Generative Models - Assignment 3

Mohammad Mohammadi - 402208592

December 14, 2024

Problem 1:

We have a parameter $\mu \in \mathbb{R}$ and a random variable z distributed as:

$$z \sim N(\mu, 1).$$

We want to estimate:

$$\eta(\mu) = \nabla_\mu E_{z\sim N(\mu,1)}[f(z)].$$

Part A: Reparameterization Trick (Pathwise Derivative)

The reparameterization trick expresses a random variable $z \sim N(\mu, 1)$ as:

$$z = \mu + \epsilon, \quad \epsilon \sim N(0, 1).$$

Which yields:

$$E_{z\sim N(\mu,1)}[f(z)] = E_{\epsilon\sim N(0,1)}[f(\mu + \epsilon)].$$

Since ϵ is independent of μ, we can differentiate inside:

$$\eta(\mu) = \nabla_\mu E_\epsilon[f(\mu + \epsilon)] = E_\epsilon[f'(\mu + \epsilon)].$$

Monte Carlo Approximation:

1. Sample $\epsilon_i \sim N(0, 1)$ for $i = 1, \ldots, N$.
2. Let $z_i = \mu + \epsilon_i$.
3. Estimate:

$$\hat{\eta}(\mu) = \frac{1}{N} \sum_{i=1}^{N} f'(z_i).$$

Example 1: $f(z) = mz$

$$f'(z) = m \implies \eta(\mu) = E_\epsilon[m] = m.$$

Since $E[f(z)] = E[mz] = m\mu$, we have $\frac{d}{d\mu}(m\mu) = m$. Thus, they both match.

Example 2: $f(z) = (mz)^2$

$$f(z) = m^2 z^2 \implies f'(z) = 2m^2 z.$$

Thus:

$$\eta(\mu) = E_{z\sim N(\mu,1)}[2m^2 z] = 2m^2 E[z] = 2m^2\mu.$$

Check by direct differentiation:

$$E[f(z)] = E[m^2 z^2] = m^2(\mu^2 + 1).$$

Then:

$$\frac{d}{d\mu}(m^2(\mu^2 + 1)) = 2m^2\mu,$$

Thus, they both match..

Part B: Score Function Estimator (REINFORCE)

REINFORCE

The REINFORCE algorithm [7], also known as the score function estimator [4], uses a simple differentiation rule called the *log-derivative trick*, which is simply the differentiation rule for the logarithm:

$$\nabla_\theta p_\theta(x) = p_\theta(x)\nabla_\theta \log p_\theta(x). \tag{1}$$

Although written as above, the 'trick' seems very plain, it is very useful in situations where p_θ is the likelihood for a random variable . The term $\nabla_\theta \log p_\theta(x)$ is called the *score* and regularly comes up in maximum likelihood estimation. It also has many wonderful properties, like having zero expected value (which proves useful when using it for variational inference, among other things).

With this, we get back to our problem of estimating the gradient. Using the definition of expectation,

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] = \nabla_\theta \int f(x)p_\theta(x)\,dx \tag{2}$$

$$= \int f(x)\nabla_\theta p_\theta(x)\,dx \tag{3}$$

$$= \int f(x)p_\theta(x)\nabla_\theta \log p_\theta(x)\,dx \tag{4}$$

$$= \mathbb{E}_{x\sim p_\theta(x)}[f(x)\nabla_\theta \log p_\theta(x)]. \tag{5}$$

The reason why the integral and differentiation can be switched in Equation (3) is because of the *Leibniz integral rule*. Equation (4) is just the application of the log-derivative trick from Equation (1). Now, since we know the distribution under the expectation, we can use Monte Carlo sampling to approximate the expectation:

$$\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)\nabla_\theta \log p_\theta(x_i). \tag{6}$$

The score function estimator therefore, in the assignment denoted by $\beta(\mu)$, is:

$$\eta(\mu) = \nabla_\mu E_{z\sim N(\mu,1)}[f(z)] = E_z[f(z)\nabla_\mu \log p(z;\mu)].$$

As seen in [5].

For $z \sim N(\mu, 1)$:

$$p(z;\mu) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(z-\mu)^2}{2}\right), \quad \log p(z;\mu) = -\frac{1}{2}\log(2\pi) - \frac{(z-\mu)^2}{2}.$$

Differentiating w.r.t. μ:

$$\nabla_\mu \log p(z;\mu) = z - \mu.$$

Thus:

$$\eta(\mu) = E_z[f(z)(z-\mu)].$$

Monte Carlo Approximation:

1. Sample $z_i \sim N(\mu, 1)$ for $i = 1, \ldots, N$.
2. Compute:

$$\beta(\mu) = \frac{1}{N}\sum_{i=1}^{N} f(z_i)(z_i - \mu).$$

For the functions we have:

Example 1: $f(z) = mz$

$$\eta(\mu) = E_z[mz(z-\mu)] = mE[z(z-\mu)].$$

For $z \sim N(\mu, 1)$:

$$E[z(z-\mu)] = E[z^2] - \mu E[z] = (\mu^2 + 1) - \mu^2 = 1.$$

Hence:

$$\eta(\mu) = m \cdot 1 = m.$$

Example 2: $f(z) = (mz)^2 = m^2 z^2$

$$\eta(\mu) = E_z[m^2 z^2(z-\mu)] = m^2 E[z^2(z-\mu)].$$

We know $E[z] = \mu$, $E[z^2] = \mu^2 + 1$, and $E[z^3] = \mu^3 + 3\mu$.

So:

$$E[z^2(z-\mu)] = E[z^3] - \mu E[z^2] = (\mu^3 + 3\mu) - \mu(\mu^2 + 1) = \mu^3 + 3\mu - \mu^3 - \mu = 2\mu.$$

Thus:

$$\eta(\mu) = m^2 \cdot 2\mu = 2m^2\mu.$$

Matches the pathwise result.

Part C: Measure-Valued (Weak Derivative) Estimator

Concept: A measure-valued estimator (weak derivative) uses a symmetric perturbation approach with a specific distribution. According to the corrected problem statement, the form is:

$$\gamma(\mu) = \frac{1}{N\sqrt{2\pi}} \sum_{i=1}^{N} [f(\mu + y_i^+) - f(\mu - y_i^-)].$$

Here, y_i^+ and y_i^- are drawn from a distribution chosen so that $E[\gamma(\mu)] = \eta(\mu)$. The problem mentions a Weibull distribution with $\lambda = \sqrt{2}$ and $k = 2$:

$$Y = \lambda(-\ln(U))^{1/k}, \quad U \sim \text{Uniform}(0, 1).$$

With $\lambda = \sqrt{2}$, $k = 2$:

$$Y = \sqrt{2}\sqrt{-\ln(U)}.$$

By appropriate scaling and symmetrical evaluation $f(\mu+Y)$ and $f(\mu-Y)$, one can show the expectation matches the desired gradient. This is known as a weak derivative or measure-valued estimator.

If we plot these three estimators for these 2 functions we can see the results are as below:

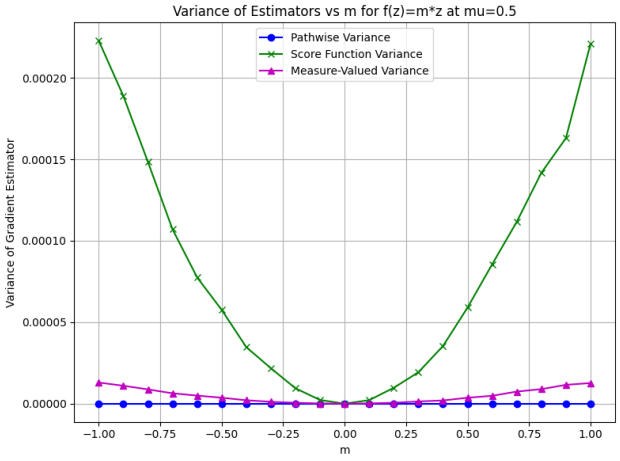

Figure 1: Variance of Estimators vs m for $f(z) = mz$ at $\mu = 0.5$.

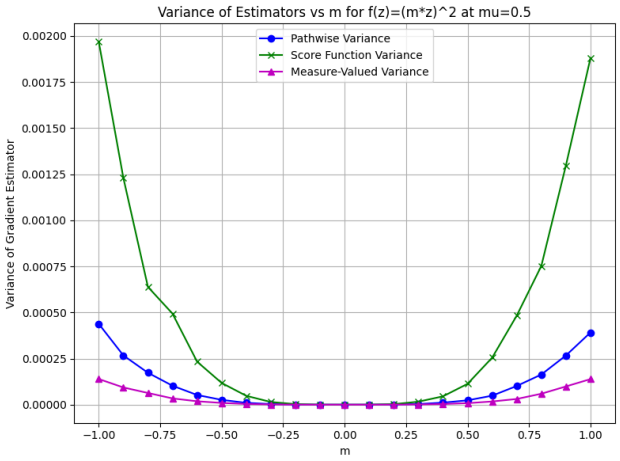

Figure 2: Variance of Estimators vs m for $f(z) = (mz)^2$ at $\mu = 0.5$.

By plotting we can see that:

- The pathwise estimator will typically have low variance and match the analytical curve closely.
- The score function estimator will have correct expectation but may exhibit higher variance.
- The measure-valued estimator also converges to the correct gradient and may have different variance characteristics compared to the score function estimator.

As N increases, all methods converge to the true gradient, but their rate and variance may differ. So, I guess we should choose an appropriate one based on our use case.

Problem 2:

Part A:

Intuitionally the Cauchy-Schwarz measure D_{CS} essentially captures how well p and q overlap. The KL divergences, on the other hand, heavily penalize differences especially in the tails. Typically, D_{CS} grows more slowly than the KL divergence as distributions diverge. Hence, it generally stays below both $D_{KL}(p\|q)$ and $D_{KL}(q\|p)$.

Now consider two single-variate normal distributions:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right), \quad q(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right).$$

They have the same variance σ^2 but possibly different means μ_1 and μ_2. (as question mentioned they have different averages but this holds even if they have equall averages, which is the boundry condition). We want to verify the inequality:

$$D_{CS}(p\|q) \leq \min\{D_{KL}(p\|q), D_{KL}(q\|p)\}.$$

Computing $D_{CS}(p\|q)$ By definition:

$$D_{CS}(p\|q) = -\log\left(\frac{\int p(x)q(x)dx}{\sqrt{\int p(x)^2dx \int q(x)^2dx}}\right).$$

We need to compute these integrals.

Integral $\int p(x)q(x)dx$:

$$p(x)q(x) = \frac{1}{(\sqrt{2\pi}\sigma)^2} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right) \exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right).$$

Combine the exponents:

$$(x-\mu_1)^2 + (x-\mu_2)^2 = 2(x-m)^2 + \frac{(\mu_1-\mu_2)^2}{2},$$

where $m = \frac{\mu_1+\mu_2}{2}$.

Thus:

$$p(x)q(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{2(x-m)^2 + (\mu_1-\mu_2)^2/2}{2\sigma^2}\right).$$

This simplifies to:

$$p(x)q(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right) \exp\left(-\frac{(x-m)^2}{\sigma^2}\right).$$

Integrating w.r.t. x:

$$\int_{-\infty}^{\infty} p(x)q(x)dx = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right) \int_{-\infty}^{\infty} \exp\left(-\frac{(x-m)^2}{\sigma^2}\right) dx.$$

Make a substitution $u = \frac{x-m}{\sigma}$, $dx = \sigma du$:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{(x-m)^2}{\sigma^2}\right) dx = \sigma \int_{-\infty}^{\infty} e^{-u^2} du = \sigma\sqrt{\pi}.$$

Thus:

$$\int p(x)q(x)dx = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right) \cdot (\sigma\sqrt{\pi}) = \frac{\sqrt{\pi}}{2\pi\sigma} \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right).$$

Simplify:

$$\int p(x)q(x)dx = \frac{1}{2\sqrt{\pi}\sigma} \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right).$$

Integral $\int p(x)^2dx$:

$$p(x)^2 = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^2 \exp\left(-\frac{(x-\mu_1)^2}{\sigma^2}\right) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu_1)^2}{\sigma^2}\right).$$

Integrate over x:

$$\int p(x)^2dx = \frac{1}{2\pi\sigma^2} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu_1)^2}{\sigma^2}\right) dx.$$

Using the same substitution method:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu_1)^2}{\sigma^2}\right) dx = \sigma\sqrt{\pi}.$$

So:

$$\int p(x)^2dx = \frac{1}{2\pi\sigma^2}(\sigma\sqrt{\pi}) = \frac{\sqrt{\pi}}{2\pi\sigma} = \frac{1}{2\sqrt{\pi}\sigma}.$$

Similarly:

$$\int q(x)^2dx = \frac{1}{2\sqrt{\pi}\sigma}.$$

Now:

$$\sqrt{\int p(x)^2dx \int q(x)^2dx} = \sqrt{\frac{1}{2\sqrt{\pi}\sigma} \cdot \frac{1}{2\sqrt{\pi}\sigma}} = \frac{1}{2\sqrt{\pi}\sigma}.$$

Putting it together we get:

$$\frac{\int p(x)q(x)dx}{\sqrt{\int p(x)^2dx \int q(x)^2dx}} = \frac{\frac{1}{2\sqrt{\pi}\sigma}\exp(-(\mu_1-\mu_2)^2/(4\sigma^2))}{\frac{1}{2\sqrt{\pi}\sigma}} = \exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right).$$

Thus:

$$D_{CS}(p\|q) = -\log\left(\exp\left(-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}\right)\right) = \frac{(\mu_1-\mu_2)^2}{4\sigma^2}.$$

Now computing $D_{KL}(p\|q)$ and $D_{KL}(q\|p)$ for equal variances

For two normal distributions with the same variance σ^2:

$$D_{KL}(p\|q) = \frac{(\mu_1-\mu_2)^2}{2\sigma^2}, \quad D_{KL}(q\|p) = \frac{(\mu_2-\mu_1)^2}{2\sigma^2}.$$

Since $(\mu_2-\mu_1)^2 = (\mu_1-\mu_2)^2$, we have:

$$D_{KL}(p\|q) = D_{KL}(q\|p) = \frac{(\mu_1-\mu_2)^2}{2\sigma^2}.$$

Now we can verify the inequality

$$D_{CS}(p\|q) \leq \min\{D_{KL}(p\|q), D_{KL}(q\|p)\}.$$

Since $D_{KL}(p\|q) = D_{KL}(q\|p)$ in this symmetric variance case:

$$\min\{D_{KL}(p\|q), D_{KL}(q\|p)\} = D_{KL}(p\|q) = \frac{(\mu_1-\mu_2)^2}{2\sigma^2}.$$

We have:

$$D_{CS}(p\|q) = \frac{(\mu_1-\mu_2)^2}{4\sigma^2}.$$

Comparing, we get:

$$\frac{(\mu_1-\mu_2)^2}{4\sigma^2} \leq \frac{(\mu_1-\mu_2)^2}{2\sigma^2}.$$

Since $(\mu_1-\mu_2)^2 \geq 0$ and $\sigma^2 > 0$, we can divide both sides by $(\mu_1-\mu_2)^2/(\sigma^2)$ (unless $\mu_1 = \mu_2$, in which case both sides are zero and the inequality holds trivially). The inequality reduces to:

$$\frac{1}{4} \leq \frac{1}{2},$$

which is clearly true.

Thus, for the single-variate Gaussian case with equal variances:

$$D_{CS}(p\|q) \leq D_{KL}(p\|q) = D_{KL}(q\|p),$$

and therefore:

$$D_{CS}(p\|q) \leq \min\{D_{KL}(p\|q), D_{KL}(q\|p)\}.$$

Hence, we have proved that the inequality holds.

Part B:

In standard Variational Autoencoder (VAE) framework, the marginal likelihood of observed data x can be decomposed using the following well-known identity:

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z)) + D_{KL}(q_\phi(z|x)\|p(z|x)).$$

This identity can be derived from Bayes' rule and the definition of KL divergence:

$$p(z|x) = \frac{p_\theta(x|z)p(z)}{p(x)} \implies \log p(x) = \log p_\theta(x|z) + \log p(z) - \log p(z|x).$$

Taking the expectation w.r.t. $q_\phi(z|x)$:

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \mathbb{E}_{q_\phi(z|x)}[\log p(z)] - \mathbb{E}_{q_\phi(z|x)}[\log p(z|x)].$$

By adding and subtracting $\mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x)]$ and rearranging terms, we obtain:

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z)) + D_{KL}(q_\phi(z|x)\|p(z|x)).$$

This is a standard identity that expresses the log-evidence $\log p(x)$ in terms of expectation under $q_\phi(z|x)$, the KL divergence from q_ϕ to the prior $p(z)$, and the KL divergence from q_ϕ to the posterior $p(z|x)$.

Now, considering the modified objective mentioned in the problem:

$$\mathcal{L}_{CS}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \lambda D_{CS}(q_\phi(z|x)\|p(z)),$$

where D_{CS} is the Cauchy–Schwarz divergence measure and $\lambda > 0$ is a penalty coefficient.

Our goal is to rewrite $\mathcal{L}_{CS}$ in a form analogous to the standard identity for $\log p(x)$. Starting from the known identity for $\log p(x)$, we substitute it into the expression for $\mathcal{L}_{CS}$:

First, add and subtract $\log p(x)$ inside $\mathcal{L}_{CS}$:

$$\mathcal{L}_{CS}(x;\theta,\phi) = \log p(x) - D_{KL}(q_\phi(z|x)\|p(z|x)) + D_{KL}(q_\phi(z|x)\|p(z)) - \lambda D_{CS}(q_\phi(z|x)\|p(z)).$$

1. Start from the given $\log p(x)$ decomposition:

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z)) + D_{KL}(q_\phi(z|x)\|p(z|x)).$$

2. From the definition of $\mathcal{L}_{CS}$:

$$\mathcal{L}_{CS}(x;\theta,\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \lambda D_{CS}(q_\phi(z|x)\|p(z)).$$

3. Substitute the term $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ from the $\log p(x)$ identity:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] = \log p(x) + D_{KL}(q_\phi(z|x)\|p(z|x)) - D_{KL}(q_\phi(z|x)\|p(z)).$$

4. Thus:

$$\mathcal{L}_{CS}(x;\theta,\phi) = \big(\log p(x) + D_{KL}(q_\phi(z|x)\|p(z|x)) - D_{KL}(q_\phi(z|x)\|p(z))\big) - \lambda D_{CS}(q_\phi(z|x)\|p(z)).$$

5. Rearrange terms to match the desired form:

$$\mathcal{L}_{CS}(x;\theta,\phi) = \log p(x) - D_{KL}(q_\phi(z|x)\|p(z|x)) + D_{KL}(q_\phi(z|x)\|p(z)) - \lambda D_{CS}(q_\phi(z|x)\|p(z)).$$

This is exactly the decomposition presented in the problem. It shows how the modified objective $\mathcal{L}_{CS}$ relates to the original evidence $\log p(x)$, the posterior $p(z|x)$, and the prior $p(z)$, along with the new D_{CS} regularization term.

Decomposition of the Modified Objective

It is stated that can be $\mathcal{L}_{CS}$ decomposed as:

$$\mathcal{L}_{CS} = \log p(x) - D_{KL}(q_\phi(z|x)\|p(z|x)) + D_{KL}(q_\phi(z|x)\|p(z)) - \lambda D_{CS}(q_\phi(z|x)\|p(z)).$$

This decomposition is instructive because it shows how the new objective relates to known quantities and how the new D_{CS} term fits in.

1. $\log p(x)$: This is the (log)-marginal likelihood of the observed data x. It is a term that does not depend on the variational parameters ϕ or the model parameters θ directly in the optimization since $p(x) = \int p_\theta(x|z)p(z)dz$ is fixed for given θ. The purpose of variational inference is often seen as maximizing a lower bound to this quantity. Including it in the decomposition is mostly conceptual, helping us understand what the rest of the terms represent relative to the true evidence.

2. $-D_{KL}(q_\phi(z|x)\|p(z|x))$: This term compares the variational posterior $q_\phi(z|x)$ to the true posterior $p(z|x)$. The KL divergence $D_{KL}(q_\phi(z|x)\|p(z|x))$ measures how well q_ϕ approximates the exact posterior. Minimizing this divergence is the ultimate goal of Bayesian inference. Having a $-D_{KL}$ term means we are adding an incentive for $q_\phi(z|x)$ to approach the true posterior $p(z|x)$. In an ideal scenario (with infinite model capacity and data), this term would vanish as $q_\phi(z|x) \to p(z|x)$.

3. $+D_{KL}(q_\phi(z|x)\|p(z))$: In the standard VAE, we have $-D_{KL}(q_\phi(z|x)\|p(z))$ which penalizes deviations of $q_\phi(z|x)$ from the prior $p(z)$. Here, after rearranging, it appears with a positive sign. The presence of this term in the decomposition helps us see the "tension" in the solution: we want $q_\phi(z|x)$ to be close to the true posterior $p(z|x)$ (which may not be simple), and we also want to consider how complex or different $q_\phi(z|x)$ is from the simpler prior $p(z)$.

In simpler terms, this term can be thought of as a complexity penalty: if $q_\phi(z|x)$ deviates significantly from the prior $p(z)$, then this KL will be large, indicating a "complex" posterior that must be justified by better explaining the data.

4. $-\lambda D_{CS}(q_\phi(z|x)\|p(z))$: This is the new term introduced to handle complex distributions more gracefully. By adding $-\lambda D_{CS}$, we are penalizing large Cauchy-Schwarz divergence between $q_\phi(z|x)$ and $p(z)$. Unlike the standard KL, the CS-based measure could be more tractable or provide a better gradient signal when working with more flexible distributions (like Gaussian mixture models).

Adjusting λ controls how strongly this term influences the optimization. Increasing λ places more emphasis on making $q_\phi(z|x)$ close to $p(z)$ under the D_{CS} measure. This could lead to posterior distributions that are more "regularized" or simpler, potentially improving generalization or stability at the cost of possibly reducing the capacity to model complex posteriors if λ is too large.

Interpretation and Effects of Increasing λ

When we increase λ, we are effectively increasing the importance of the D_{CS} penalty relative to the reconstruction and other divergence terms. It effects:

1. If D_{CS} is easier or more stable to approximate for our chosen distributions, a larger λ might lead to more stable or robust inference.

2. A higher λ might prevent $q_\phi(z|x)$ from becoming too different from the prior, acting as a stronger regularizer. This can help control overfitting, analogous to how more weight on the KL term in a standard VAE shrinks the posterior distribution closer to the prior.

3. If λ is too large, we might overly constrain $q_\phi(z|x)$, making it less capable of capturing the complexities of the true posterior and thus harming the overall generative performance.

For verification and integral results of Gaussian distributions and divergences, see also [3, 1, 6, 2].

Problem 3:

It is fully implemented ans the questions are answered all the in the GM_HW3_Q3.ipynb file.

Problem 4:

Part A - The architecture of Conditional Generative Adversarial Networks (CGAN):

A Conditional Generative Adversarial Network (CGAN) is a type of GAN architecture where both the generator and the discriminator are conditioned on some extra information in addition to the input noise vector. In a standard GAN, the generator receives only a random noise vector and tries to produce realistic outputs from it, with the discriminator distinguishing between real and fake samples. In a CGAN, both the generator and the discriminator also receive conditional information (like a class label, an input image, a map, or any auxiliary data).

A CGAN differs from a regular GAN in these aspects:

- **Generator (G):** Receives a noise and also a condition (for instance, an input image and class label) and generates output based on this condition.
- **Discriminator (D):** Receives both the generated (or real) image and the same condition and learns whether the output image is both realistic and consistent with the given condition.

The main advantage of CGAN is that it allows control over the characteristics of the generated output, which is crucial for tasks like image-to-image translation. For example, in Pix2Pix, the input image acts as the condition provided to both the generator and the discriminator to achieve targeted image transformations.

Part B - Image Translation:

Image translation is the process of converting an image from one domain (or representation) to another. Three significant applications are:

1. **Converting sketches to photorealistic images:** In this application, the model transforms a simple sketch or a drawn outline into a realistic image, for example turning line drawings into detailed cityscapes or faces. This is useful for artists and designers for rapid concept generation. https://arxiv.org/abs/1611.07004 demonstrates such applications using paired datasets of sketches and corresponding real images.

2. **Colorizing black-and-white images:** This use case is about adding color to grayscale images. Image translation can learn the appropriate color palette to produce realistic colorized versions of old photographs. "Colorful Image Colorization" (Zhang et al., ECCV 2016)/

3. **Transforming realistic images into paintings (style transfer):** Style transfer is a subset of image translation where the style of one image is applied to another (usually style of a painting to a real-world photo. E.g., transferring style of Starry Night by Van Gogh to a family portrait used to be an artistic trend back in 2017-2018), producing a new image with the artistic style. https://arxiv.org/abs/1703.10593 is a prominent work that translates images between domains without paired data, for example, turning landscape photographs into Van Gogh-style paintings.

Part C - Pix2Pix Vs. CycleGAN:

Pix2Pix is a CGAN-based model that requires paired data. For each input image in the source domain, there must be a corresponding target image in the destination domain. For example, to convert edge maps to realistic photos, you need pairs of edge-images and their corresponding real photographs.

- **Generator in Pix2Pix:** Uses a U-Net architecture, which preserves both global structural information and local details through skip connections. This helps maintain the spatial structure of the input.
- **Discriminator in Pix2Pix:** Often uses a PatchGAN architecture. Instead of classifying the entire image as real or fake, it operates on local patches of the image, encouraging the generator to produce realistic details at a local scale.

CycleGAN addresses scenarios where paired data is not available. It employs two generators and two discriminators, enabling translation from domain X to Y and back from Y to X. A cycle consistency loss ensures that translating from X to Y and then back to X results in a reconstruction close to the original image.

- **Generator in CycleGAN:** Uses two generators, each responsible for one direction of translation, generally based on encoder-decoder architectures.

- **Discriminator in CycleGAN:** Two discriminators, one for each direction, often based on PatchGAN to evaluate local realism.

Differences:

- Pix2Pix requires paired data and works optimally with a U-Net generator.
- CycleGAN does not need paired data, relying on cycle consistency loss to learn mappings.
- Pix2Pix is ideal for tasks where you have aligned pairs of images, whereas CycleGAN works great in scenarios where such aligned data does not exist.

Part D - Answering the next 4 parts based on Pix2Pix model:

1. **Difference between Pix2Pix and a simple GAN:** In a simple GAN, the goal is to make the generator produce images that resemble the distribution of real images, with no additional input conditions. The discriminator only checks if the image looks real or fake.

In Pix2Pix, we have a CGAN setup. The generator and discriminator are conditioned on an input image. The generator must produce an output that is not only realistic but also consistent with the input condition image. The discriminator checks both the input condition and the output image to decide if they are consistent and plausible together. This structure enables solving image-to-image translation tasks.

2. **Difference in the Discriminator of Pix2Pix compared to a regular GAN and their performances:** The Pix2Pix discriminator usually adopts a PatchGAN architecture. Unlike a standard discriminator that assigns a single "real/fake" label to the entire image, PatchGAN divides the image into smaller patches and determines realism locally. By focusing on local patches, the model is encouraged to produce detailed and realistic textures. This leads to more natural and higher-quality outputs. As we can see in the illustration below, the PatchGAN is dividing image to segments and then labeling each segment independently:

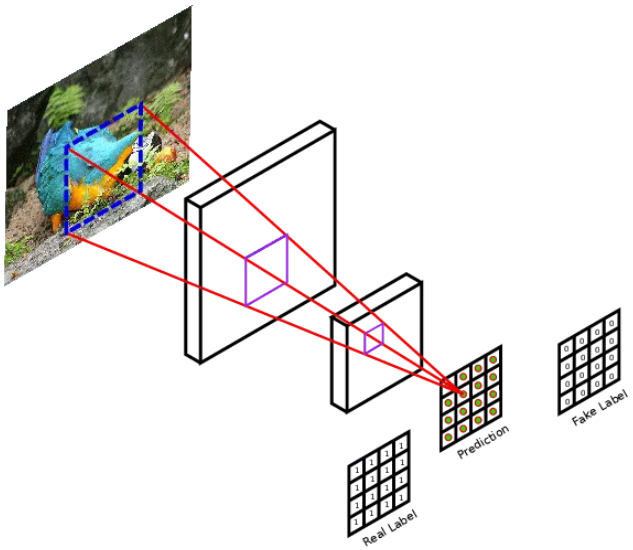

Figure 3: Illustration of PatchGAN.

3. **Use of U-Net architecture in the Generator, input/output dimensions, dropout:**

In the Pix2Pix model, the generator uses a U-Net architecture to effectively capture both high-level features and low-level details. The U-Net structure consists of an encoder (downsampling) path and a decoder (upsampling) path, connected by skip connections that transfer detailed spatial information from the encoder layers directly to the corresponding decoder layers. In figure 4, we can see detailed structure of the U-Net in pix2pix model.

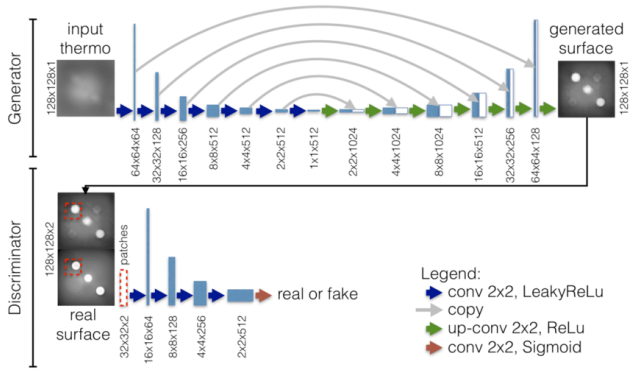

Figure 4: Structure of U-Net in pix2pix model with 128x128 features input.

- **Input/Output Dimensions:** Typically, Pix2Pix works with images of size 256×256 pixels. The original pix2pix article also presents the images in downsampled 256×256 pixels format. The encoder progressively downsamples the input image, halving the spatial resolution at each step, until it reaches a bottleneck layer (e.g., from 256×256 to 128×128, then 64×64, down to 1×1). Each downsampling step also increases the number of feature channels. For example:
  - Layer 1: $256 \times 256 \rightarrow 128 \times 128$ features
  - Layer 2: $128 \times 128 \rightarrow 64 \times 64$
  - ...
  - Final Bottleneck: 1×1

  The decoder then performs the inverse operation, upsampling back to 256×256. The skip connections between corresponding layers ensure that low-level detail lost during downsampling is recovered in the final output.

- **Number of Layers and Parameters:** The U-Net typically used in Pix2Pix has about 8 levels of down/upsampling. The total number of parameters depends on the number of filters at each layer, but this architecture is manageable on standard GPU hardware. The sequence of layers ensures a rich hierarchical representation, capturing global structure at the bottleneck and fine details through skip connections. In the figure 4, which is brought from a pix2pix implementation, has $12 + 1$ layers.

- **Dropout:** Dropout is used in the decoder part of the U-Net to prevent overfitting. The original Pix2Pix paper applies a dropout rate of 0.5 on the first three upsampling layers. This means 50% of the units in these layers are randomly dropped during training, forcing the network to learn more robust and generalizable features. As a result, the model can produce more consistent outputs even with smaller datasets.

4. **Training Cost, Overhead, and Implementation Details**
   Pix2Pix employs two main loss functions:

- **Adversarial Loss:** Uses a PatchGAN discriminator, which classifies local 70×70 patches instead of the whole image. By imposing realism constraints at the patch level, the generator is encouraged to produce fine-grained details. The PatchGAN discriminator is relatively small and computationally efficient.

- **L1 Loss:** In addition to the adversarial loss, Pix2Pix includes an L_1 loss to ensure the generated output closely matches the target image at a pixel level. The L_1 loss is weighted by $\lambda = 100$, providing a strong incentive for structural accuracy. This helps maintain image content and prevents mode collapse.

Computational Details and Training Overhead:

- **Training Setup:** Training is commonly done on a single modern GPU. For 256×256 images, training up to 200 epochs can take from several hours to a couple of days, depending on dataset size and hardware capabilities.
- **Batch Size and Complexity:** Pix2Pix often uses a batch size of 1 due to memory constraints and the model's complexity. This is typical for many image-to-image translation tasks.
- **Network Architecture Choices:** The use of a U-Net generator and a PatchGAN discriminator keeps the number of parameters and thus the computational overhead manageable. The U-Net's skip connections and the PatchGAN's local receptive fields strike a good balance between quality and efficiency.
- **Hyperparameters:** The model is typically trained using the Adam optimizer with a learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$. The simple L_1 loss is computationally inexpensive and easy to optimize, keeping overhead minimal.

Problem 5:

It is fully implemented and the questions are answered all the in the GM_HW3_Q5.ipynb file.

References

[1] Shun-ichi Amari and Hiroshi Nagaoka. Methods of information geometry. 191, 2007.

[2] Yuan Cao, Emmanuel Tannenbaum, Cheng Guo, Monica Bianchini, Marco Maggini, Gil Zuhovitsky, Mattia Rigotti, and Daniel Soudry. The curve of divergences: Score matching, neural adjoint learning, and beyond. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[3] Thomas M Cover and Joy A Thomas. Elements of information theory. 2006.

[4] Michael C. Fu. Chapter 19 gradient estimation. In Shane G. Henderson and Barry L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 575–616. Elsevier, 2006.

[5] Syed Ashar Javed. Learning to learn: Reinforce vs reparameterization trick. `https://stillbreeze.github.io/REINFORCE-vs-Reparameterization-trick/`, 2018.

[6] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[7] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.