Assignment 3: *Implicit Methods for Generation*

Date: *July 21, 2025* Due : *August 4, 2025*

Preface

This is the last series of assignments for the course *ECE1508: Deep Generative Models*. The exercises are aimed to review the implicit methods for generation studied in Chapters 4 to 6. Please adhere to **Code of Honor** outlined on the course page. Below, you can find the information about the contents, as well as instructions on how to submit them. Please read them *carefully*.

General Information The assignments are given in two sections. In the first section, you have written questions that can be answered in words or by derivation. The questions are consistent with the material of Chapter 1 and you do not need any further resources to answer them. The second section includes the programming assignments. In the case that a question is unclear or there is any flaws, please contact over Piazza. Also, in case that any particular assumption is required to solve the problem, feel free to consider the assumption and state it in your solution. The total mark of the assignments is **100 points** with total mark of written questions adds to **40 points** and the total mark of the programming assignments adds to **60 points**.

How to Submit A notebook has been provided with starter code that you can complete. The submission is carried out through the Crowdmark. Please submit the answer of each question **separately**, following the**steps below**. Please note that *failure to meet the formatting can lead to mark deduction*.

1. For Written Questions, you can submit handwritten or typed answers as a `.pdf`, `.jpg` or `.png` file.
2. For Programming Questions, the answer should **complete the Python Notebook** shared with this assignment. For *each question*, please print the corresponding part of the notebook as a `.pdf` file and upload it in the corresponding field on Crowdmark. Your uploaded `.pdf` should contain **all figures, diagrams and codes requested** in the question.
3. The completed Notebook, i.e., the `.ipynb` file, including all the codes and the outputs should also be submitted as the attachment to the last item on Crowdmark.

When submitting your notebook, please pay attention to the following points:

1. The file should be renamed as `<YourLastname>_<YourFirstname>_Assgn3.ipynb`
2. Please make sure to name the files with the name that is displayed on the Quercus account

The deadline for your submission is on **August 4, 2025 at 11:59 PM**.

- You can delay up to two days, i.e., until **August 6, 2025 at 11:59 PM**. After this extended deadline no submission is accepted.
- For each day of delay, 10% of the mark after grading is deducted.

Please submit your assignment **only through Crowdmark, and not by email.**

1 Written Exercises

Question 1 [15 Points] **(Dummy GAN)** Consider a dummy GAN which tries to learn distribution of *scalar* data $x \sim \mathcal{N}(\mu, 1)$ from the latent variable $z \sim \mathcal{N}(0, 1)$. To this end, it uses the following generator

$$G_w(z) = w + z.$$

In this question, we intend to see how vanilla GAN and W-GAN learn the data distribution in their training loop. Throughout the question, assume that we have collected significantly large number of data samples such that empirical and mathematical expectations lead to the same results, i.e.,

$$\mathbb{E}_{x\sim P(x)}\{\ldots\} = \hat{\mathbb{E}}_{x\sim P(x)}\{\ldots\} \qquad \mathbb{E}_{z\sim P(z)}\{\ldots\} = \hat{\mathbb{E}}_{z\sim P(z)}\{\ldots\}.$$

Therefore, you can always replace $\hat{\mathbb{E}}$ with the mathematical expectation.

Let us start with a vanilla GAN which uses the following discriminator

$$D_\phi(x) = \sigma(\phi x)$$

with $\sigma(\cdot)$ denoting the Sigmoid function. For this setting, answer the following items.

1. For this vanilla GAN, write the training objective function. You may present your final answer in terms of Gaussian expectations, i.e., $\mathbb{E}_{x\sim\mathcal{N}^0}\{f(x)\}$ for some function $f(\cdot)$.
2. Find the gradients of this objective with respect to the generator and discriminator parameters. Similar to Part 1, present your result in terms of Gaussian expectations.
3. Assume that w is set to its optimal choice, such that the generator can sample exactly from data distribution. Find the value of ϕ which maximizes the objective in this case. What do you conclude?
   Hint: *To find the optimal ϕ in this case, you may set the derivative of the objective with respect to ϕ to zero. You may further use the fact that $\sigma(0) = 0.5$.*

Next, we consider the Wasserstein GAN with the following discriminator

$$D_\phi(x) = \phi x$$

for $\phi \in [-1, 1]$. For this setting, answer the following items.

4. Determine the objective function for W-GAN training.
5. Compute the gradients of the objective function.
6. Solve the min-max training problem analytically, by first solving the inner loop maximization with respect to discriminator and then the outer loop minimization with respect to the generator. Explain your conclusion.
   Hint: *While solving the inner loop maximization, you should treat w as a general variable which can take any real value.*

Question 2 [10 Points] **(Learning Variance at Decoder)** A basic VAE aims to learn the data distribution $P(x)$. The encoder and decoder of this VAE are as follows:

- The encoder learns *only the mean value* via a neural network $\eta_{\mathbf{w}} : \mathbb{R}^d \mapsto \mathbb{R}^m$. It then samples the latent from a normal distribution with mean $\eta_{\mathbf{w}}(x)$ and unit variance, i.e., $Q_{\mathbf{w}}(z|x) \equiv \mathcal{N}(\eta_{\mathbf{w}}(x), \mathbf{I}_m)$, where $\mathbf{I}_m$ is the $m \times m$ identity matrix.
- The decoder learns *the mean, as well as a scalar variance* via a neural network $\mu_\theta, \sigma_\theta : \mathbb{R}^m \mapsto \mathbb{R}^d \times \mathbb{R}$. It then samples from a normal distribution with mean $\mu_\theta(z)$ and variance $\sigma_\theta^2(z)$, i.e., $P_\theta(x|z) \equiv \mathcal{N}\left(\mu_\theta(z), \sigma_\theta^2(z)\mathbf{I}_d\right)$. Note that $\sigma_\theta^2(z)$ is a scalar value, while $\mu_\theta(z) \in \mathbb{R}^d$.

We intend to understand the impacts that variance learning at the decoder can have on the training loop.

1. Find the evidence lower bound (ELBO) for a single data sample x in terms of the mean and variance of the encoder-decoder pair.
2. Explain how the ELBO is different as compared to the one determined in the lecture for a decoder with unite variance.
3. Assume that we are training this computational model by maximizing the ELBO. Explain how the recovery loss, i.e.,

   $$\|x - \mu_\theta(z)\|^2,$$

   impacts the gradient computation when $\sigma_\theta(z)$ becomes relatively large. What does this behavior suggest?
4. Propose a solution that prevents us from getting into such numerical instability.

QUESTION 3 [8 Points] **(Score Function)** Let $x \sim P(x)$ be a *continuous random variable* with $P(x)$ being its density function. We build y from x as

$$y = ax + b$$

for some known constants $a, b \in \mathbb{R}$.

1. Find the density of y in terms of x.
2. Let $s(x)$ denote the score function of x, i.e.,

   $$s(x) = \nabla_x \log P(x)$$

   Find the score function of y in terms of $s(x)$.
3. Explain how we can modify the Langevin dynamics that samples $x \sim P(x)$ to sample from y.

QUESTION 4 [7 Points] **(Codebook and Commitment Losses in VQ-VAE)** In the lectures, we discussed the idea of vector quantized (VQ)-VAE initilly proposed in [2]. In this assignment, we try to get a better understanding of its training loop which we later implement in the first programming question.

In the VQ-VAE proposal, the loss function consists of two components: (i) the reconstruction loss, which is computed via mean squared error (MSE), and (ii) the quantization loss which computes the difference between the quantized latent z and the output of the encoder before quantization u. The quantization loss is computed via the following sum

$$\mathcal{L}_{\text{VQ}}(u, z) = \|\text{sg}(u) - z\|^2 + \|\text{sg}(z) - u\|^2$$

where one of the terms in this sum is called *codebook loss* and the other is called *commitment loss*. The operator $\text{sg}(\cdot)$ if further a new operator defined in the original paper.

Go through the paper to find the answers to the following questions:

1. What is the operator $\text{sg}(\cdot)$ and what does it do?
2. Which term is the *codebook loss* and which one is the *commitment loss*? Explain why they are called so?
3. In what sense the codebook and commitment loss are different?
4. How can we implement these losses in PyTorch?

2 Programming Exercises

Question 1 [28 Points] **(Implementing VQ-VAE)** In this question, we implement a simple version of VQ-VAE architecture proposed in [2]. We train this model on the Fashion MNIST and sample from it by training a simple autoregressive model for latent distribution learning.

1. Complete Question 1 in `Lastname_Firstname_Asgn3.ipynb` to implement a vector quantized VAE.
2. Answer all parts marked by `# COMPLETE`.

Question 2 [32 Points] **(Diffusion Score Matching)** In this question, we implement a very basic diffusion model which invokes diffusion score matching (DSM) along with basic reverse SDE to learn MNIST image generation. You can learn more about DSM and reverse SDE in [1, 3], though what we had in the lecture is enough to complete this question.

1. Complete Question 2 in `Lastname_Firstname_Asgn3.ipynb`.
2. Answer all parts marked by `# COMPLETE`.

References

[1] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Available at `https://arxiv.org/abs/2101.09258`.

[2] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. Available at `https://arxiv.org/abs/1711.00937`.

[3] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.