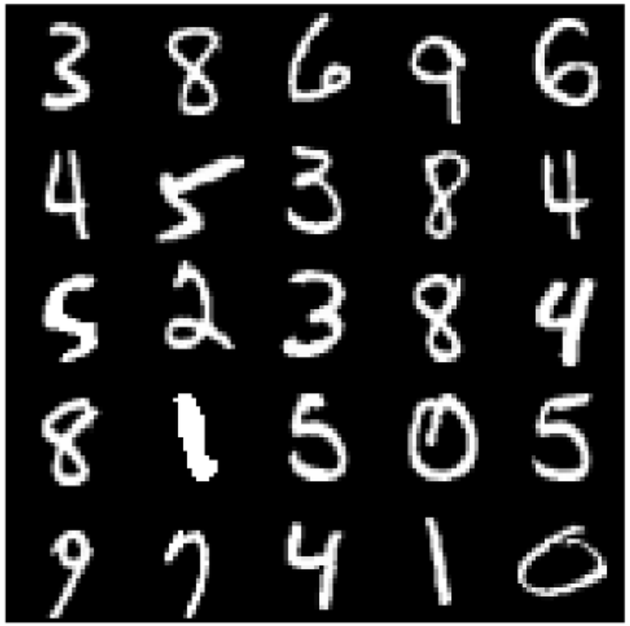
  - previous state acts like mean, and noise acts like variance
  - Each step in forward process is analogous to reparametrization trick

- Possible to have a closed form sampling: (repeatedly substituting the equation)

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon$$

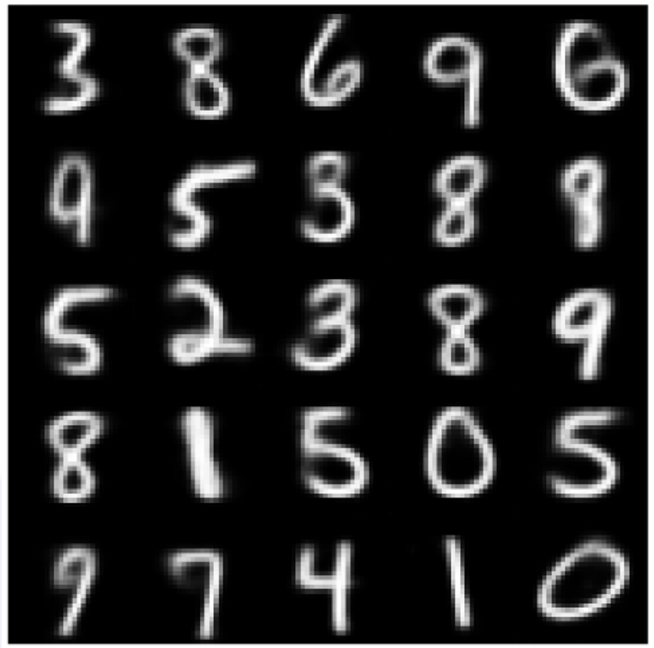
- Known: Estimate Noisy image t any timestep t, in a single step, makes training faster
- Reason: adding two gaussian random var is a random gaussian var with sum of means/variances

VAE MNIST example



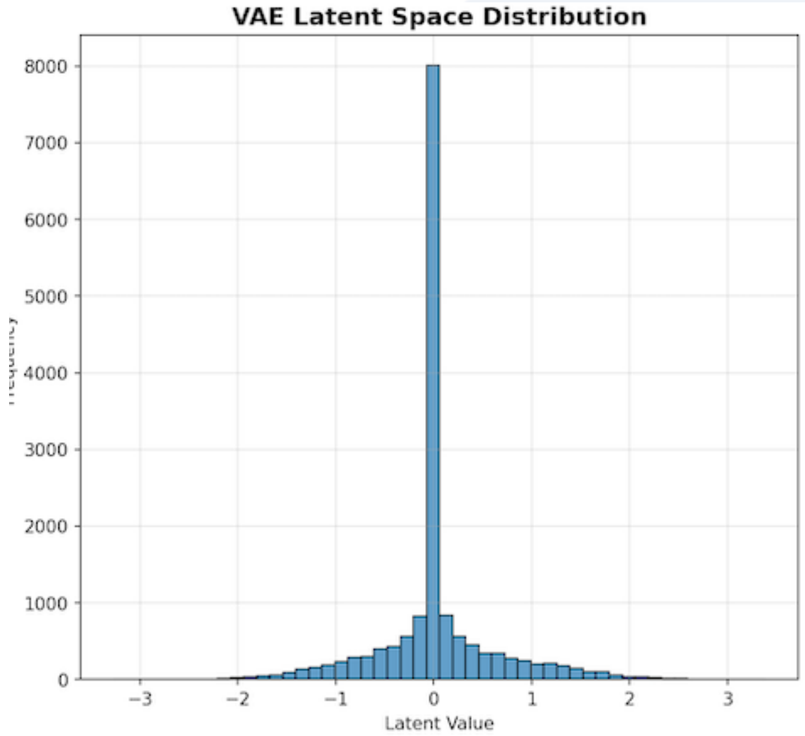
Original image set

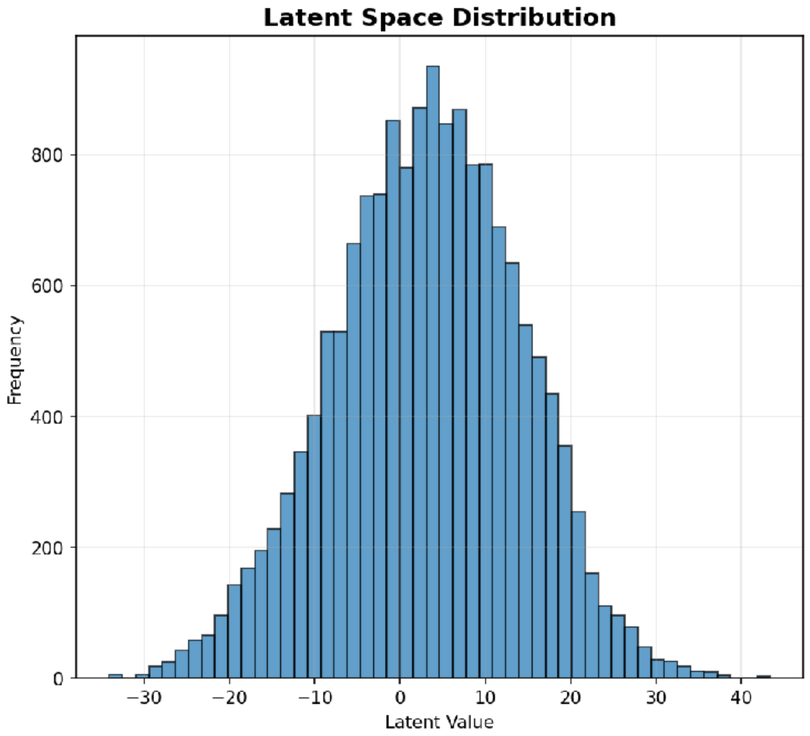
```
📊 MODEL ARCHITECTURE:
--------------------------------------------------
Encoder parameters:          427,680
Decoder parameters:          339,009
Total trainable parameters: 766,689
Estimated model size:         2.92 MB

🔢 LATENT SPACE:
--------------------------------------------------
Input dimensions:            (1, 28, 28) = 784 values
Latent dimensions:           16

📉 COMPRESSION:
--------------------------------------------------
Compression ratio:           49.00x
Compression percentage:      97.96%
Information retention:       2.04%

🔍 ANALYZING LATENT SPACE...
--------------------------------------------------
Encoded 1000 images
Latent codes shape:          (1000, 16)
Latent mean:                 0.0299
Latent std:                  0.6002
Latent min:                  -3.2739
Latent max:                  3.3905
```



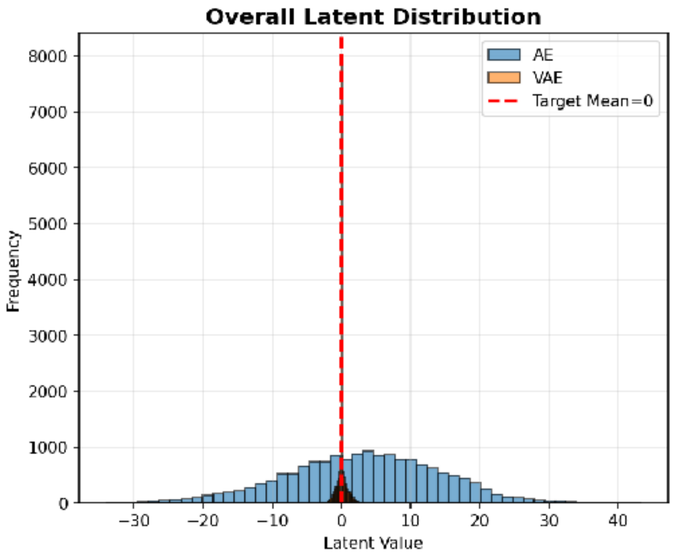
Reconstructed image

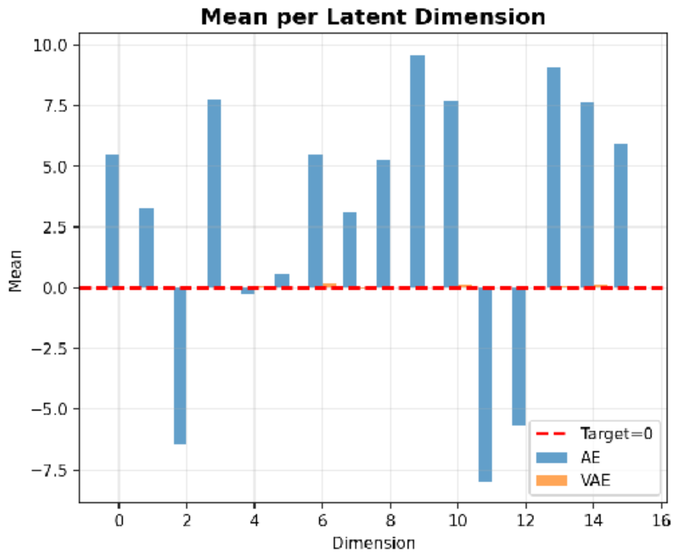
VAE latent space organization compared to AE

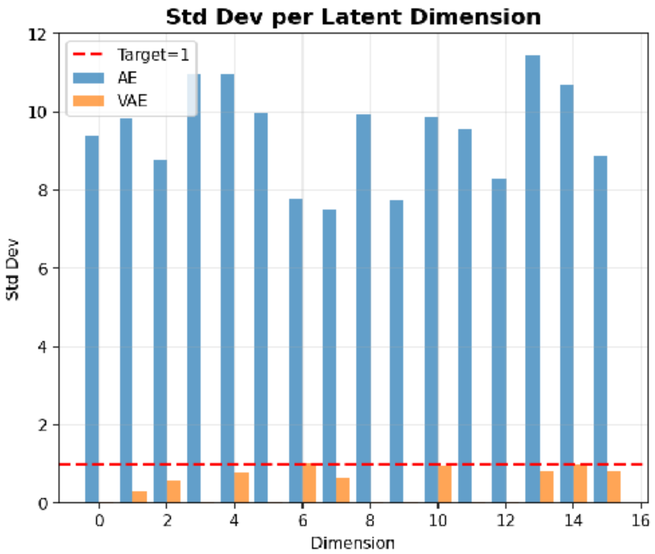


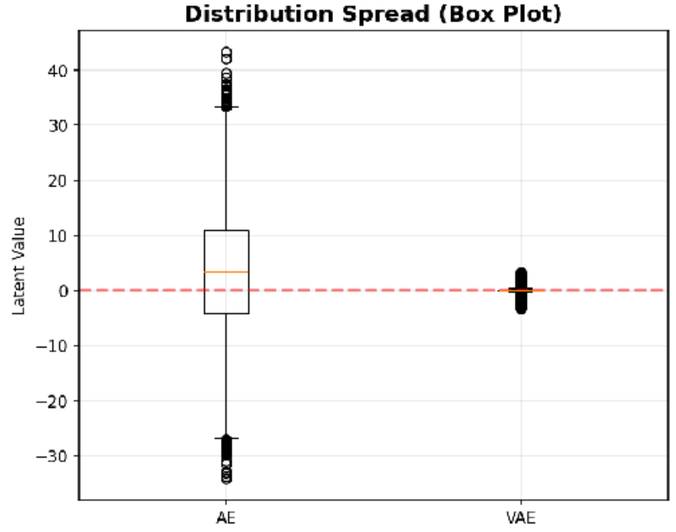


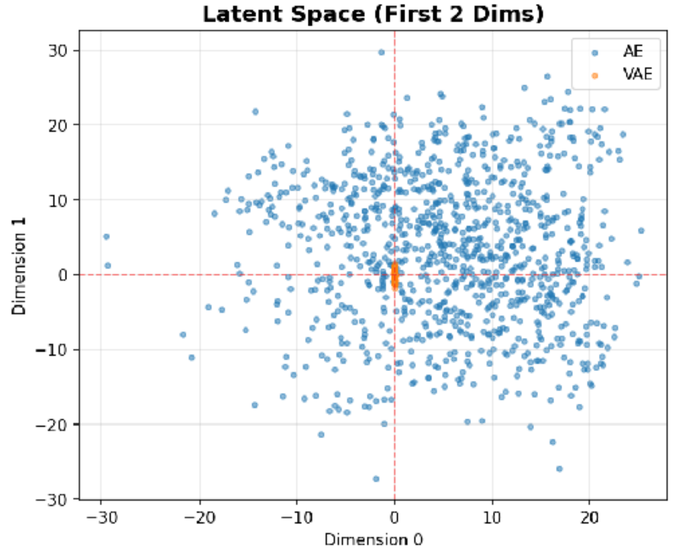
VAE latent space organization compared to AE

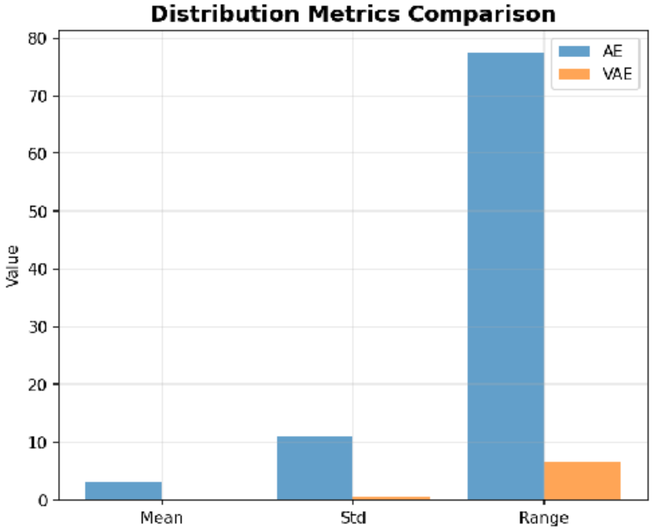




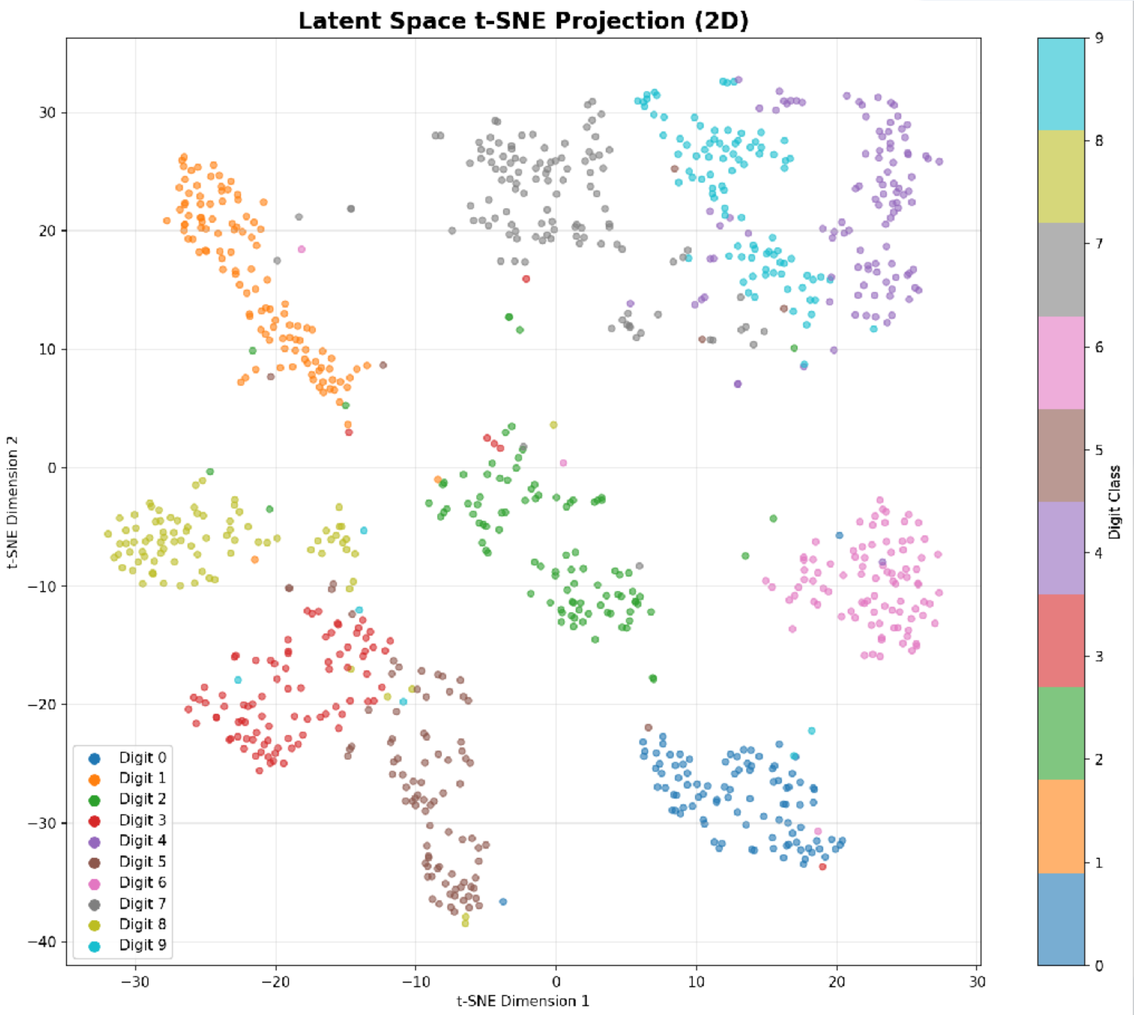


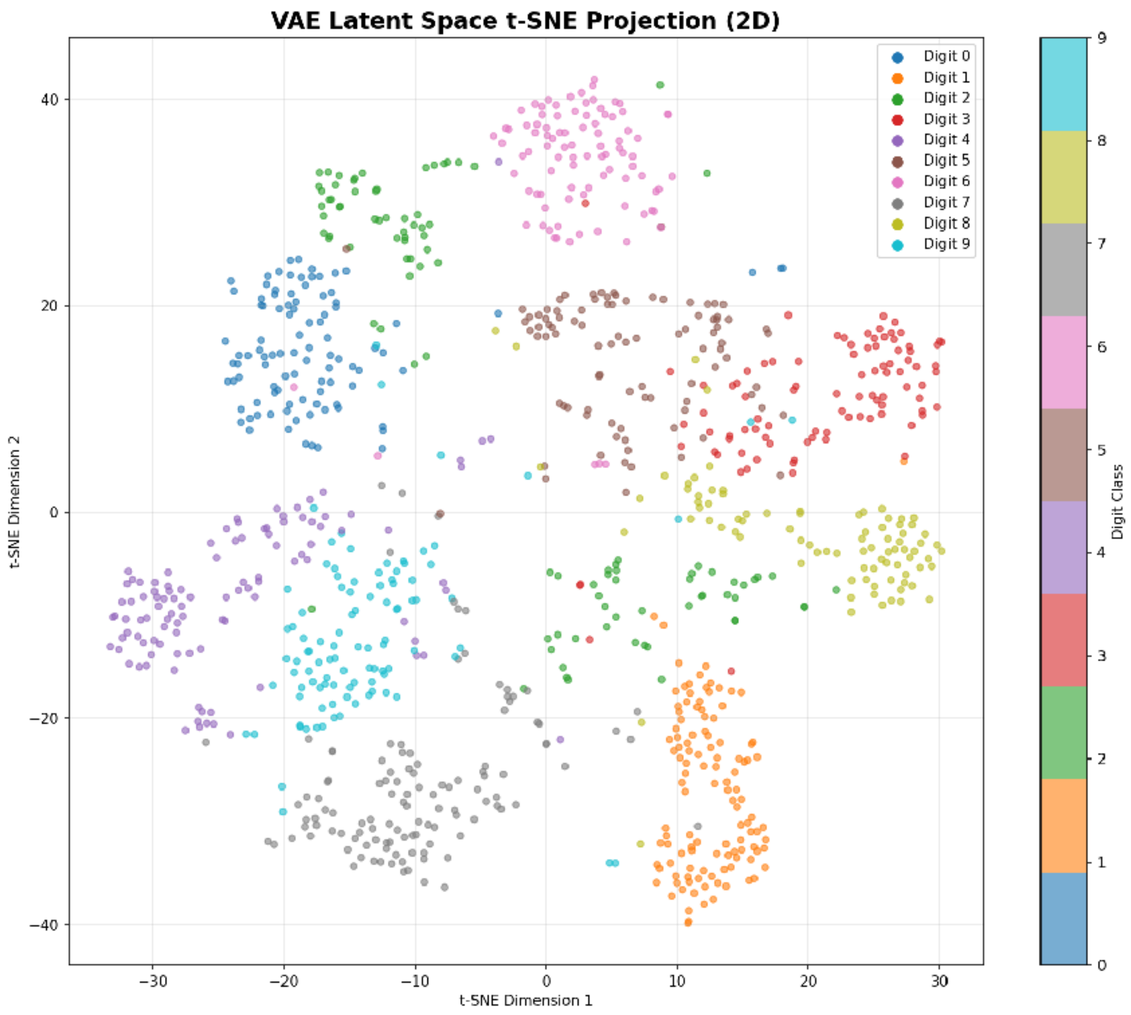






VAE latent space organization compared to AE





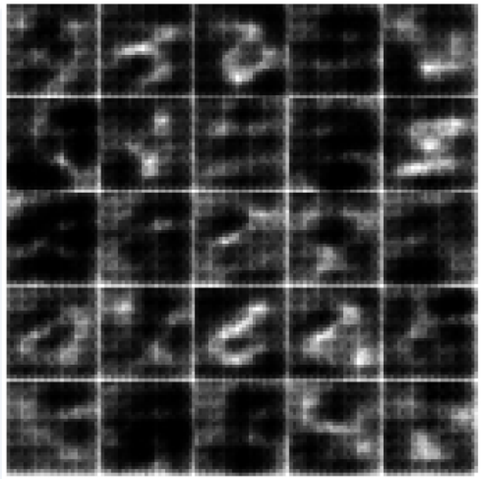
- T SNE projections are also much closer in VAE

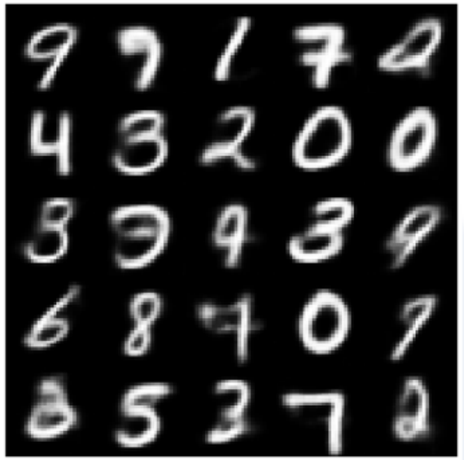
VAE vs AE sampling

AE

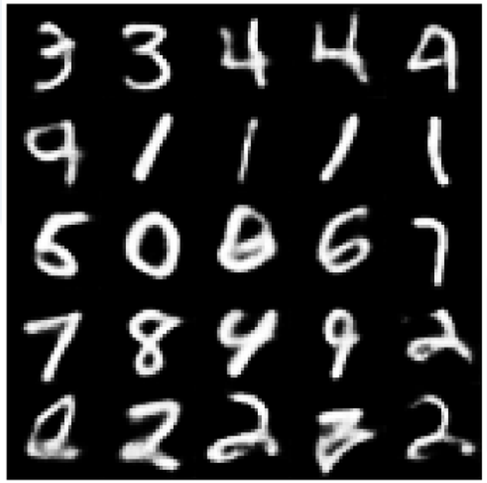
VAE

Using a random sampler

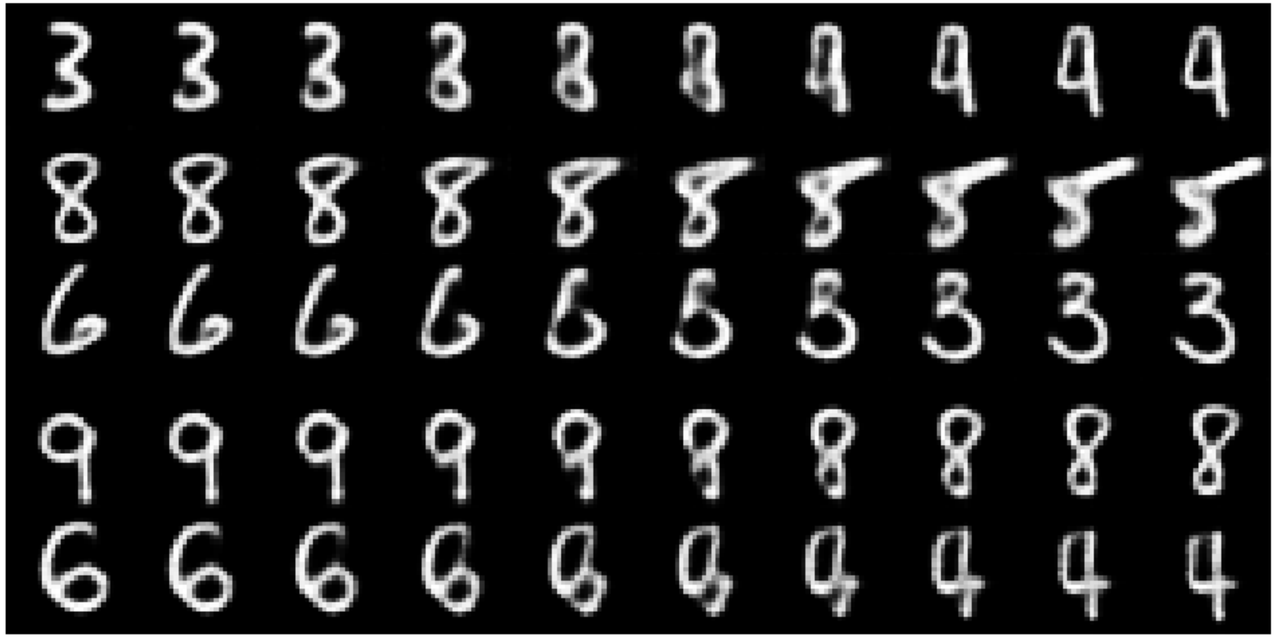


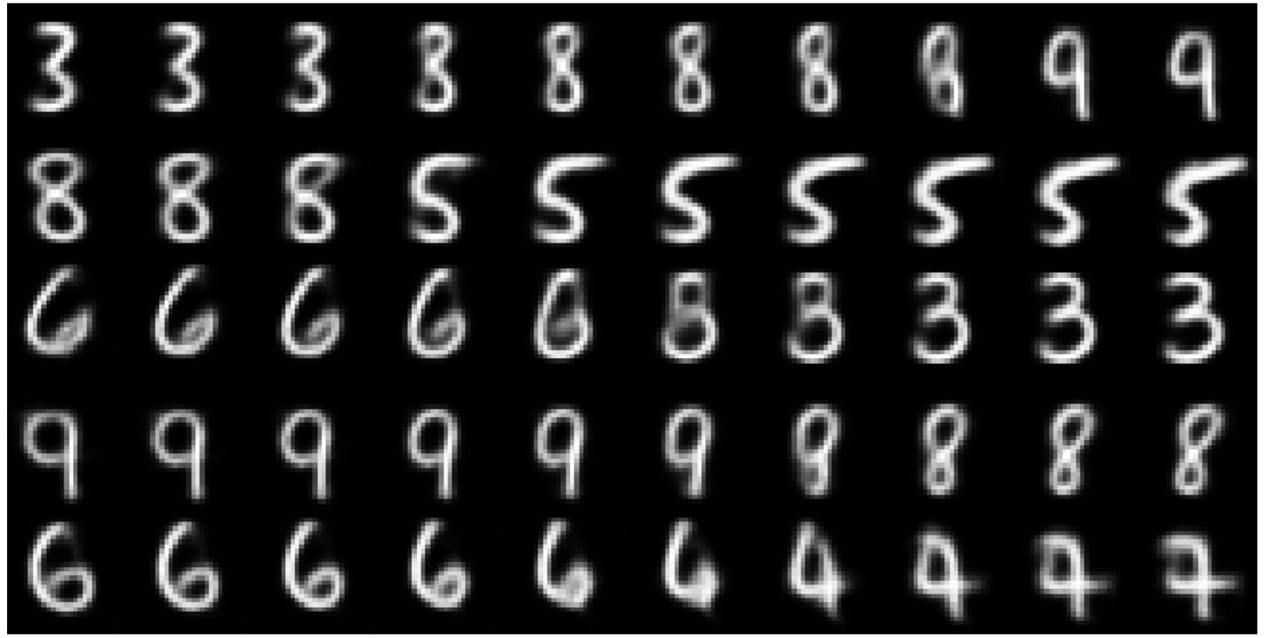


Using a GMM sampler



VAE vs AE interpolation





Can train VAE better but interpolation is smoother with lesser artifacts

Keep training…