Name: **Zahra Soukhtedel** Assignment Number: **4**
Student Id: **98105138** Course: **Deep Learning**

Problem 1

1.1

The assumption made in this case is that the decoder output follows a multivariate Gaussian distribution characterized by mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
Given an input data point $x^{(i)}$, the objective is to compute the negative log-likelihood of the data point given the reconstructed output assuming it follows a multivariate Gaussian distribution. This can be written using Bayes' rule as(D represents the dimensionality of the input data):

$$\log p_\theta(x^{(i)}|z') = -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(x^{(i)} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(x^{(i)} - \boldsymbol{\mu}) - \frac{D}{2}\log 2\pi,$$

To obtain the reconstruction term in the ELBO loss function under this assumption, we take the expectation over the posterior distribution $q_\phi(z|x^{(i)})$ of the latent variable z' and average over the entire dataset. This results in the expression:

$$E_{z'\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z')] = -\frac{1}{2}\sum_{j=1}^{D}\left(\log\sigma_j^2 + \frac{(x_j^{(i)} - \mu_j)^2}{\sigma_j^2} + \log 2\pi\right),$$

where μ_j and σ_j^2 represent the mean and variance of the j-th component of the reconstructed output, respectively.

1.2

Definition of the reconstruction term in the ELBO loss:

$$E_{z'\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z')]$$

This represents the expected log-likelihood of the input $x^{(i)}$ under the decoder distribution p_θ given a latent variable z' sampled from the encoder distribution $q_\phi(z|x^{(i)})$.
Since we are now assuming that $x^{(i)}$ is binary and follows a multivariate Bernoulli distribution, we can express the likelihood term as(m is the dimensionality of the output):

$$p_\theta(x^{(i)}|z') = \prod_{j=1}^{m} p_\theta(x_j^{(i)}|z')^{x_j^{(i)}}(1 - p_\theta(x_j^{(i)}|z'))^{(1-x_j^{(i)})}$$

Where $p_\theta(x_j^{(i)}|z')$ is the probability that the j-th component of the reconstructed input is 1 given the latent variable z'. Taking the logarithm of this expression and simplifying yields:

$$\log p_\theta(x^{(i)}|z') = \sum_{j=1}^{m} x_j^{(i)}\log p_\theta(x_j^{(i)}|z') + (1 - x_j^{(i)})\log(1 - p_\theta(x_j^{(i)}|z'))$$

Substituting this expression into the reconstruction term, we obtain:

$$E_{z' \sim q_\phi(z|x^{(i)})}[\log_{p_\theta}(x^{(i)}|z')] = \sum_{j=1}^{m} E_{z' \sim q_\phi(z|x^{(i)})}[x_j^{(i)} \log p_\theta(x_j^{(i)}|z') + (1 - x_j^{(i)}) \log(1 - p_\theta(x_j^{(i)}|z'))]$$

Since $x_j^{(i)}$ is binary, either 0 or 1, we can simplify further by considering only one of the terms in the sum, depending on whether $x_j^{(i)} = 0$ or $x_j^{(i)} = 1$. For example, if $x_j^{(i)} = 1$, then the corresponding term in the sum reduces to:

$$\begin{aligned} E_{z' \sim q_\phi(z|x^{(i)})}[x_j^{(i)} \log p_\theta(x_j^{(i)} = 1|z')] &= E_{z' \sim q_\phi(z|x^{(i)})}[\log p_\theta(x_j^{(i)} = 1|z')] \\ &= \sum_{z'} q_\phi(z'|x^{(i)}) \log p_\theta(x_j^{(i)} = 1|z') \end{aligned}$$

Similarly, if $x_j^{(i)} = 0$, then the corresponding term reduces to:

$$\begin{aligned} E_{z' \sim q_\phi(z|x^{(i)})}[(1 - x_j^{(i)}) \log p_\theta(x_j^{(i)} = 0|z')] &= E_{z' \sim q_\phi(z|x^{(i)})}[\log(1 - p_\theta(x_j^{(i)} = 1|z'))] \\ &= \sum_{z'} q_\phi(z'|x^{(i)}) \log(1 - p_\theta(x_j^{(i)} = 1|z')) \end{aligned}$$

Putting these together, we get the final formula:

$$E_{z' \sim q_\phi(z|x^{(i)})}[\log_{p_\theta}(x^{(i)}|z')] = \sum_{j=1}^{m} x_j^{(i)} \log p_\theta(x_j^{(i)} = 1|z') + (1 - x_j^{(i)}) \log p_\theta(x_j^{(i)} = 0|z')$$

where $p_\theta(x_j^{(i)} = 1|z')$ is the probability that the j-th component of the reconstructed input is 1 given the latent variable z'. This formula allows us to compute the expected reconstruction loss for a binary VAE with multivariate Bernoulli outputs, given a set of observed inputs and corresponding latent variables.

1.3

The choice of distribution can have an impact on both the target function and the optimization process. If the decoder output is assumed to follow a multivariate Gaussian distribution, then the target function used in the VAE will be the negative log-likelihood of the data under the Gaussian model. This loss function encourages the VAE to learn a latent representation that can effectively decode into points that are more likely to belong to the original dataset. In the optimization process, this loss function is minimized using techniques such as stochastic gradient descent.

On the other hand, if the decoder output is assumed to follow a multivariate Bernoulli distribution, then the target function used in the VAE will be the binary cross-entropy loss between the generated output and the target output. This loss function encourages the VAE to learn a latent representation that can effectively generate binary pixel values for image datasets. In the optimization process, this loss function is also minimized using techniques such as stochastic gradient descent.

The choice of distribution for the decoder output is often dependent on the type of data being modeled. For continuous data, such as images with pixel values ranging from 0 to 255, a multivariate Gaussian distribution is typically used. For binary data, such as black and white images or text data, a multivariate Bernoulli distribution is often used.

1.4

VAEs can experience overfitting when the model's complexity causes it to memorize the training data instead of learning how to generalize to new data. As a consequence, the model may produce data that is nearly identical or completely indistinguishable from the training data but unable to generate new data that is different from the training data.
The KL Divergence term found in ELBO is used to regularize the latent space by encouraging it to follow a standard normal distribution. This helps prevent overfitting by making the model generate new data rather than just memorizing the training data.
However, if the weight of the KL Divergence term in the ELBO loss function is too high, it can cause the model to focus too much on regularization and not enough on fitting the data, leading to underfitting. On the other hand, if the weight is too low, the model may start to overfit the training data. Therefore, finding the right balance between regularization and fitting the data is crucial for preventing overfitting in VAEs.

2

2.1

For P, we have that for any $(x, y) \sim P$, $x = 0$ and $y \sim Uniform(0, 1)$. Thus, the joint pdf of P is:

$$p(x, y) = \begin{cases} 1 & \text{if } x = 0 \text{ and } 0 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

For Q, we have that for any $(x, y) \sim Q$, $y \sim Uniform(0, 1)$ and $x = \theta$ with $0 \leq \theta \leq 1$. Hence, the joint pdf of Q is:

$$q(x, y) = \begin{cases} 1 & \text{if } x = \theta (0 \leq \theta \leq 1) \text{ and } 0 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Now, let's compute the pdf of $M = \frac{1}{2}P + \frac{1}{2}Q$. By definition, for any (x, y) in this distribution:

$$m(x, y) = \frac{1}{2}p(x, y) + \frac{1}{2}q(x, y) = \begin{cases} \frac{1}{2} & \text{if } x = 0 \text{ and } 0 \leq y \leq 1 \\ \frac{1}{2} & \text{if } x = \theta (0 \leq \theta \leq 1) \text{ and } 0 \leq y \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

To compute the **JSD** between two distributions P and Q, we use the following formula($M = \frac{1}{2}(P + Q)$):

$$JSD(P, Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M)$$

Using this formula and the given information about the distributions P and Q, we can compute:

$$KL(P||M) = \int_0^1 \log \frac{p(0, y)}{m(0, y)} p(0, y) dy = \log(2)$$

$$KL(Q||M) = \int_0^1 \log \frac{q(\theta, y)}{m(\theta, y)} q(\theta, y) dy = \log(2)$$

Therefore,

$$JSD(P,Q) = \frac{1}{2}\log(2) + \frac{1}{2}\log(2) = \log(2)$$

To compute the **KL divergence** between two distributions P and Q, we use the following formula:

$$KL(P||Q) = \int_{-\infty}^{\infty} p(x)\log\frac{p(x)}{q(x)}dx$$

For $KL(P||Q)$ we have:

$$\begin{aligned} KL(P||Q) &= \int_{-\infty}^{\infty} p(x,y)\log\frac{p(x,y)}{q(x,y)}dxdy \\ &= \int_0^1 \log\frac{p(0,y)}{q(0,y)}dy \\ &= \int_0^1 \log\frac{p(0,y)}{q(0,y|\theta=0)P[\theta=0] + q(0,y|\theta\neq 0)P[\theta\neq 0]}dy \\ &= \int_0^1 \log 1 dy \\ &= 0 \end{aligned}$$

Using the same approach as before, we can compute $KL(Q||P)$ as:

$$\begin{aligned} KL(Q||P) &= \int_{-\infty}^{\infty} q(x,y)\log\frac{q(x,y)}{p(x,y)}dxdy \\ &= \int_0^1 q(\theta,y)\log\frac{q(\theta,y)}{p(\theta,y)}dxdy \\ &= \int_0^1 q(\theta,y)\log\frac{q(\theta,y)}{p(\theta,y|\theta=0)P[\theta=0] + p(\theta,y|\theta\neq 0)P[\theta\neq 0]}dxdy \\ &= \int_0^1 \log\frac{1}{1}dy \\ &= 0 \end{aligned}$$

To compute the Earth-Mover (EM) distance between two probability distributions P and Q, we need to find the minimum cost of transforming P into Q. The cost is defined as the amount of "work" required to move the mass from one distribution to the other. In this case, we can assume that the mass is distributed uniformly over the unit interval [0, 1].
Since P has all its mass at x = 0, and Q has all its mass at some point $x = \theta$, we can think of the problem as moving the mass from x = 0 to $x = \theta$. The cost of moving a unit of mass from x to y is equal to the absolute difference between x and y. Therefore, the cost of moving the mass from P to Q is:

$$\begin{aligned} W(P,Q) &= \int_0^{\theta} |x-0|dx + \int_{\theta}^1 |x-\theta|dx \\ W(P,Q) &= \int_0^{\theta} xdx + \int_{\theta}^1 (\theta - x)dx \\ W(P,Q) &= \frac{\theta^2}{2} + \frac{(1-\theta)^2}{2} \end{aligned}$$

So, the Earth-Mover distance between P and Q is $\frac{\theta^2}{2} + \frac{(1-\theta)^2}{2}$.

2.2

To construct two distributions P and Q that are not differentiable with respect to their parameters, we can use discrete distributions with a finite number of outcomes. Let's define P and Q as follows:
P: A discrete distribution with two outcomes, A and B, with probabilities p_A and p_B, where $p_A + p_B = 1$.
Q: A discrete distribution with two outcomes, C and D, with probabilities q_C and q_D, where $q_C + q_D = 1$.
Now, let's compute the JSD between P and Q:
Compute the average distribution M:

$$M(x) = \frac{1}{2} * (P(x) + Q(x))$$

For $x = A$, $M(A) = 0.5 * (p_A + 0) = 0.5 * p_A$, For $x = B$, $M(B) = 0.5 * (p_B + 0) = 0.5 * p_B$
For $x = C$, $M(C) = 0.5 * (0 + q_C) = 0.5 * q_C$, For $x = D$, $M(D) = 0.5 * (0 + q_D) = 0.5 * q_D$
Compute the KL divergences:

$$KL(P||M) = \sum_x P(x) * log(\frac{P(x)}{M(x)})$$

$$KL(P||M) = p_A * log(\frac{p_A}{0.5 * p_A}) + p_B * log(\frac{p_B}{0.5 * p_B})$$

$$KL(Q||M) = \sum_x Q(x) * log(\frac{Q(x)}{M(x)})$$

$$KL(Q||M) = q_C * log(\frac{q_C}{0.5 * q_C}) + q_D * log(\frac{q_D}{0.5 * q_D})$$

Thus we have:

$$JSD(P, Q) = \frac{1}{2} * (KL(P||M) + KL(Q||M))$$

$$JSD(P, Q) = \frac{1}{2} * (p_A * log(2) + p_B * log(2) + q_C * log(2) + q_D * log(2))$$

$$JSD(P, Q) = log(2) * (p_A + p_B + q_C + q_D) * \frac{1}{2} = \log(2)$$

The JSD does not depend on the parameters p_A, p_B, q_C, and q_D, so it is not differentiable with respect to these parameters.
Another solution:
The JSD is defined as:
JSD(P, Q) = 1/2 * (KL(P || M) + KL(Q || M))
where KL(P || Q) is the Kullback-Leibler divergence, and M = 1/2 * (P + Q).
The Kullback-Leibler divergence is defined as:
KL(P || Q) = Σ(P(i) * log(P(i) / Q(i)))
To make JSD(P, Q) non-differentiable, we need to find P and Q such that the KL divergence is not differentiable. The KL divergence is not differentiable when P(i) > 0 and Q(i) = 0 for some i, as the logarithm is not defined for 0.
Let's construct two discrete distributions P and Q: P = [p1, p2] and Q = [q1, q2]
Let's set the parameters as follows: p1 = 1, p2 = 0 q1 = 0, q2 = 1

Now, P and Q are valid probability distributions since they sum to 1. However, the KL divergence is not differentiable for these distributions because P(1) > 0 and Q(1) = 0. Consequently, the Jensen-Shannon Divergence JSD(P, Q) is also not differentiable with respect to the parameters P and Q.

2.3

Utilizing JSD as a loss function in Generative Adversarial Networks (GANs) presents certain challenges that can impact both the training process and the quality of
1. Vanishing gradients: If the generator and discriminator distributions do not overlap (i.e., P and Q are disjoint), the JSD becomes constant, causing the gradients to vanish. This results in slow or no learning for the generator, as it lacks valuable feedback from the discriminator to enhance its samples.
2. Mode collapse: JSD's sensitivity to the relative scale of probability densities can contribute to mode collapse in GANs. This phenomenon occurs when the generator creates a limited range of samples, concentrating on a few modes of the actual data distribution rather than capturing its full diversity.
3. Non-differentiability: As shown in the previous example, JSD may be non-differentiable concerning parameters P and Q. This can create complications during optimization, as gradient-based optimization techniques depend on the presence of gradients to update model parameters.
4. Balancing training difficulties: GANs consist of a two-player minimax game between the generator and the discriminator. Employing JSD as a loss function can make it difficult to balance the training of both networks, as the generator may not receive sufficient feedback for improvement when the discriminator is too powerful.

3

3.1

We can rewrite loss expression using integrals:

$$L_D(\phi;\theta) = -\int_x p_{data}(x)\log D_\phi(x)dx - \int_x p_\theta(x)\log(1-D_\phi(x))dx$$

Our goal is to find the optimal D_ϕ that minimizes L_D. To do that, we take the derivative of L_D with respect to D_ϕ and set it to zero:

$$\frac{\partial L_D}{\partial D_\phi} = -\frac{p_{data}(x)}{D_\phi} + \frac{p_\theta(x)}{1-D_\phi} = 0$$

Solving for D_ϕ gives us:

$$p_{data}(x) - D_\phi p_{data}(x) = D_\phi p_\theta(x)$$

$$D_\phi(p_{data}(x) + p_\theta(x)) = p_{data}(x)$$

$$D_\phi = \frac{p_{data}(x)}{p_{data}(x) + p_\theta(x)}$$

Therefore, we have shown that the discriminator loss function L_D minimizes when $D_\phi = \frac{p_{data}(x)}{p_{data}(x)+p_\theta(x)}$.

3.2

Substituting the value of $D_\phi(x)$ into the equation $D_\phi(x) = \sigma(h_\phi(x))$.

$$\sigma(h_\phi(x)) = \frac{p_{data}(x)}{p_{data}(x) + p_\theta(x)}$$

Now, to find the value of $h_\phi(x)$, we can apply the inverse of the sigmoid function, which is the logit function. The logit function is defined as:

$$logit(p) = log\left(\frac{p}{1-p}\right)$$

So, applying the logit function to both sides of the equation, we get:

$$h_\phi(x) = \sigma^{-1}\left(\frac{p_{data}(x)}{p_{data}(x) + p_\theta(x)}\right) = logit\left(\frac{p_{data}(x)}{p_{data}(x) + p_\theta(x)}\right)$$

Since $\sigma^{-1}(p)$ equals $logit(p)$, we can simplify the equation further as:

$$h_\phi(x) = log\left(\frac{\frac{p_{data}(x)}{p_{data}(x)+p_\theta(x)}}{1 - \frac{p_{data}(x)}{p_{data}(x)+p_\theta(x)}}\right)$$

$$h_\varphi(x) = log\left(\frac{p_{data}(x)}{p_\theta(x)}\right)$$

Thus, we have successfully shown that if $D_\phi(x) = \frac{p_{data}(x)}{p_{data}(x)+p_\theta(x)}$, then $h_\phi(x) = log\left(\frac{p_{data}(x)}{p_\theta(x)}\right)$. This means that the logit vector $h_\phi(x)$ approximates the logarithm of the ratio between the real data distribution and the model's approximating distribution.

3.3

$$L_G(\theta, \phi) = \int_{x\sim p_\theta} p_\theta(x)\log(1 - D_\phi(x))dx - \int_{x\sim p_\theta} p_\theta(x)\log D_\phi(x)dx$$

Substituting value of $D_\phi(x)$ in to the equation we have:

$$L_G(\theta, \phi) = \int_{x\sim p_\theta} p_\theta(x)\log(\frac{p_\theta(x)}{p_{data}(x) + p_\theta(x)})dx - \int_{x\sim p_\theta} p_\theta(x)\log(\frac{p_{data}(x)}{p_{data}(x) + p_\theta(x)})dx$$

$$= \int_{x\sim p_\theta} p_\theta(x)\log(\frac{p_\theta(x)}{p_{data}(x)})$$

Considering KL Divergence formula we can write last equation as follows:

$$L_G(\theta, \phi) = KL(p_\theta(\mathbf{x})||p_{data}(\mathbf{x}))$$

3.4

Let's express the negative log-likelihood of the data in terms of KL divergence. The negative log-likelihood of the data can be written as:

$$-E_{x\sim p_{data}(x)}[\log(p_\theta(x))]$$

Now, let's introduce a variational distribution $q_\phi(z|x)$ and rewrite the expression using the Evidence Lower BOund (ELBO):

$$-E_{x\sim p_{data}(x)}[\log(p_\theta(x))] = E_{x\sim p_{data}(x)}[KL(q_\phi(z|x)||p_\theta(z|x))] - \mathcal{L}(\theta,\phi)$$

Here, $\mathcal{L}(\theta,\phi)$ is the ELBO, which is a lower bound on the log-likelihood of the data. The first term on the right-hand side is the KL divergence between the variational distribution and the true posterior distribution.
Now, let's consider the GAN generator loss function:

$$L_G(\theta,\phi) = E_{x\sim p_\theta(x)} \log(1 - D_\phi(x))dx - E_{x\sim p_\theta(x)} \log D_\phi(x)dx$$

The goal of the GAN generator is to minimize this loss function, which encourages the generator to produce samples that the discriminator cannot distinguish from real data.
Comparing the VAE decoder and the GAN generator, we can see that both models aim to generate samples that resemble the true data distribution. However, their training objectives are different:
1. The VAE decoder is trained to minimize the KL divergence between the variational distribution and the true posterior distribution, which encourages the decoder to generate samples that are likely under the true data distribution.
2. The GAN generator is trained to minimize the generator loss function, which encourages the generator to produce samples that the discriminator cannot distinguish from real data.
In summary, while both VAE decoders and GAN generators aim to generate realistic samples, their training objectives are different. VAEs focus on minimizing the KL divergence between the variational distribution and the true posterior distribution, while GANs focus on fooling the discriminator. These differences in training objectives can lead to different properties and performance characteristics in the generated samples.

4

4.1

Using the reparameterization technique, we can express x_t as $x_t = \sqrt{\alpha_t}x_{t-1} + \epsilon_{t-1}$. Where $\epsilon_{t-1} \sim N(0, \beta_t I)$. Now, let's expand this recursively:

$$x_t = \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}x_{t-2} + \epsilon_{t-2}) + \epsilon_{t-1}$$

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t}\epsilon_{t-2} + \epsilon_{t-1}$$

Then we have:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sum_{i=0}^{t-1}\sqrt{\frac{\alpha_{t-i}}{\bar{\alpha}_i}}\epsilon_i$$

Now, we want to find the distribution of $q(x_t|x_0)$. Since the sum of Gaussian random variables is also Gaussian, we can write:

$$q(x_t|x_0) = N(x_t|\mu_t, \Sigma_t) = N(x_t|\sqrt{\bar{\alpha}_t}x_0, \sum_{i=0}^{t-1} \frac{\alpha_{t-i}}{\bar{\alpha}_i}\beta_i I)$$

We know that $\beta_i = 1 - \alpha_i$, so we can rewrite the covariance matrix as:

$$\Sigma_t = \sum_{i=0}^{t-1} \frac{\alpha_{t-i}}{\bar{\alpha}_i}(1-\alpha_i)I \Rightarrow \Sigma_t = \sum_{i=0}^{t-1} \frac{\alpha_{t-i} - \alpha_{t-i}^2}{\bar{\alpha}_i} I$$

Since $\bar{\alpha}_t = \Pi_{i=1}^{t}\alpha_i$, we have:

$$\frac{\alpha_{t-i} - \alpha_{t-i}^2}{\bar{\alpha}_i} = \frac{\alpha_{t-i} - \alpha_{t-i}^2}{\bar{\alpha_{i-1}}\alpha_i} = \frac{\alpha_{t-i} - \alpha_{t-i}^2}{\bar{\alpha_{t-1}}} \Rightarrow \Sigma_t = \sum_{i=0}^{t-1}(1-\bar{\alpha_{t-1}})I = (1-\bar{\alpha_{t-1}})I$$

Therefore, we have:

$$q(x_t|x_0) = N(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

4.2

For x_T to be perfect Gaussian noise, it should be independent of x_0 and have a Gaussian distribution with zero mean and identity covariance matrix. In other words, we want
$q(x_T|x_0) = N(x_T|0, I)$, And from the previous result, we have
$q(x_T|x_0) = N(x_T|\sqrt{\bar{\alpha_T}}x_0, (1-\bar{\alpha_T})I)$.
To satisfy the condition for perfect Gaussian noise, we need the following:
The mean should be zero: $\sqrt{\bar{\alpha_T}}x_0 = 0$.
Since x_0 is a real image and is non-zero, the only way to satisfy this condition is if $\bar{\alpha_T} = 0$.
The covariance matrix should be the identity matrix $(1-\bar{\alpha_T})I = I$.
This condition is automatically satisfied when $\bar{\alpha_T} = 0$.
So, the necessary condition for x_T to be perfect Gaussian noise is that $\bar{\alpha_T} = 0$. This means that $\Pi_{t=1}^{T}\alpha_t = 0$.

4.3

The issue is that we require knowledge of the true data distribution to make this calculation. Unfortunately, this information is often unknown and impractical to obtain in practice. Essentially, we would have to know everything about real images and their relationships to each other during the reverse diffusion process to calculate $q(x_{t-1}|x_t)$ directly.
the reverse diffusion process involves going from a high-entropy state (noisy image x_T) to a low-entropy state (original image x_0).Recovering the structure and information present in the original image from the noisy image can be a complicated and nonlinear transformation, making this process inherently difficult.

4.4

To show that $q(x_{t-1}|x_t, x_0)$ is computable, we'll use Bayes' rule. First, let's recall that:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

We already know from previous derivations that:

$$q(x_t|x_{t-1}) = N(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_t|x_0) = N(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

$$q(x_{t-1}|x_0) = N(x_{t-1}|\sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})I)$$

Now, let's plug these expressions into the Bayes' rule formula:

$$q(x_{t-1}|x_t, x_0) = \frac{N(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)N(x_{t-1}|\sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})I)}{N(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)}$$

Since all the distributions are Gaussian, we can compute the product of the two Gaussian distributions in the numerator and then divide by the Gaussian distribution in the denominator. After some algebraic manipulations, we obtain:

$$q(x_{t-1}|x_t, x_0) = N(x_{t-1}|\tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

where

$$\tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{1-\bar{\alpha}_t}\epsilon_t), \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Thus, we have shown that $q(x_{t-1}|x_t, x_0)$ is computable using the given expressions for the mean and covariance.

4.5

We can rewrite given expression for L_t using the definition of KL-Divergence($D_{KL}(P\|Q) = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right]$):

$$L_t = D_{KL}\left(q\left(x_{-1} \mid x_t, x_0\right) \| p_\theta\left(x_{t-1} \mid x_t\right)\right) = \mathbb{E}_{x_{t-1}\sim q(x_{t-1}|x_t,x_0)}\left[\log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right]$$

Now, let's substitute the expressions for $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$:

$$q(x_{t-1}|x_t, x_0) = N(x_{t-1}|\tilde{\mu}(x_t, x_0), \tilde{\beta}_t I), \quad p_\theta(x_{t-1}|x_t) = N(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Using these expressions, we can rewrite the expectation in L_t as:

$$L_t = \mathbb{E}_{x_{t-1}\sim q(x_{t-1}|x_t,x_0)}\left[\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right]$$

Now, let's focus on the argument of the logarithm. We can rewrite it as:

$$\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} = \frac{\exp\left(-\frac{1}{2\tilde{\beta}_t}\|x_{t-1} - \tilde{\mu}(x_t, x_0)\|^2\right)}{\exp\left(-\frac{1}{2}\left(x_{t-1} - \mu_\theta(x_t, t)\right)^T \Sigma_\theta^{-1}(x_t, t)\left(x_{t-1} - \mu_\theta(x_t, t)\right)\right)}$$

Taking the logarithm of this expression, we get:

$$\log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} = \frac{1}{2\tilde{\beta}_t}\|x_{t-1} - \tilde{\mu}(x_t, x_0)\|^2 - \frac{1}{2}\left(x_{t-1} - \mu_\theta(x_t, t)\right)^T \Sigma_\theta^{-1}(x_t, t)\left(x_{t-1} - \mu_\theta(x_t, t)\right) + C$$

where C is a constant term that does not depend on x_{t-1}. Now let's substitute the expression for $\tilde{\mu}(x_t, x_0)$, and with reparameterization trick we have:

$$\tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{1-\bar{\alpha}_t}\epsilon_t), x_{t-1} = \sqrt{\alpha_t}x_t + \sqrt{1-\alpha_t}\epsilon_t$$

Substituting this expression in the expectation, we get:

$$L_t = \mathbb{E}_{x_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t\left(1-\bar{\alpha}_t\right)\left\|\Sigma_\theta\right\|_2^2}\left\|\epsilon_t - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t\right)\right\|_2^2\right]$$

4.6

While using ϵ_θ and parameter change in μ_θ estimation can make the loss function simpler, it can also make it harder to understand how the algorithm converges. This is because the use of Langevin dynamics to sample from the posterior distribution can introduce additional complexity in the convergence analysis. Langevin dynamics is a process that involves both deterministic and random components, and its convergence properties can be difficult to analyze. Additionally, the use of ϵ_θ and parameter change in μ_θ estimation can introduce more randomness and nonlinearity into the optimization problem, which can make it harder to analyze the convergence of the algorithm. Therefore, it is important to carefully analyze the convergence properties of the algorithm and consider the potential sources of complexity that may arise from using these techniques.