1. Reparameterization Trick

Formally, a latent variable model \( p \) is a probability distribution over observed variables \( x \) and latent variables \( z \) (variables that are not directly observed but inferred), \( p_\theta(x, z) \). Because we know \( z \) is unobserved, using learning methods learned in class (like supervised learning methods) is unsuitable. Indeed, our learning problem of maximizing the log-likelihood of the data turns from:

\[
\theta \leftarrow \arg \max_\theta \frac{1}{N} \sum_{i=1}^{N} \log[p_\theta(x_i)]
\]

to:

\[
\theta \leftarrow \arg \max_\theta \frac{1}{N} \sum_{i=1}^{N} \log\left[ \int p_\theta(x_i \mid z)p(z)dz \right]
\]

where \( p(x) \) has become \( \int p_\theta(x_i \mid z)p(z)dz \).

(a) Instead of directly optimizing the likelihood of \( p(x) \), we define the proxy likelihood as:

\[
\mathcal{L}(x_i, \theta, \phi) = E_{z \sim q_\phi(z \mid x_i)} \left[ \log[p_\theta(x_i \mid z)] \right] - D_{KL}\left[ q_\phi(z \mid x_i) \| p(z) \right]
\]

This proxy term is a \textit{lower bound} of the original likelihood. In order to optimize this variational lower bound, \textbf{which distribution do we sample from}?

(b) How do we take gradients through samples? To do we, we need to show how sampling can be done as a deterministic and continuous function of the model parameters \( \theta \) and the independent source of randomness (ie. the \textit{prior}). Such an explicit representation of sampling is called \textbf{reparameterization}. Consider the case where the data \( x \) is sampled from a normal distribution with its mean parameterized by parameters \( \theta \) and variance of 1, with our objective being a quadratic function of \( x \):

\[
\min_\theta E_q[x^2]
\]

Write \( x \) as a function of \( \epsilon \), a vector sampled from a standard Normal \( \mathcal{N}(0,1) \), and compute the gradient of the expectation term above:
2. Latent Variable Models

(a) **Describe what the encoder and decoder of the VAE are respectively doing** to capture and encode this information into a latent representation of space $z$. **Is the latent space dimension smaller than the input space?** How is the information bottleneck created in VAE as opposed to Autoencoder?

(b) Once the VAE is trained, **how do we use it to generate a new fresh sample from the learned approximation of the data-generating distribution**?

(c) In the previous question we have used a proxy likelihood:

$$
    \mathcal{L}(x_i, \theta, \phi) = E_{z \sim q_\phi(z|x_i)} \left[ \log p_\theta(x_i | z) \right] - D_{KL} \left[ q_\phi(z | x_i) || p(z) \right]
$$

**Please show that** $\mathcal{L}(x_i, \theta, \phi)$ **is always a lower bound to the true log likelihood for** $x_i$.

3. Diffusion Models

In the previous question we considered sampling from a discrete distribution. Let’s now see how iteratively adding Gaussian noise to a data point leads to a noisy sequence, and how the reverse process refines noise to generate realistic samples.

The classes of generative models we’ve considered so far (VAEs, GANs), typically introduce some sort of bottleneck (**latent representation** $z$) that captures the essence of the high-dimensional sample space ($x$). An
alternate view of representing probability distributions \( p(x) \) is by reasoning about the score function i.e. the gradient of the log probability density function \( \nabla x \log p(x) \).

Given a data point sampled from a real data distribution \( x_0 \sim q(x) \), let us define a forward diffusion process iteratively adding small amount of Gaussian noise to the sample in \( T \) steps, producing a sequence of noisy samples \( x_1, \ldots, x_T \).

\[
q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \quad (1)
\]

The data sample \( x_0 \) gradually loses its distinguishable features as the step \( t \) becomes larger. Eventually when \( T \to \infty \), \( x_T \) is equivalent to an isotropic Gaussian distribution. (You can assume \( x_0 \) is Gaussian).

To generative model is therefore the reverse diffusion process, where we sample noise from an isotropic Gaussian, and iteratively refine it towards a realistic sample by reasoning about \( q(x_{t-1}|x_t) \).

(a) **Anytime Sampling from Intermediate Distributions**

Given \( x_0 \) and the stochastic process in eq. (1), show that there exists a closed form distribution for sampling directly at the \( t^{th} \) time-step of the form

\[
q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_0, (1 - \alpha_t) I)
\]

(b) **Reversing the Diffusion Process**

Reversing the diffusion process from real to noise would allow us to sample from the real data distribution. In particular, we would want to draw samples from \( q(x_{t-1}|x_t) \). **Show that given \( x_0 \), the reverse conditional probability distribution is tractable and given by**

\[
q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu(x_t, x_0), \hat{\beta}_t I)
\]

- **Hint:** Use Bayes Rule on eq. (1), assuming that \( x_0 \) is drawn from Gaussian \( q(x) \)
- **Hint:** When applying Bayes rule to compute \( q(x_{t-1}|x_t, x_0) \), don’t expand the entire Gaussian pdf. Instead just compute the exponent parts to simplify your work.
- **Hint:** Scalar form of Gaussian pdf is given as \( f(z) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left\{ -\frac{1}{2} \left( \frac{z - \mu}{\sigma} \right)^2 \right\} \)