XCS236 | Assignment 2 | Group OH | Jonathan Hague (CF) | jhague@stanford.edu | 02.28.2025

Below, I compare Standard VAE and its variations in terms of the objective function, prior distribution, and inference method as they relate to Assignment 2.

Remember, our objective in a VAE is to learn the underlying data distribution over data points x and generate new sample close to the true data. Instead of directly maximizing p(x), VAEs optimize their respective loss function. Means $\boldsymbol{\mu_i}$ and variances $\boldsymbol{\sigma_i^2}$ are learned during training.

	Standard VAE	**GMVAE**	**IWAE**
The Idea	**Fixed prior** acts as a regularizer of latent space. Uses **a single latent sample** per datapoint to approximate the marginal likelihood. This is simpler, but weaker bound on likelihood. Simple and efficient training.	Uses a **Mixture of Gaussians prior** instead of a single Gaussian. This allows the latent space to learn clusters and discover subgroups. **The log probability density function** for each element of in the last dimension of a **multivariate normal distribution** assuming independent dimensions: log p(x)=−1/2∑(log vi + (xi−mi)^2/vi + log (2π))	Uses **multiple latent samples** for a **tighter bound** on marginal likelihood. This leads to improved density estimation.
Prior p(z)	Fixed standard gaussian as the prior $\boldsymbol{\mathcal{N}(0, I)}$	Learnable **Mixture of Gaussians** $\sum_{i} \boldsymbol{\mathcal{N}(\mu_i, \sigma_i^2)}$ Set in **gmvae.py** under **self.z_pre** The first k rows correspond to the means $\boldsymbol{\mu_i}$. The next k rows correspond to the variances $\boldsymbol{\sigma_i^2}$. Initializations are scaled to ensure that the values do not lead to unstable training. **Parametrized learnable prior** as: **z_pre = 1/(sqrt (k·zdim) * $\mathcal{N}(0, I)$**	Fixed standard gaussian as the prior $\boldsymbol{\mathcal{N}(0, I)}$
KL Term	Computed **analytically**.	Computed against a mixture of Gaussians (non-trivial, requiring approximation via **Monte Carlo sampling**).	Approximated via **Monte Carlo sampling**.
Inference Model (encoder output) $\boldsymbol{q_\phi(z|x)}$	**Single** latent sample per data point. The process of using a neural network to approximate the posterior is called **amortized inference.** qϕ(z|x) represents an approximation of the true posterior p(z|x). **Variational approximate posterior** qϕ(z|x) is a multivariate Gaussian distribution: qϕ(z|x)=N(z | μϕ(x),diag(σϕ^2(x))) Since the covariance is **diagonal**, the latent variables are assumed to be independent.	**Single** latent sample per data point.	**Multiple** latent samples per data point. By increasing m (the number of samples), IWAE provides a better approximation of the true log-likelihood compared to the standard VAE.

	Standard VAE	GMVAE	IWAE
	We often do not know the exact relationships between latent variables. The model can still learn meaningful structure despite this assumption.		
Loss Function	Optimizes the **Evidence Lower Bound (ELBO)**: $L(x;\theta,\phi) = \mathbb{E}_{q_\phi(z\|x)}[\log p_\theta(x\|z)] - D_{KL}(q_\phi(z\|x)\|\|p(z))$ Uses one sample z per datapoint to approximate the expectation. **The KL divergence term** regularizes the latent space **to follow p(z) = $\mathcal{N}(\mathbf{0},\mathbf{I})$** VAE learning objective is to **minimize the negative ELBO**: **−ELBO = Reconstruction Loss + KL Divergence** Reconstruction Loss: measures how well the decoded output matches the input. Encourages the model to generate realistic data by maximizing the likelihood of x given z. KL Divergence: Ensures the latent space follows a normal distribution. Regularizes the latent space, ensuring qϕ(z|x) is close to the prior p(z).	Modifies **the prior distribution** in the ELBO objective: $L(x;\theta,\phi) = \mathbb{E}_{q_\phi(z\|x)}[\log p_\theta(x\|z)] - D_{KL}(q_\phi(z\|x)\|\|p_\theta(z))$ where the prior $p_\theta(z)$ is a **Mixture of Gaussians** instead of a standard normal: $p_\theta(z) = \sum_{i=1}^{k} \frac{1}{k}\mathcal{N}(z\|\mu_i,\sigma_i^2)$	Refines ELBO by using multiple latent samples $\{z^{(i)}\}_{i=1}^{m}$: $L_m(x;\theta,\phi) = \mathbb{E}_{z^{(1)},\ldots,z^{(m)}\sim q_\phi(z\|x)}$ $[\log(\frac{1}{m}\sum_{i=1}^{m}\frac{p_\theta(x,z^{(i)})}{q_\phi(z^{(i)}\|x)})]$ If m = 1 IWAE reduces to a standard VAE. Since IWAE **draws multiple samples per datapoint** (iw = number of importance-weighted samples), we **expand** the mean, variance, and input data x **to match the number of samples**. Instead of using a single sample, we estimate the expectation **with multiple samples in niwae.** We use the **log-mean-exp trick** for numerical stability when averaging importance-weighted samples.

HINTS for problems 1, 2, and 3:

sample gaussian (m, v): Reparameterization trick

$\mathbf{z=\mu+\sigma\cdot\epsilon}$, where $\boldsymbol{\epsilon\sim N(0,I)}$ → Transform the standard normal sample using the given mean m and variance v.
This allows gradients to flow through μ **(mean) and** σ **(standard deviation)** during backpropagation.

Why?

The VAE's latent space follows a normal distribution.
Instead of sampling directly (which disrupts backpropagation), we rewrite *z* as a deterministic function of *m* and *v*.

β-VAE:

A higher β **compresses the information stored in** z.
The model is encouraged to **discard unimportant information**, which can help **disentangle latent factors**.
However, **if** β **is too large, reconstructions suffer** because the latent space does not capture enough information.

Mixture of Gaussian:

We need to expand z before computing **the log probability** of z under each Gaussian component. This allows broadcasting so that z can be compared against each mixture component.

IWAE Lower Bounds the Log-Likelihood and ELBO:

$$\text{Log } p\theta(x) \geq Lm(x) \geq L1(x)$$

where:
Log pθ(x) is the **true marginal log-likelihood**.
*L*m(x) is the **IWAE bound with m samples**.
L1(x) is the **standard ELBO (when m=1)**.

Jensen's Inequality states that for a convex function f(x), we have:

$$f(E[X]) \leq E[f(X)]$$

Since log is a **concave function**, we can **move it inside the expectation**.
This step shows that **IWAE (*L*m(x)) is always a lower bound on the true log-likelihood**.

IWAE:

ut.duplicate() creates iw copies of the input tensor along a new dimension, effectively **increasing the batch size**.

Loss function:

We take the **mean over the batch** to ensure the loss is scalable across different batch sizes.