Deep Learning:
Homework #02

Prof. Tongyao Pang

Deadline: December 7, 2025

Name: **Zhou Shouchen**
Student ID: 2025213446

Problem 1

1. 3D Reconstuction.
Describe the NeRF (Neural Radiance Fields) algorithm by answering the following three points clearly and concisely (3 points):

- What are the inputs and outputs of NeRF?
- What is the training loss? How to compute the gradient?
- What training data does NeRF require?

Solution
(a) In the original NeRF paper, a static 3D scene is represented by an implicit continuous function

$$F_\Theta : (x, y, z, \theta, \phi) \mapsto (\mathbf{c}, \sigma)$$

where Θ denotes the network parameters, (x, y, z) is the 3D coordinate of a point in space, and (θ, ϕ) are two angles that parameterize the viewing direction.
The corresponding output is a 4-dimensional vector (r, g, b, σ), where (r, g, b) is the RGB color $\mathbf{c}$ observed from direction (θ, ϕ) at the point (x, y, z), and $\sigma \geq 0$ is the volume density at that point, which controls how strongly the light is absorbed there.
(b) Consider a ray that originates from a camera:

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, \quad t \in [t_n, t_f]$$

where $\mathbf{o}$ is the camera center, $\mathbf{d}$ is the unit direction for a given pixel, which could be computed through (θ, ϕ) and $[t_n, t_f]$ is the near-to-far depth range. We sample N depths $t_1, \ldots, t_N$ within this interval, then we can get the 3D sample points $\mathbf{x}_i = \mathbf{r}(t_i) = \mathbf{o} + t_i\mathbf{d}$, where the spacing is $\delta_i = t_{i+1} - t_i$.
For each sample point $\mathbf{x}_i$, with the ray direction encoded as the two angles (θ, ϕ), run the network's forward process to get

$$(\mathbf{c}_i, \sigma_i) = F_\Theta(\mathbf{x}_i, \theta, \phi)$$

Then use the discrete volume rendering equation to calculate the predicted color of the corresponding pixel by letting the alpha value at the i-th segment $\alpha_i = 1 - \exp(-\sigma_i\delta_i)$ and the accumulated transmittance $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$. Thus predicted color of the pixel is

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i\alpha_i\mathbf{c}_i$$

Let $\mathbf{C}(\mathbf{r})$ be the ground-truth RGB color of the corresponding pixel in the training image. Then the loss for an image is taken to be the squared error

$$\mathcal{L}(\Theta) = \sum_{\mathbf{r}} \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2$$

The gradients are computed by the chain rule.
First, we differentiate the loss with respect to the predicted color $\mathbf{c}_i$:
$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{C}}(\mathbf{r})} = 2\left(\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\right)$. Since $\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i\alpha_i\mathbf{c}_i$, so we have for each sample color $\frac{\partial \hat{\mathbf{C}}(\mathbf{r})}{\partial \mathbf{c}_i} = T_i\alpha_i$, thus

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}_i} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{C}}(\mathbf{r})} \cdot \frac{\partial \hat{\mathbf{C}}(\mathbf{r})}{\partial \mathbf{c}_i} = 2\left(\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\right) \cdot T_i\alpha_i$$

For the density σ_i, it affects both α_i and all T_k with $k > i$:

$$\alpha_k = 1 - \exp(-\sigma_k \delta_k) \quad \Rightarrow \quad \frac{\partial \alpha_i}{\partial \sigma_i} = \delta_i \exp(-\sigma_i \delta_i) = \delta_i (1 - \alpha_i)$$

For the transmittance, when $k > i$:

$$T_k = \exp\left(- \sum_{j<k} \sigma_j \delta_j \right) \quad \Rightarrow \quad \frac{\partial T_k}{\partial \sigma_i} = -\delta_i T_k$$

and when $k \leq i$, σ_i does not appear in the sum, so $\dfrac{\partial T_k}{\partial \sigma_i} = 0$. Thus

$$\frac{\partial \hat{\mathbf{C}}(\mathbf{r})}{\partial \sigma_i} = T_i \frac{\partial \alpha_i}{\partial \sigma_i} \mathbf{c}_i + \sum_{k=i+1}^{N} \frac{\partial T_k}{\partial \sigma_i} \alpha_k \mathbf{c}_k = T_i \delta_i (1 - \alpha_i) \mathbf{c}_i - \delta_i \sum_{k=i+1}^{N} T_k \alpha_k \mathbf{c}_k$$

$$\Rightarrow \qquad \frac{\partial \mathcal{L}}{\partial \sigma_i} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{C}}(\mathbf{r})} \cdot \frac{\partial \hat{\mathbf{C}}(\mathbf{r})}{\partial \sigma_i} = 2 \left(\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right) \cdot \left(T_i \delta_i (1 - \alpha_i) \mathbf{c}_i - \delta_i \sum_{k=i+1}^{N} T_k \alpha_k \mathbf{c}_k \right)$$

Finally, each pair $(\mathbf{c}_i, \sigma_i)$ is an output of the MLP F_Θ given the 5D input. Thus we can continue to apply the chain rule layer by layer inside the network, propagating $\dfrac{\partial \mathcal{L}}{\partial \mathbf{c}_i}$ and $\dfrac{\partial \mathcal{L}}{\partial \sigma_i}$ back to all parameters, obtaining $\nabla_\Theta \mathcal{L}$.

(c) The original NeRF requires multi-view RGB images of a single static 3D scene together with the corresponding camera parameters, i.e. the camera's intrinsics and extrinsics.

Problem 2

2. Instantaneous Change of Variables.
Derive the Instantaneous Change of Variables formula:

$$\frac{\mathrm{d}}{\mathrm{d}t}\log p(\mathbf{z}(t)) = -\operatorname{tr}\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{z}(t)}\right),$$

where the dynamics are given by

$$\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = f(\mathbf{z}(t), t),$$

and the function f is assumed to be uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t. Provide at least **two ways** to prove it. (4 points)
Hints:

- Hint 1: Consider the time limit of the discrete change-of-variables formula.
- Hint 2: You may utilize the Fokker-Planck equation in the deterministic case.
- Hint 3: You may introduce a smooth test function to rigorously justify the derivation

Solution
<1>. Method 1: Consider the time limit of the discrete change-of-variables formula:
For a fixed time t, consider a sufficiently small time step $\epsilon > 0, \epsilon \to 0$. Then we can use the Taylor expansion of $\mathbf{z}(t)$ at time t:

$$\mathbf{z}(t+\epsilon) = \mathbf{z}(t) + \epsilon\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} + O(\epsilon^2) = \mathbf{z}(t) + \epsilon f(\mathbf{z}(t), t) + O(\epsilon^2)$$

i.e. for each $\mathbf{z} \in \mathbb{R}^d$ we can define the mapping

$$g_\epsilon(\mathbf{z}) = \mathbf{z} + \epsilon f(\mathbf{z}, t) + O(\epsilon^2)$$

Let $p_t(\mathbf{z})$ and $p_{t+\epsilon}(\mathbf{z})$ be the densities of $\mathbf{z}(t)$ and $\mathbf{z}(t+\epsilon)$, respectively. During the time interval of length ϵ, the random variable $\mathbf{z}(t)$ is approximately mapped into $\mathbf{z}(t+\epsilon)$ by g_ϵ. For any $\mathbf{z}' \in \mathbb{R}^d$, we can define that $\mathbf{z} = g_\epsilon^{-1}(\mathbf{z}')$, i.e. $g_\epsilon(\mathbf{z}) = \mathbf{z}'$. Since g_ϵ is invertible and differentiable, the discrete change of variables formula is

$$p_{t+\epsilon}(\mathbf{z}') = p_t(\mathbf{z})\left|\det\frac{\partial\mathbf{z}}{\partial\mathbf{z}'}\right|$$

where $\frac{\partial\mathbf{z}}{\partial\mathbf{z}'}$ is the Jacobian matrix of the inverse mapping g_ϵ^{-1} at the point $\mathbf{z}'$. From the definition of g_ϵ and $\mathbf{z}'$:

$$g_\epsilon(\mathbf{z}) = \mathbf{z} + \epsilon f(\mathbf{z}, t) + O(\epsilon^2), \quad \mathbf{z} = g_\epsilon^{-1}(\mathbf{z}') \Leftrightarrow g_\epsilon(\mathbf{z}) = \mathbf{z}'$$

we can get that

$$\frac{\partial\mathbf{z}'}{\partial\mathbf{z}} = \frac{\partial g_\epsilon(\mathbf{z})}{\partial\mathbf{z}} = I + \epsilon\frac{\partial f}{\partial\mathbf{z}}(\mathbf{z}, t) + O(\epsilon^2)$$

where I is the $d \times d$ identity matrix. As $\epsilon \to 0$, we could say that the Jacobian matrix is a small perturbation of I, which means the Jacobian matrix is invertible. Combined with the condition that f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t, we can say that Inverse Function Theorem holds.
From the theorem in matrix, we can get that

$$\partial(\det X) = \det(X)\operatorname{tr}(X^{-1}\partial X)$$

Thus we can use Taylor expansion of the determinant $\det\left(I + \epsilon\frac{\partial f}{\partial\mathbf{z}}(\mathbf{z}, t)\right)$ at I:

$$\det\left(I + \epsilon\frac{\partial f}{\partial\mathbf{z}}(\mathbf{z}, t)\right) = \det(I) + \operatorname{tr}\left(I^{-1}\epsilon\frac{\partial f}{\partial\mathbf{z}}\right)\cdot\det(I) + O\left(\left\|\epsilon\frac{\partial f}{\partial\mathbf{z}}\right\|^2\right) = 1 + \epsilon\operatorname{tr}\frac{\partial f}{\partial\mathbf{z}}(\mathbf{z}, t) + O(\epsilon^2)$$

Problem 2 continued on next page...

Thus using $(1+x)^{-1} = 1 - x + O(x^2)$, we can get that

$$\left|\det \frac{\partial \mathbf{z}}{\partial \mathbf{z}'}\right| = \det\left(I + \epsilon \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}, t) + O(\epsilon^2)\right)^{-1} = 1 - \epsilon \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}, t) + O(\epsilon^2)$$

Again, from the Taylor expansion of g_ϵ mentioned above, and $\epsilon \to 0$, we have

$$\mathbf{z}' = g_\epsilon(\mathbf{z}) = \mathbf{z} + \epsilon f(\mathbf{z}, t) + O(\epsilon^2), \quad \mathbf{z} = g_\epsilon^{-1}(\mathbf{z}') = \mathbf{z}' - \epsilon f(\mathbf{z}', t) + O(\epsilon^2)$$

From the smoothness on p_t and $\frac{\partial f}{\partial \mathbf{z}}$, apply Taylor expansion to them at $\mathbf{z}'$, we can get that

$$p_t(\mathbf{z}) = p_t(\mathbf{z}' - \epsilon f(\mathbf{z}', t) + O(\epsilon^2)) = p_t(\mathbf{z}') - \epsilon f(\mathbf{z}', t) \cdot \nabla p_t(\mathbf{z}') + O(\epsilon^2), \qquad \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}, t) = \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}', t) + O(\epsilon)$$

Put these expansions altogether into the change of variables formula, we can get that

$$\begin{aligned}
p_{t+\epsilon}(\mathbf{z}') &= p_t(\mathbf{z}) \left|\det \frac{\partial \mathbf{z}}{\partial \mathbf{z}'}\right| \\
&= p_t(\mathbf{z}) \left(1 - \epsilon \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}, t) + O(\epsilon^2)\right) \\
&= \left(p_t(\mathbf{z}') - \epsilon f(\mathbf{z}', t) \cdot \nabla p_t(\mathbf{z}') + O(\epsilon^2)\right) \left(1 - \epsilon \left(\operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}', t) + O(\epsilon)\right) + O(\epsilon^2)\right) \\
&= p_t(\mathbf{z}') - \epsilon f(\mathbf{z}', t) \cdot \nabla p_t(\mathbf{z}') - \epsilon p_t(\mathbf{z}') \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}', t) + O(\epsilon^2)
\end{aligned}$$

Divide ϵ to the both side, and setting $\epsilon \to 0$, we get

$$\begin{aligned}
\lim_{\epsilon \to 0} \frac{p_{t+\epsilon}(\mathbf{z}') - p_t(\mathbf{z}')}{\epsilon} &= \lim_{\epsilon \to 0} \left(-f(\mathbf{z}', t) \cdot \nabla p_t(\mathbf{z}') - p_t(\mathbf{z}') \operatorname{tr} \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}', t) + O(\epsilon)\right) \\
\Rightarrow \quad \partial_t p_t(\mathbf{z}) &= -f(\mathbf{z}, t) \cdot \nabla p_t(\mathbf{z}) - p_t(\mathbf{z}) \operatorname{tr} \left(\frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}, t)\right)
\end{aligned}$$

Then apply the chain rule, we can get that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(\mathbf{z}(t)) &= \partial_t \log p_t(\mathbf{z}(t)) + \frac{\partial \left(\log p_t(\mathbf{z}(t))\right)}{\partial \mathbf{z}(t)} \cdot \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} \\
&= \frac{\partial_t p_t(\mathbf{z}(t))}{p_t(\mathbf{z}(t))} + \frac{1}{p_t(\mathbf{z}(t))} \cdot \frac{\partial p_t(\mathbf{z}(t))}{\partial \mathbf{z}(t)} \cdot f(\mathbf{z}(t), t) \\
&= -\frac{1}{p_t(\mathbf{z}(t))} \cdot f(\mathbf{z}(t), t) \cdot \nabla p_t(\mathbf{z}(t)) - \operatorname{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}(\mathbf{z}(t), t)\right) + \frac{1}{p_t(\mathbf{z}(t))} \cdot \nabla p_t(\mathbf{z}(t)) \cdot f(\mathbf{z}(t), t) \\
&= -\operatorname{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}(\mathbf{z}(t), t)\right)
\end{aligned}$$

So above all, we have proved that

$$\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(\mathbf{z}(t)) = -\operatorname{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}(\mathbf{z}(t), t)\right)$$

<2>. Method 2: Using the smooth test function.
To show that the density $p_t(\mathbf{z})$ satisfies

$$\partial_t p_t(\mathbf{z}) + \nabla_{\mathbf{z}} \cdot (p_t(\mathbf{z}) f(\mathbf{z}, t)) = 0$$

we can take a smooth test function $\phi(\mathbf{z})$: assume that $\phi(\mathbf{z}) p_t(\mathbf{z}) f(\mathbf{z}, t) \to 0$ as $\|\mathbf{z}\| \to \infty$.
Then we have

$$\int_{\mathbb{R}^d} \phi(\mathbf{z}) p_t(\mathbf{z}) \mathrm{d}\mathbf{z} = \mathbb{E}_{p_t}[\phi(\mathbf{z})]$$

Differentiating both sides with respect to t and using the chain rule together with $\dot{\mathbf{z}}(t) = f(\mathbf{z}(t), t)$, we can get that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d}\phi(\mathbf{z})p_t(\mathbf{z})\mathrm{d}\mathbf{z} &= \int_{\mathbb{R}^d}\phi(\mathbf{z})\partial_t p_t(\mathbf{z})\mathrm{d}\mathbf{z} \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_{p_t}[\phi(\mathbf{z})] = \mathbb{E}_{p_t}[\nabla_{\mathbf{z}}\phi(\mathbf{z})\cdot f(\mathbf{z},t)] &= \int \nabla_{\mathbf{z}}\phi(\mathbf{z})\cdot f(\mathbf{z},t)p_t(\mathbf{z})\mathrm{d}\mathbf{z} \\
\Rightarrow \quad \int \phi(\mathbf{z})\partial_t p_t(\mathbf{z})\mathrm{d}\mathbf{z} &= \int \nabla_{\mathbf{z}}\phi(\mathbf{z})\cdot f(\mathbf{z},t)p_t(\mathbf{z})\mathrm{d}\mathbf{z}
\end{aligned}$$

Applying integration by parts(the divergence theorem) to the right-hand side and using the assumption mentioned at the beginning of the method: $\phi(\mathbf{z})p_t(\mathbf{z})f(\mathbf{z},t) \to 0$ as $\|\mathbf{z}\| \to \infty$. We can get that

$$\int_{\mathbb{R}^d}\nabla_{\mathbf{z}}\phi(\mathbf{z})\cdot f(\mathbf{z},t)p_t(\mathbf{z})\mathrm{d}\mathbf{z} = -\int_{\mathbb{R}^d}\phi(\mathbf{z})\nabla_{\mathbf{z}}\cdot\left(p_t(\mathbf{z})f(\mathbf{z},t)\right)\mathrm{d}\mathbf{z}$$

Thus we have for all smooth test function ϕ with $\phi(\mathbf{z})p_t(\mathbf{z})f(\mathbf{z},t) \to 0$ as $\|\mathbf{z}\| \to \infty$:

$$\int \phi(\mathbf{z})\partial_t p_t(\mathbf{z})\mathrm{d}\mathbf{z} = -\int_{\mathbb{R}^d}\phi(\mathbf{z})\nabla_{\mathbf{z}}\cdot\left(p_t(\mathbf{z})f(\mathbf{z},t)\right)\mathrm{d}\mathbf{z} \quad \Rightarrow \quad \int_{\mathbb{R}^d}\phi(\mathbf{z})\left(\partial_t p_t(\mathbf{z}) + \nabla_{\mathbf{z}}\cdot\left(p_t(\mathbf{z})f(\mathbf{z},t)\right)\right)\mathrm{d}\mathbf{z} = 0$$

Thus we have almost everywhere in $\mathbf{z}$:

$$\partial_t p_t(\mathbf{z}) + \nabla_{\mathbf{z}}\cdot\left(p_t(\mathbf{z})f(\mathbf{z},t)\right) = 0$$

Then expand the divergence term, where $\nabla_{\mathbf{z}}\cdot f(\mathbf{z},t) = \sum_{i=1}^{d}\frac{\partial f_i}{\partial z_i}(\mathbf{z},t)$:

$$\nabla_{\mathbf{z}}\cdot\left(p_t(\mathbf{z})f(\mathbf{z},t)\right) = \nabla_{\mathbf{z}}p_t(\mathbf{z})\cdot f(\mathbf{z},t) + p_t(\mathbf{z})\nabla_{\mathbf{z}}\cdot f(\mathbf{z},t)$$

Put it back into the original equation:

$$\partial_t p_t(\mathbf{z}) + f(\mathbf{z},t)\cdot\nabla_{\mathbf{z}}p_t(\mathbf{z}) + p_t(\mathbf{z})\nabla_{\mathbf{z}}\cdot f(\mathbf{z},t) = 0$$

Using the chain rule, $\dot{\mathbf{z}}(t) = f(\mathbf{z}(t), t)$, and the equation above, we can get that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}p_t(\mathbf{z}(t)) &= \partial_t p_t(\mathbf{z}(t)) + \nabla_{\mathbf{z}}p_t(\mathbf{z}(t))\cdot\dot{\mathbf{z}}(t) \\
&= \partial_t p_t(\mathbf{z}(t)) + f(\mathbf{z}(t),t)\cdot\nabla_{\mathbf{z}}p_t(\mathbf{z}(t)) \\
&= -p_t(\mathbf{z}(t))\nabla_{\mathbf{z}}\cdot f(\mathbf{z}(t),t) \\
\Rightarrow \quad \frac{\frac{\mathrm{d}}{\mathrm{d}t}p_t(\mathbf{z}(t))}{p_t(\mathbf{z}(t))} &= -\nabla_{\mathbf{z}}\cdot f(\mathbf{z}(t),t) \\
\Rightarrow \quad \frac{\mathrm{d}}{\mathrm{d}t}\log p_t(\mathbf{z}(t)) &= -\nabla_{\mathbf{z}}\cdot f(\mathbf{z}(t),t)
\end{aligned}$$

Finally, using the relation between divergence and the trace of the Jacobian,

$$\nabla_{\mathbf{z}}\cdot f(\mathbf{z}(t),t) = \sum_{i=1}^{d}\frac{\partial f_i}{\partial z_i}(\mathbf{z}(t),t) = \sum_{i=1}^{d}\left(\frac{\partial f}{\partial z}(\mathbf{z}(t),t)\right)_{ii} = \mathrm{tr}\left(\frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}(t),t)\right)$$

So above all, we have proved that

$$\frac{\mathrm{d}}{\mathrm{d}t}\log p_t(\mathbf{z}(t)) = -\,\mathrm{tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}(\mathbf{z}(t),t)\right)$$

Problem 3

3. Large Language Models.
LLMs model a joint distribution over a sequence $x_1, \ldots, x_T$ by factorizing it into conditional distributions

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_{<t}),$$

and learning these conditionals through next-token prediction.
Answer the following:

- How is next-token prediction implemented in practice using a Transformer decoder? (1 pts)
- Identify the training setup:including the input,output and traning objective. (2 pts)
- What is the difference between sampling during training and inference? Why? (2 pts)

Solution
(a) In a Transformer decoder, next-token prediction proceeds as: given tokens $x_1, \ldots, x_T$, embed them and add positional encodings to get the embedding:

$$e_t = \text{Embed}(x_t) + \text{PosEnc}(t)$$

These are processed by masked self-attention, where the causal mask enforces t attends only to $\{1, \ldots, t\}$. So the decoder produces hidden states

$$h_t = \text{DecoderLayer}(e_{\leq t})$$

and the next-token distribution is calculated through a shared linear layer and then softmax:

$$p_\theta(x_t \mid x_{<t}) = \text{softmax}(W h_t + b)$$

Where h_t is the hidden state for the decoder, depend only on $x_{<t}$ through causal mask. During inference, this distribution is used to choose or sample the next token.
(b) The model receives an input sequence that is shifted right by one position (e.g., $(x_0, x_1, \ldots, x_{T-1})$), where x_0 may contain a BOS or a padding token depending on implementation. And the target output is $(x_1, \ldots, x_T)$.
At each time step the model outputs $p_\theta(\cdot \mid x_{<t})$ and compares it to target token x_t. This uses teacher forcing, that is, the model is trained to predict the next token given the previous ones, instead of the truly predicted token. This makes the model more stable and could train parallelly.
The objective is to maximize log-likelihood

$$\max_\theta \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}) \quad \Leftrightarrow \quad \mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t})$$

(c) During training, the model does not sample tokens; it always conditions on the ground-truth previous tokens (teacher forcing), which provides the true prefix $x_{<t}$, and the model's distribution is only used to compute

$$-\log p_\theta(x_t \mid x_{<t})$$

During inference, the true next token is unavailable, so the model must choose or sample from the network's forward distribution:

$$x_t \sim p_\theta(\cdot \mid x_{<t}) \quad \text{or} \quad x_t = \text{argmax}\, p_\theta(\cdot \mid x_{<t})$$

and the generated token becomes the next input.
The key difference is that training requires differentiable likelihood optimization, while inference must convert probability distributions into actual tokens.

Problem 4

4. Coding: VAE on MNIST.
A Python notebook with the code to be completed is provided. Please complete it using the following intructions. This problem is adapted from the pset1 of Course 6.S978 Deep Generative Models given by Professor Kaiming He at MIT.
VAEs are trained by maximizing the Evidence Lower Bound (ELBO) on the marginal log-likelihood:

$$\log p(x) \geq \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] = \text{ELBO},$$

where x is the data (binary images for MNIST) and z is the latent code.
(a) Give a detailed mathematical proof of the ELBO starting from the marginal log-likelihood $\log p(x)$. (2 Points)
(b) Complete the implementation of the "self.encoder" and "self.decoder" in the "VAE()" model. (2 Points)
(c) Implement the reparameterization trick in the "reparameterize()" function. In this assignment, we only sample one latent code z_i for each x_i. (1 Points)
(d) In practice, the above expectation in ELBO is estimated using Monte Carlo sampling, yielding the generic Stoachastic Gradient Variational Bayes (SGVB) estimator,

$$\text{ELBO} \approx \sum_i \left[\log p(x_i|z_i) + \log p(z_i) - \log q(z_i|x_i)\right],$$

where z_i is sampled from $q(z|x_i) = \mathcal{N}(z; \mu_i, \sigma_i^2\mathbf{I})$.
Finalize the SGVB estimator by completing the "log_normal_pdf()" function, which computes the log probability for a normal distribution given its mean and variance. (1 Points)
(e) In many cases, Monte Carlo sampling is not necessary to estimate all the terms of ELBO, as some terms can be integrated analytically. In particular, when both $q(z|x) = \mathcal{N}(z; \mu(x), \text{diag}(\sigma^2(x)))$ and $p(z) = \mathcal{N}(z; 0, I)$ are Gaussian distributions, the ELBO can be decomposed into an analytical KL divergence plus the expected reconstruction error:

$$\text{ELBO} \approx -D_{KL}(q(z|x)\|p(z)) + \sum_i \log p(x_i|z_i) = \frac{1}{2}\sum_d (1 + \log((\sigma_d)^2) - (\mu_d)^2 - (\sigma_d)^2) + \sum_i \log p(x_i|z_i)$$

where d is the dimension of the latent space, and i is the indices of the data.
Run the verfirication code to check if the analytical KL divergence matches the Monte Carlo estimate. (2 Points)
(f) Using the above two losses, train two VAE models on the MNIST dataset (manual tuning of parameters such as epochs, hidden dims, lr, coeff may be necessary). Use the provided evaluation code to visualize the reconstruction results and the generated images (in 2D grid) for both models. (3 Points)
(g) Latent Interpolation: Encode two MNIST test images with different digit labels to obtain latent codes z_1 and z_2. Linearly interpolate between them using $z(\alpha) = (1-\alpha)z_1 + \alpha z_2$ for $\alpha \in \{0, 0.1, \ldots, 1\}$, decode each interpolated code with your trained VAE decoder, and display the generated images in order. Briefly describe how the generated digits gradually transform from one class to the other along the interpolation. (2 Points)
Solution
For the marginal log-likelihood of $\log p(x)$, we can introduce an arbitrary conditional distribution $q(z|x)$ over the latent variable z, then we can get that

$$\log p(x) = \log p(x) \underbrace{\int q(z|x)\mathrm{d}z}_{1} = \int q(z|x)\log p(x)\mathrm{d}z = \mathbb{E}_{q(z|x)}\left[\log p(x)\right]$$

Then according to Bayes' rule, we can get that

$$p(x,z) = p(z|x)p(x)$$

Thus we have

$$\begin{aligned}
\log p(x) &= \mathbb{E}_{q(z|x)}\left[\log p(x)\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{p(z|x)}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log \left(\frac{p(x,z)}{q(z|x)} \cdot \frac{q(z|x)}{p(z|x)}\right)\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)} + \log \frac{q(z|x)}{p(z|x)}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right]}_{ELBO} + KL(q(z|x), p(z|x))
\end{aligned}$$

According to the fact that $\log(\cdot)$ is a concave function. From Jensen's inequality, we have for any nonnegative random variable Y: $\mathbb{E}[\log(Y)] \leq \log[E(Y)]$, and setting $Y = \frac{\nu(x)}{\mu(x)}$, we can get that the KL divergence is nonnegative:

$$-KL(\mu\|\nu) = -\mathbb{E}_\mu\left[\log \frac{\mu(x)}{\nu(x)}\right] = \mathbb{E}_\mu\left[\log \frac{\nu(x)}{\mu(x)}\right] \leq \log \mathbb{E}_\mu\left[\frac{\nu(x)}{\mu(x)}\right] = \log\left(\int \mu(x) \cdot \frac{\nu(x)}{\mu(x)}\right) = \log\left(\int \nu(x)\right) = 0$$

i.e. $KL(\mu\|\nu) \geq 0$ with equality if and only if $\mu = \nu$. So above all, we have proved that

$$\log p(x) \geq \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] = \text{ELBO}$$

With equality if and only if $q(z|x) = p(z|x)$.
(b), (c), (d), (e), (f), (g)'s implementation code are in `hw2_coding.ipynb`.
(e) As shown in the figure 1, as the number of samples increases, the Monte-Carlo estimation SGVB quickly converges to the analytic KL value. The two curves gradually get almost overlap, with only minor stochastic fluctuations. This demonstrates that our derivation and implementation of the analytic KL are correct.

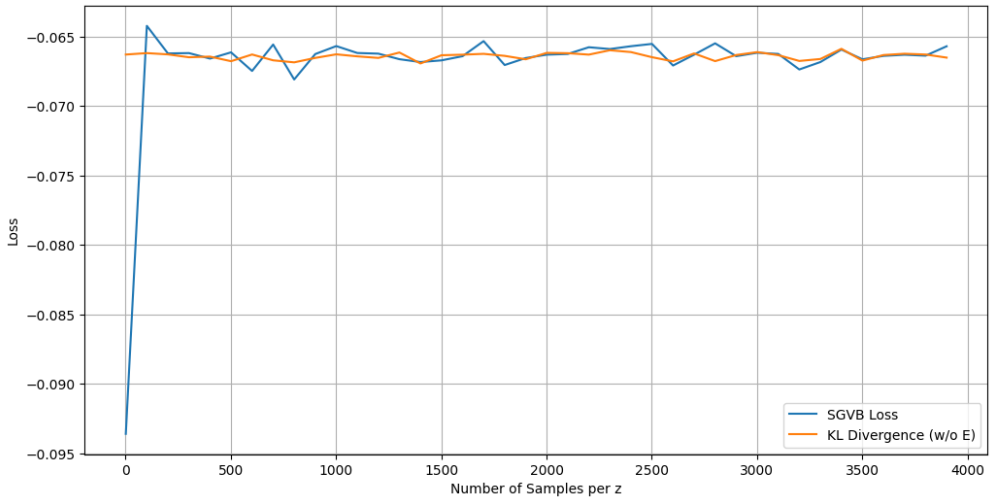

Figure 1: Loss comparison between SGVB and analytic KL.

(f) As shown in the figure 2, the left column shows reconstructions obtained using the SGVB loss, the right column shows those from the model trained with the analytic KL loss. Both models capture the global digit structure, and could reconstruct the input images.

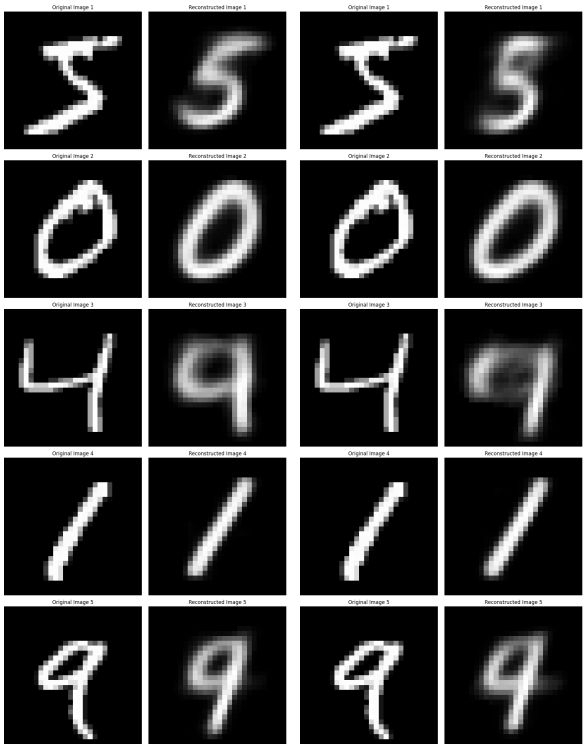

Figure 2: Reconstruction comparison between models trained with SGVB loss (left) and analytic KL loss (right).

(g) As shown in the figure 3, starting from the digit represented by z_1, the generated images gradually morph as α increases, with intermediate samples blending structural features from both digits. Early images retain the shape of the first digit, mid-range images show mixed or ambiguous patterns, and the final images fully adopt the appearance of the digit represented by z_2. This progression demonstrates that the learned latent space is well-structured and supports semantic interpolation between different digit classes.

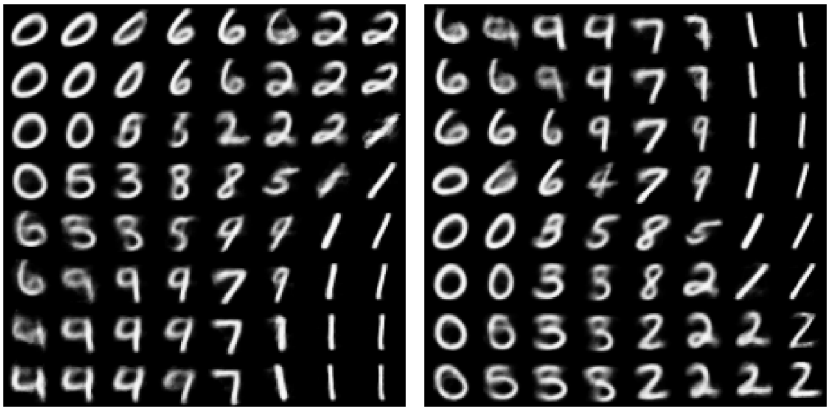

Figure 3: Generated samples from VAEs trained with SGVB loss (left) and analytic KL loss (right)