ZDC Simulation with ML

Wen-Chen Chang

2024/8/5

* Date: August 5 (Monday), 2024

* Time: 12:00-13:00 PM at Taiwan (GMT+8)

* Zoom link:

https://cern.zoom.us/j/66342263280?pwd=DBemHUOnO6QIiQyU5y2WbeaEaBGcyT.1

arXiv:2407.16704

- “Applying generative neural networks for fast simulations of the ALICE (CERN) experiment”
  - Generative neural networks
  - Fast simulations
  - Analysis
  - Summary

Generative neural networks

- Variational autoencoder (VAE)
- Generative adversarial network (GAN)
- Autoregressive models
- Normalizing flows (NFs)
- Vector-Quantized Variational Autoencoder (VQ-VAE)
- Diffusion models

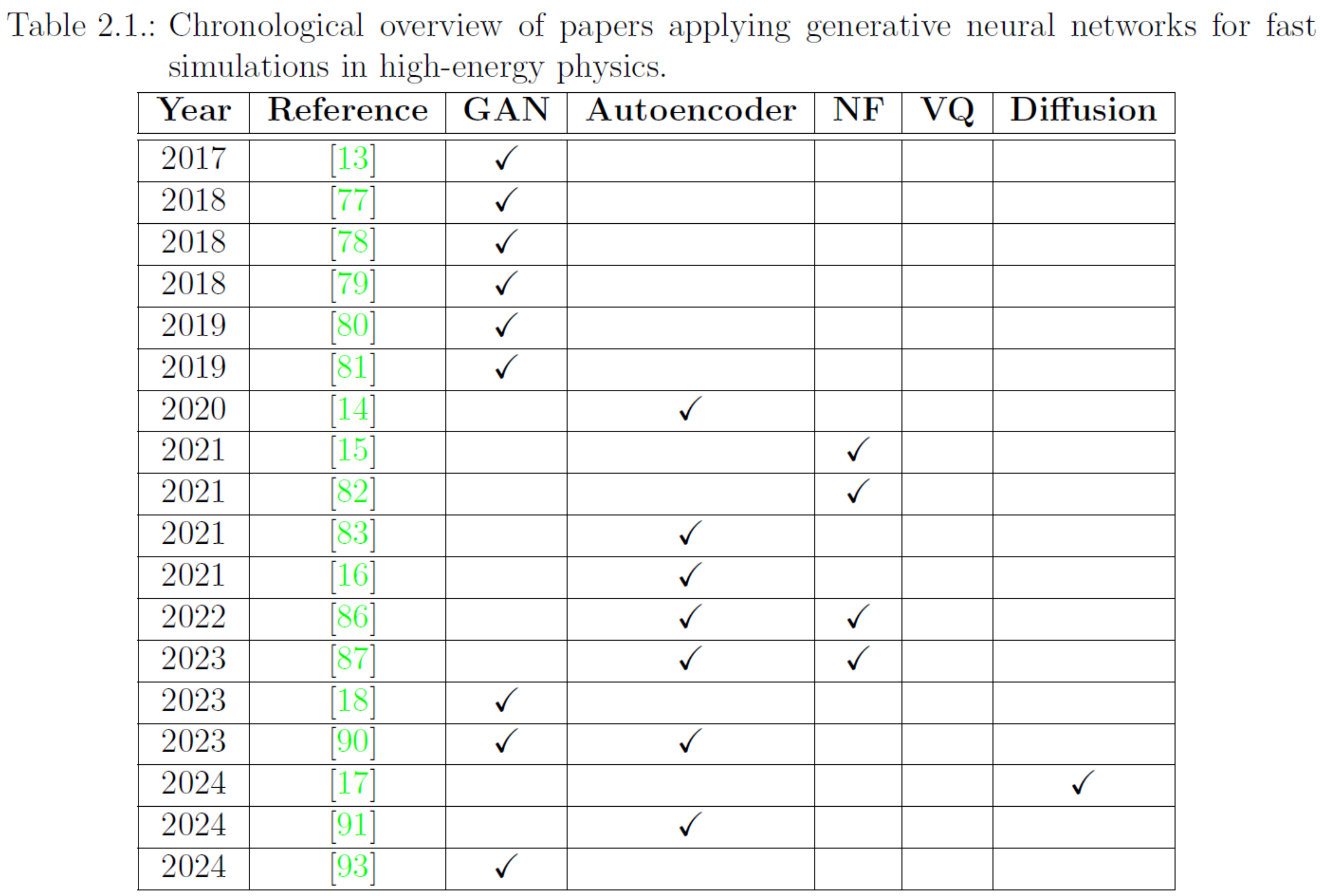


Table 2.1.: Chronological overview of papers applying generative neural networks for fast simulations in high-energy physics.

Year	**Reference**	**GAN**	**Autoencoder**	**NF**	**VQ**	**Diffusion**
2017	[13]	✓				
2018	[77]	✓				
2018	[78]	✓				
2018	[79]	✓				
2019	[80]	✓				
2019	[81]	✓				
2020	[14]		✓			
2021	[15]			✓		
2021	[82]			✓		
2021	[83]		✓			
2021	[16]		✓			
2022	[86]		✓	✓		
2023	[87]		✓	✓		
2023	[18]	✓				
2023	[90]	✓	✓			
2024	[17]					✓
2024	[91]		✓			
2024	[93]	✓				

VAE from ChatGPT

Variational Autoencoders (VAEs) 是一種生成模型，可以對數據進行壓縮和重建，同時還能生成新數據。這裡是對 VAE 的詳細解釋：

主要概念

1. **編碼器 (Encoder)**：

  - 編碼器將輸入數據 \(x\) 映射到潛在空間 \(z\) 的分佈上。

  - 與傳統自編碼器不同，VAE 將輸入編碼為概率分佈的參數（通常是高斯分佈），即輸出均值 \(\mu\) 和標準差 \(\sigma\)。

2. **潛在空間 (Latent Space)**：

  - 潛在空間是一個低維空間，代表壓縮後的輸入數據。

  - 從這個空間中采樣可以生成新的數據點。

  - 目的是確保潛在空間連續且結構良好。

3. **解碼器 (Decoder)**：

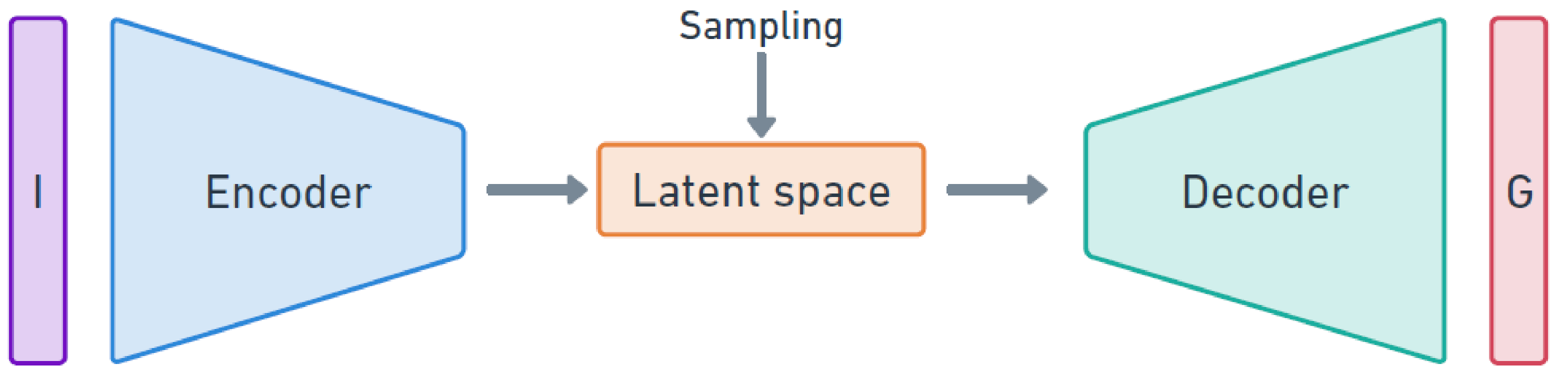
  - 解碼器將潛在空間中的樣本映射回原始數據空間。

  - 確保從潛在分佈中抽樣的數據點與原始數據分佈相似。

VAE from ChatGPT

核心組件
- **重參數化技巧 (Reparameterization Trick)**：
  - 為了使模型可微分並啟用反向傳播，使用了重參數化技巧。
  - 這樣做的目的是將樣本從標準正態分佈中抽取輔助變量 \(\epsilon\)，並計算 \(z = \mu + \sigma \cdot \epsilon\)。
  - 這允許梯度通過 \(\mu\) 和 \(\sigma\) 傳播。
- **損失函數 (Loss Function)**：
  - VAE 的損失函數包含兩個主要部分：
    - **重建損失 (Reconstruction Loss)**：測量解碼器重建輸入數據的效果。通常使用均方誤差 (MSE) 或二進制交叉熵來計算原始數據與重建數據之間的差異。
    - **KL 散度 (KL Divergence)**：測量編碼分佈 \(q(z|x)\) 與先驗分佈 \(p(z)\)（通常是標準正態分佈）之間的差異。正則化潛在空間，確保其遵循先驗分佈，促進潛在空間的平滑和連續性。

VAE



To construct a generative model, VAE incorporates variational inference of a posterior distribution of a latent variable [19]. The main objective of VAE is to represent the probability distribution $p(x)$ of the data x by estimating a latent variable model $p(x|z)$, where z is the latent variable. Since the true posterior $p(z|x)$ is usually intractable, VAE approximates it with a tractable distribution $q(z|x)$ determined by the output of the encoder. Typically, a

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\theta(z|x)}[\log p_\phi(x|z)] - D_{KL}(q_\theta(z|x)||p(z)), \tag{3.1}$$

where θ and ϕ represent the parameters of the encoder E and decoder D, respectively, D_{KL} denotes the KL divergence, and $p(z)$ stands for the prior distribution over the latent variables.

VAE with noise

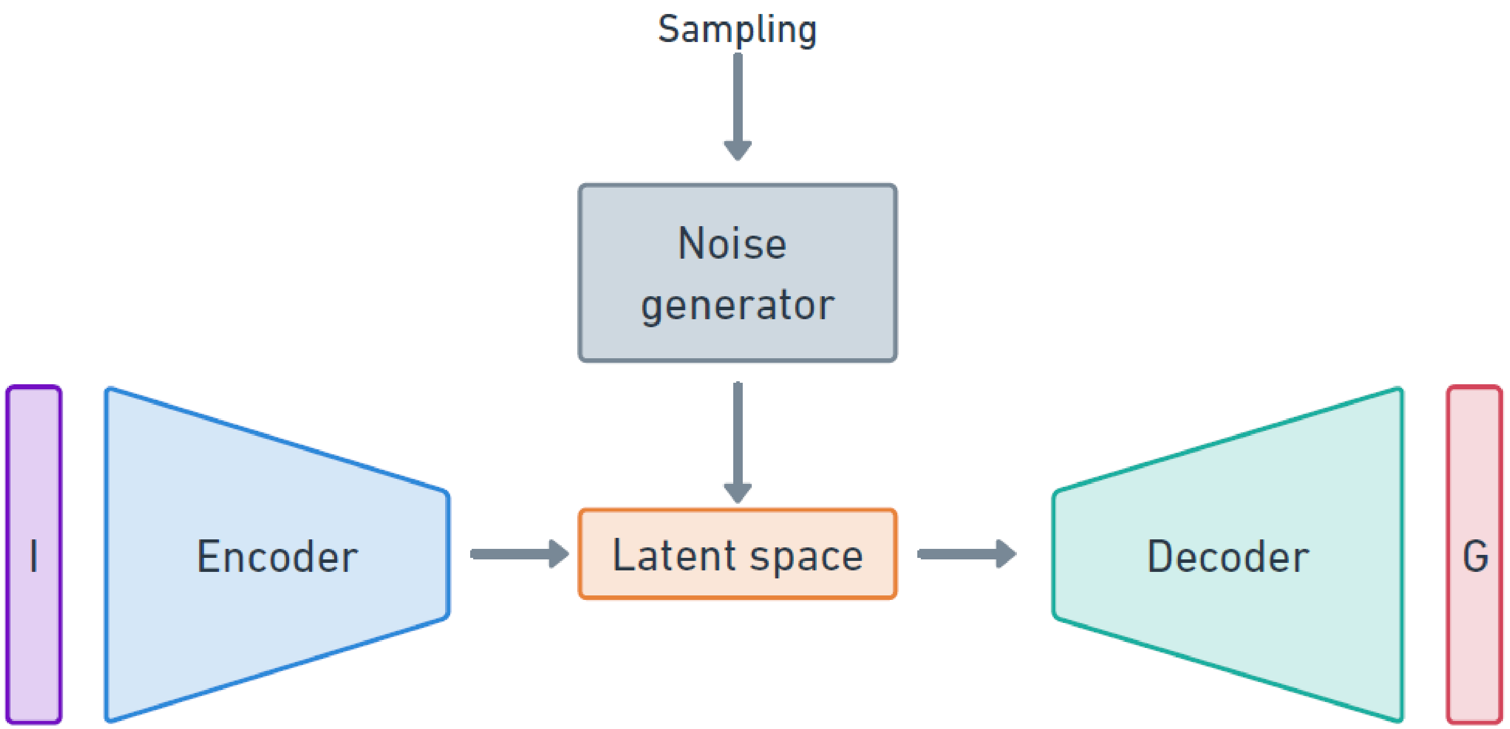
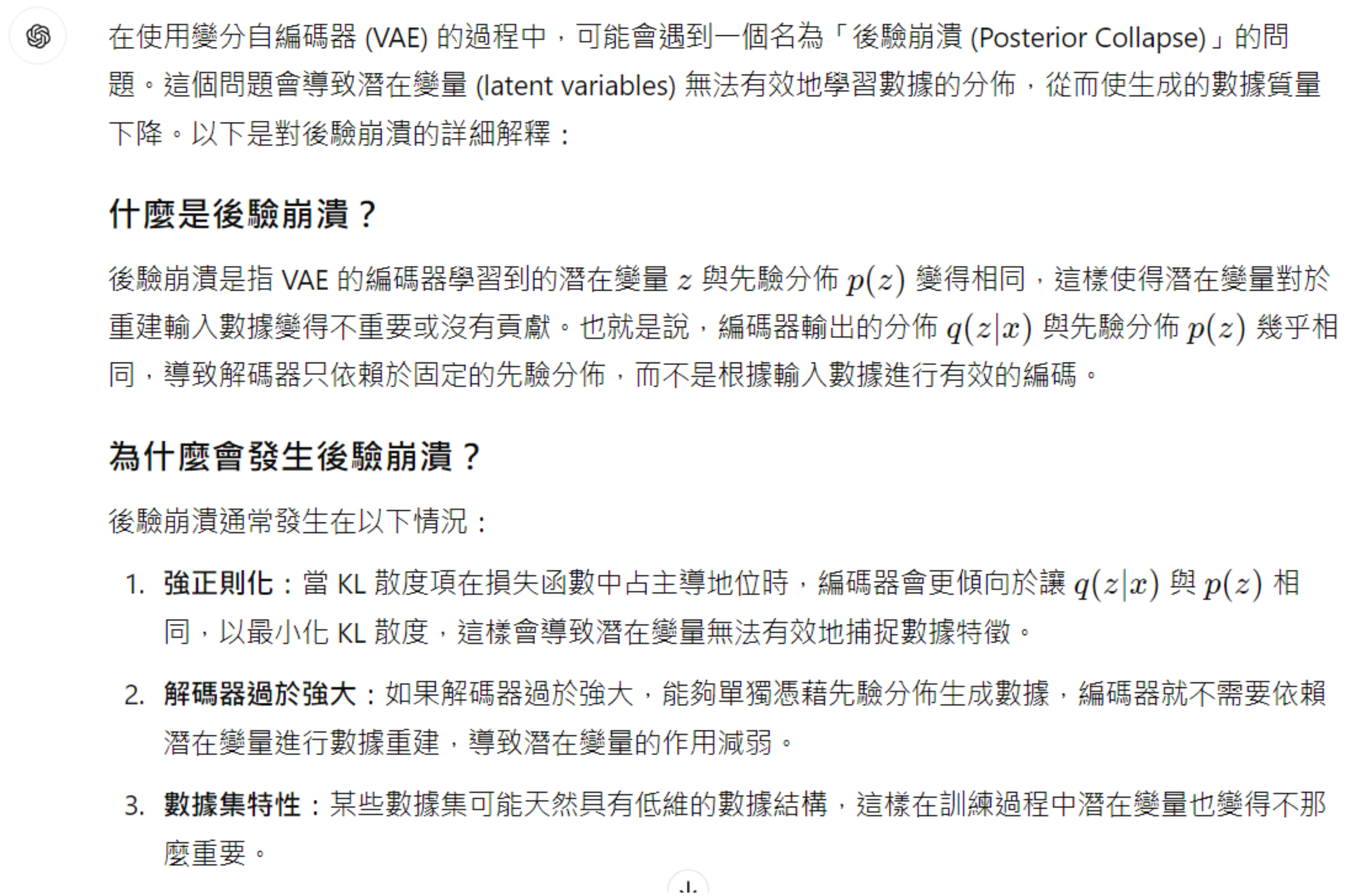


Figure 3.4.: The autoencoder with noise generator design.

Posterior collapse



在使用變分自編碼器 (VAE) 的過程中，可能會遇到一個名為「後驗崩潰 (Posterior Collapse)」的問題。這個問題會導致潛在變量 (latent variables) 無法有效地學習數據的分佈，從而使生成的數據質量下降。以下是對後驗崩潰的詳細解釋：

什麼是後驗崩潰？

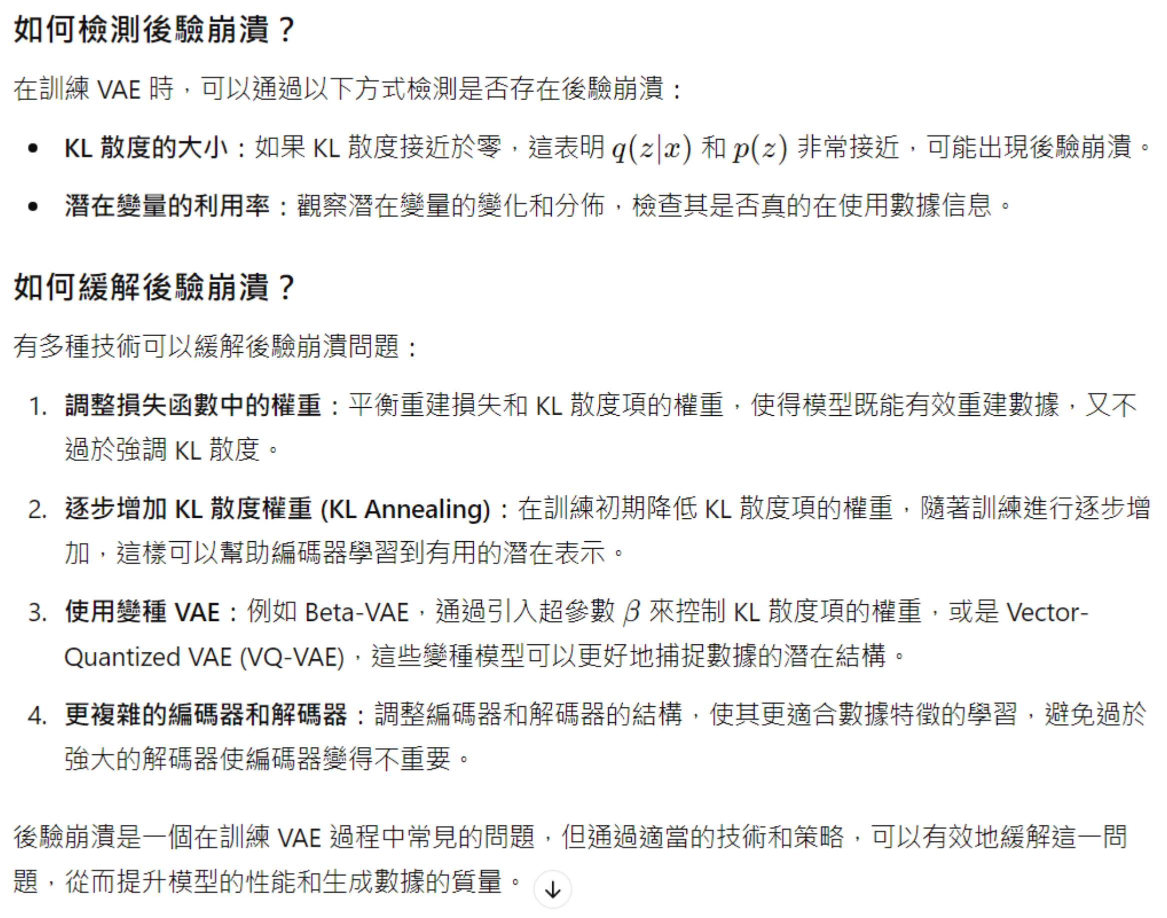
後驗崩潰是指 VAE 的編碼器學習到的潛在變量 z 與先驗分佈 $p(z)$ 變得相同，這樣使得潛在變量對於重建輸入數據變得不重要或沒有貢獻。也就是說，編碼器輸出的分佈 $q(z|x)$ 與先驗分佈 $p(z)$ 幾乎相同，導致解碼器只依賴於固定的先驗分佈，而不是根據輸入數據進行有效的編碼。

為什麼會發生後驗崩潰？

後驗崩潰通常發生在以下情況：

1. **強正則化**：當 KL 散度項在損失函數中占主導地位時，編碼器會更傾向於讓 $q(z|x)$ 與 $p(z)$ 相同，以最小化 KL 散度，這樣會導致潛在變量無法有效地捕捉數據特徵。
2. **解碼器過於強大**：如果解碼器過於強大，能夠單獨憑藉先驗分佈生成數據，編碼器就不需要依賴潛在變量進行數據重建，導致潛在變量的作用減弱。
3. **數據集特性**：某些數據集可能天然具有低維的數據結構，這樣在訓練過程中潛在變量也變得不那麼重要。

Posterior collapse



如何檢測後驗崩潰？

在訓練 VAE 時，可以通過以下方式檢測是否存在後驗崩潰：

- **KL 散度的大小**：如果 KL 散度接近於零，這表明 $q(z|x)$ 和 $p(z)$ 非常接近，可能出現後驗崩潰。
- **潛在變量的利用率**：觀察潛在變量的變化和分佈，檢查其是否真的在使用數據信息。

如何緩解後驗崩潰？

有多種技術可以緩解後驗崩潰問題：

1. **調整損失函數中的權重**：平衡重建損失和 KL 散度項的權重，使得模型既能有效重建數據，又不過於強調 KL 散度。
2. **逐步增加 KL 散度權重 (KL Annealing)**：在訓練初期降低 KL 散度項的權重，隨著訓練進行逐步增加，這樣可以幫助編碼器學習到有用的潛在表示。
3. **使用變種 VAE**：例如 Beta-VAE，通過引入超參數 β 來控制 KL 散度項的權重，或是 Vector-Quantized VAE (VQ-VAE)，這些變種模型可以更好地捕捉數據的潛在結構。
4. **更複雜的編碼器和解碼器**：調整編碼器和解碼器的結構，使其更適合數據特徵的學習，避免過於強大的解碼器使編碼器變得不重要。

後驗崩潰是一個在訓練 VAE 過程中常見的問題，但通過適當的技術和策略，可以有效地緩解這一問題，從而提升模型的性能和生成數據的質量。

Visualization of “Posterior collapse”

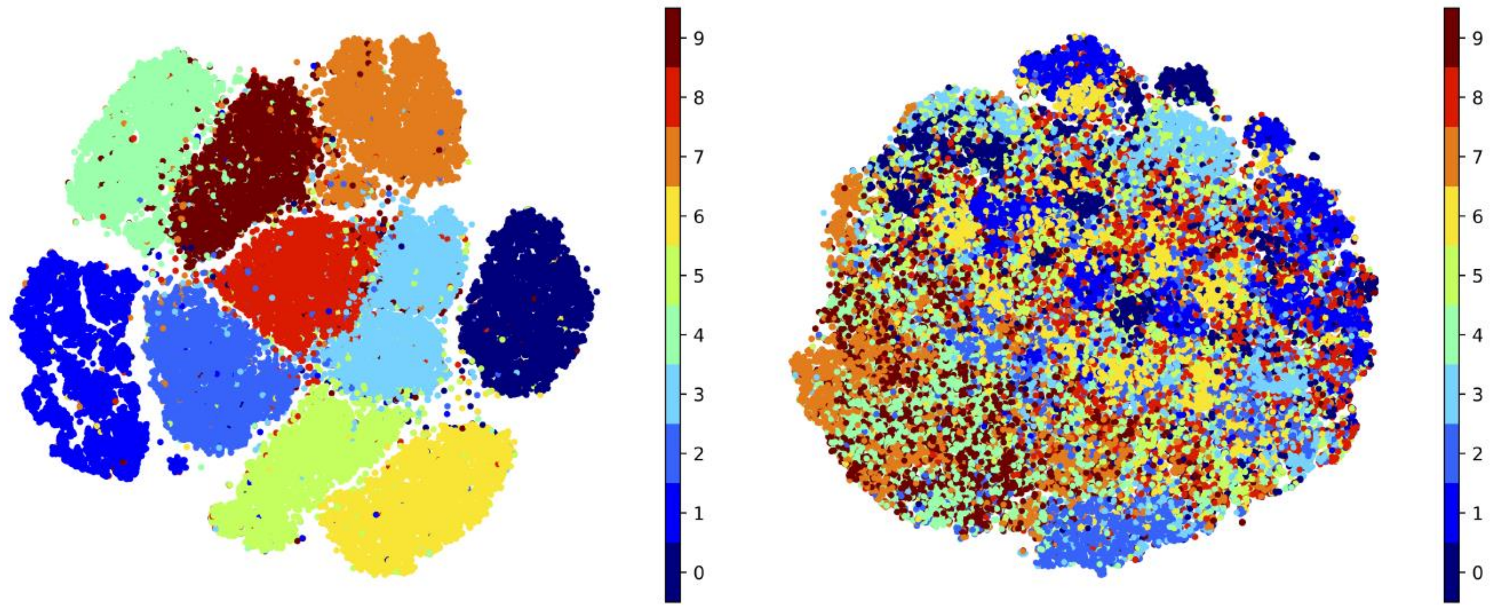
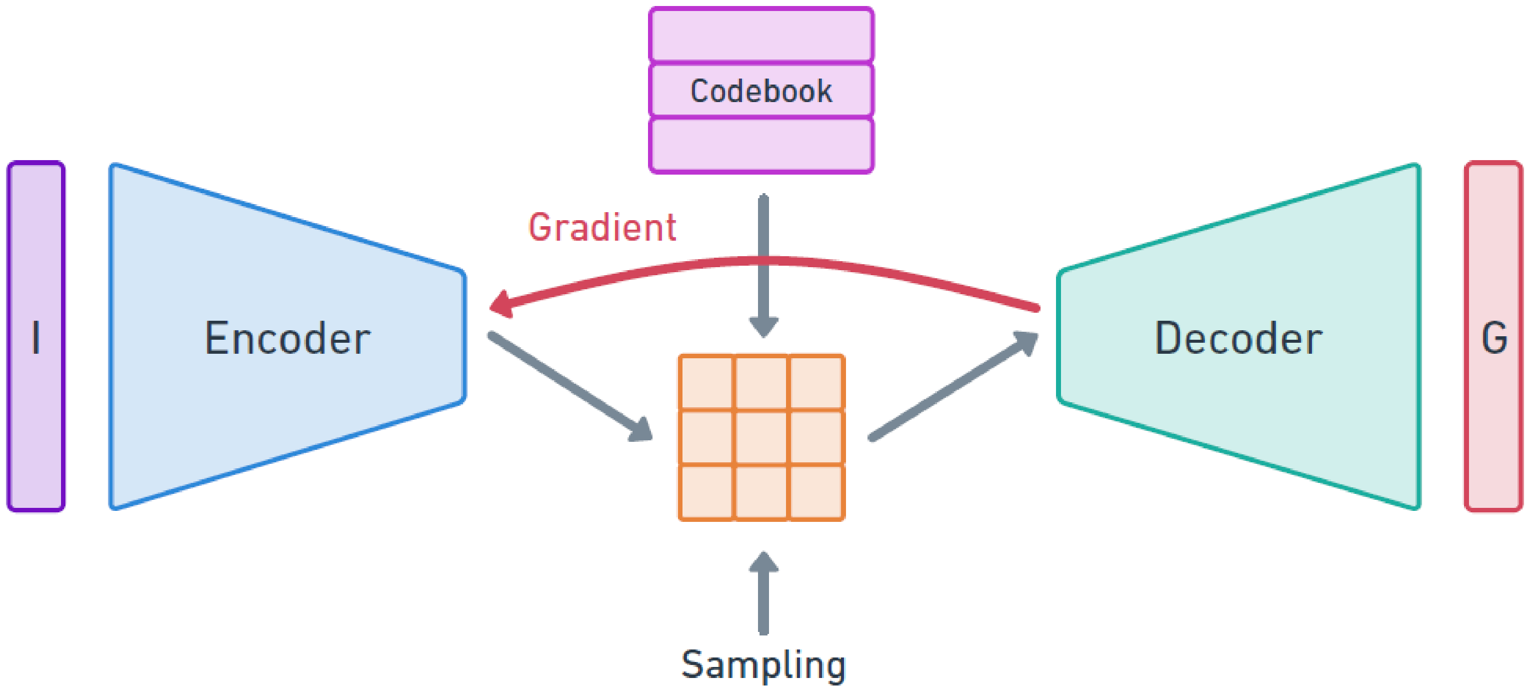


Figure 3.2.: Visualization of a properly structured latent space (left) and a collapsed latent space of a VAE (right).

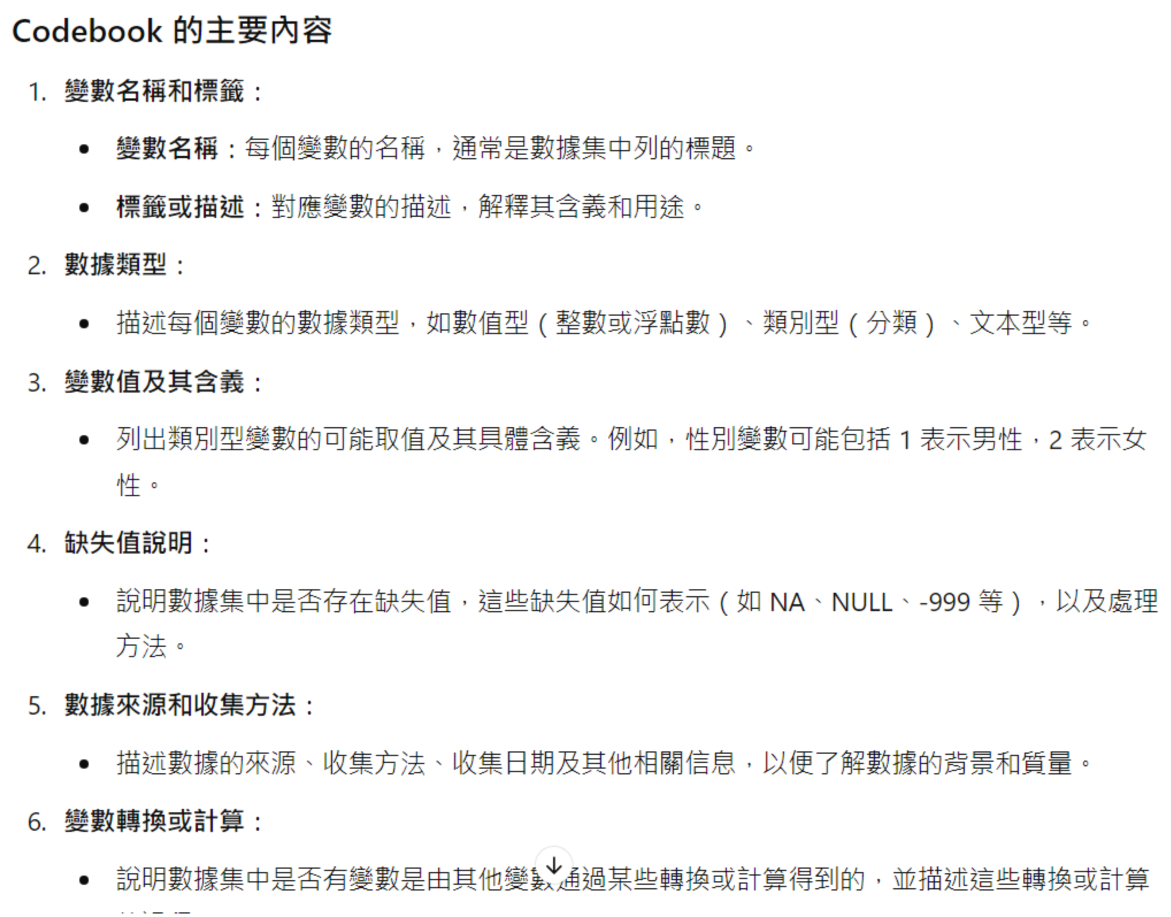
VQ-VAE

VQ-VAE [46] introduce a discrete latent representation through vector quantization. This design involves the use of quantized vectors z_q that are selected as the nearest neighbors from the codebook $\mathcal{Z}$:

$$z_q(x) = \operatorname*{argmin}_{z \in \mathcal{Z}} ||E_\theta(x) - z||_2. \tag{3.15}$$



Codebook



Codebook 的主要內容

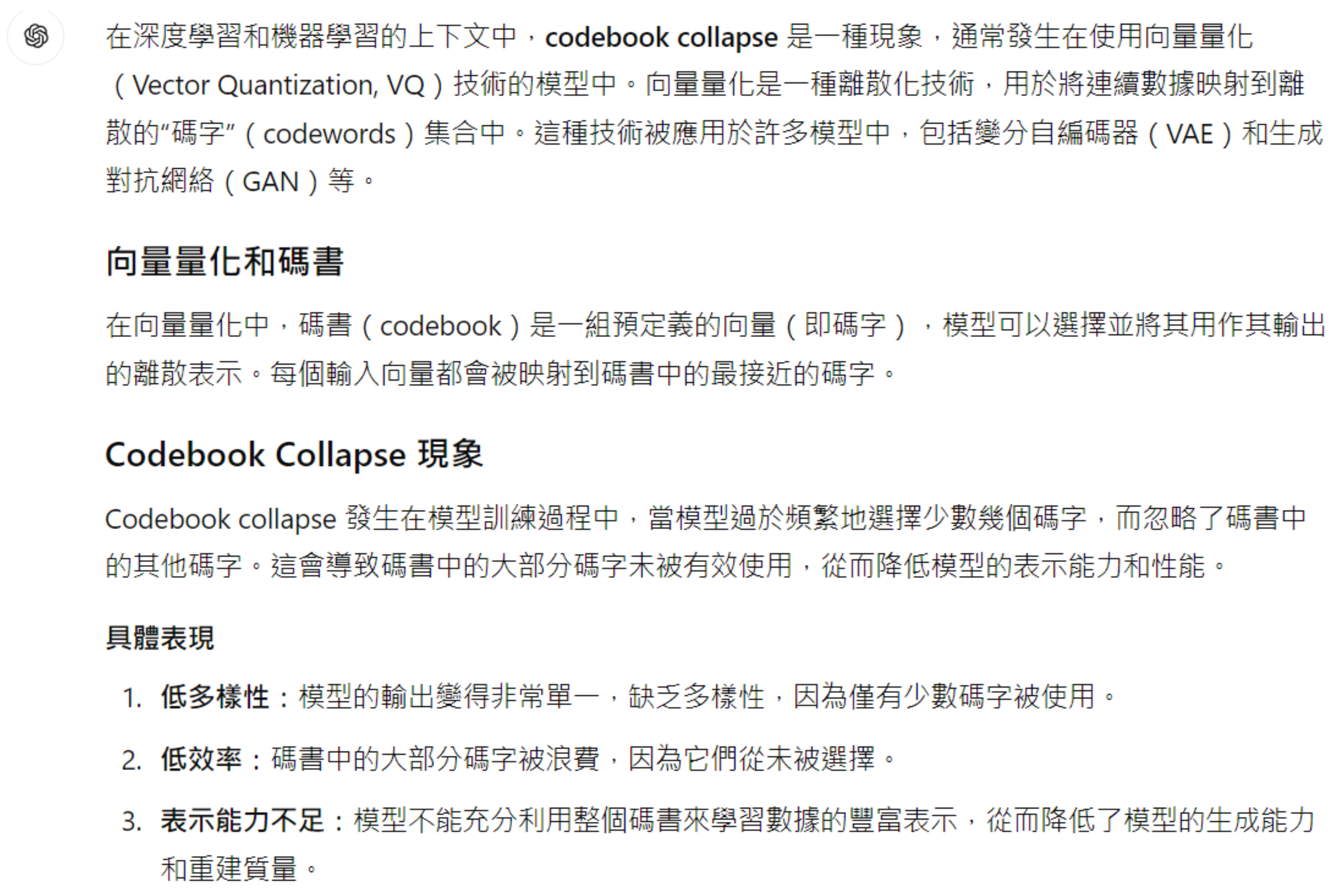
1. **變數名稱和標籤：**
   - **變數名稱**：每個變數的名稱，通常是數據集中列的標題。
   - **標籤或描述**：對應變數的描述，解釋其含義和用途。
2. **數據類型：**
   - 描述每個變數的數據類型，如數值型（整數或浮點數）、類別型（分類）、文本型等。
3. **變數值及其含義：**
   - 列出類別型變數的可能取值及其具體含義。例如，性別變數可能包括 1 表示男性，2 表示女性。
4. **缺失值說明：**
   - 說明數據集中是否存在缺失值，這些缺失值如何表示（如 NA、NULL、-999 等），以及處理方法。
5. **數據來源和收集方法：**
   - 描述數據的來源、收集方法、收集日期及其他相關信息，以便了解數據的背景和質量。
6. **變數轉換或計算：**
   - 說明數據集中是否有變數是由其他變數通過某些轉換或計算得到的，並描述這些轉換或計算

Codebook

Codebook 的作用

1. **提供數據理解：**
   - 幫助數據分析師和研究人員理解數據集中的每個變數及其含義，確保數據的正確使用。
2. **確保數據質量：**
   - 通過詳細記錄數據來源、收集方法和缺失值處理方法，確保數據的質量和可靠性。
3. **促進數據共享：**
   - 在與其他研究團隊或數據使用者共享數據時，提供詳細的 codebook 可以幫助他們快速理解數據，提高合作效率。
4. **增強數據再現性：**
   - 通過詳細記錄數據處理和轉換過程，確保分析過程的透明性和再現性。

Codebook Collapse



在深度學習和機器學習的上下文中，**codebook collapse** 是一種現象，通常發生在使用向量量化（Vector Quantization, VQ）技術的模型中。向量量化是一種離散化技術，用於將連續數據映射到離散的“碼字”（codewords）集合中。這種技術被應用於許多模型中，包括變分自編碼器（VAE）和生成對抗網絡（GAN）等。

向量量化和碼書

在向量量化中，碼書（codebook）是一組預定義的向量（即碼字），模型可以選擇並將其用作其輸出的離散表示。每個輸入向量都會被映射到碼書中的最接近的碼字。

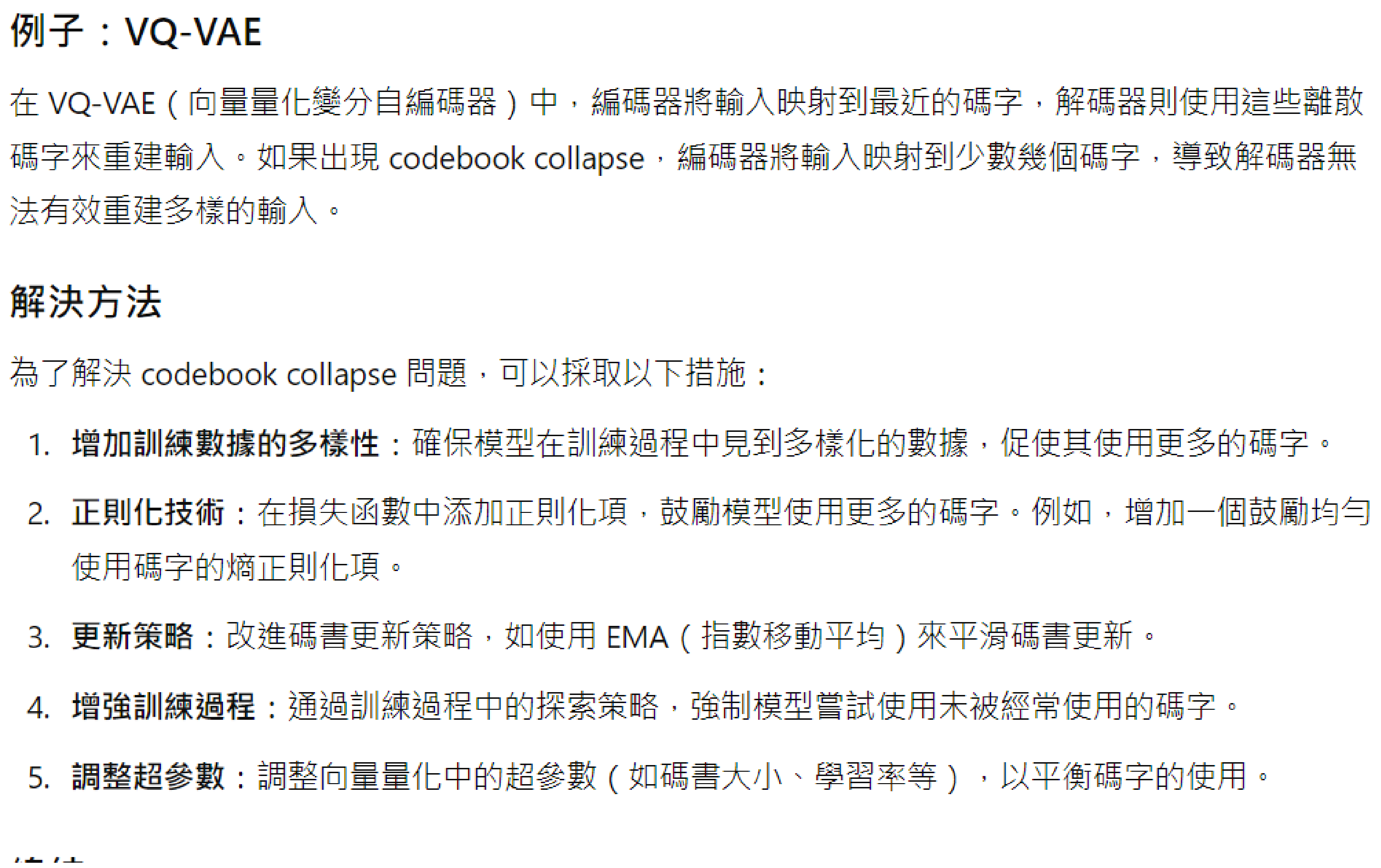
Codebook Collapse 現象

Codebook collapse 發生在模型訓練過程中，當模型過於頻繁地選擇少數幾個碼字，而忽略了碼書中的其他碼字。這會導致碼書中的大部分碼字未被有效使用，從而降低模型的表示能力和性能。

具體表現

1. **低多樣性**：模型的輸出變得非常單一，缺乏多樣性，因為僅有少數碼字被使用。
2. **低效率**：碼書中的大部分碼字被浪費，因為它們從未被選擇。
3. **表示能力不足**：模型不能充分利用整個碼書來學習數據的豐富表示，從而降低了模型的生成能力和重建質量。

Codebook Collapse



例子：VQ-VAE

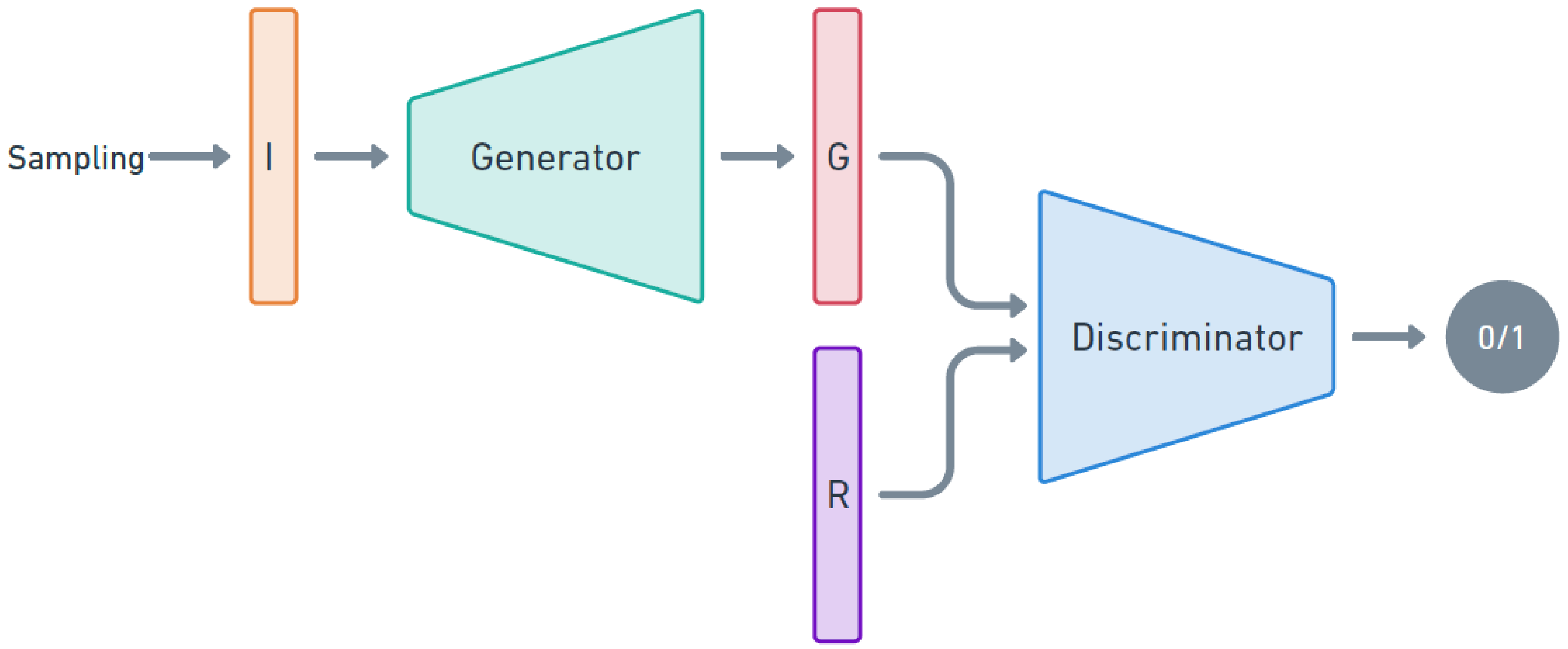
在 VQ-VAE（向量量化變分自編碼器）中，編碼器將輸入映射到最近的碼字，解碼器則使用這些離散碼字來重建輸入。如果出現 codebook collapse，編碼器將輸入映射到少數幾個碼字，導致解碼器無法有效重建多樣的輸入。

解決方法

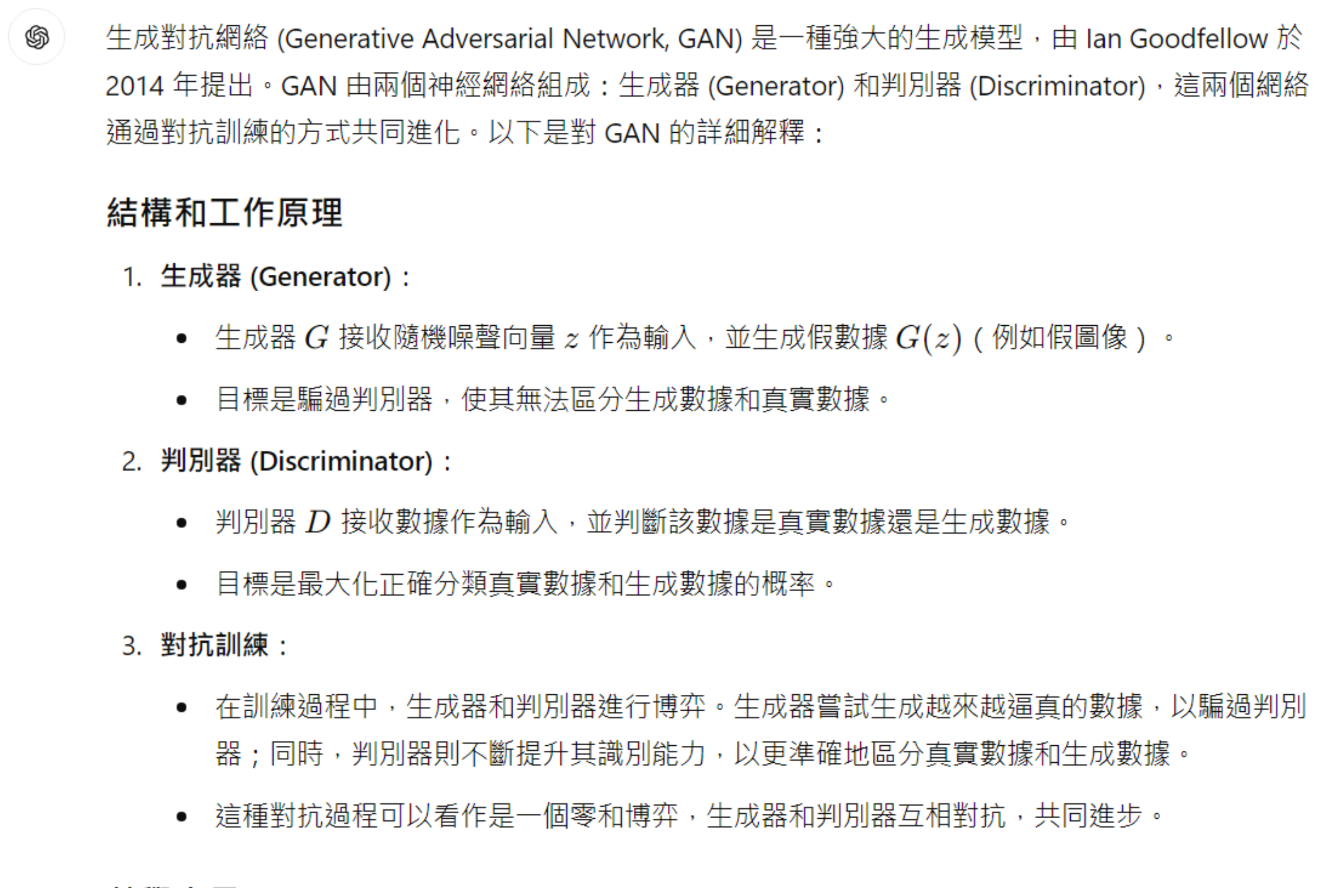
為了解決 codebook collapse 問題，可以採取以下措施：

1. **增加訓練數據的多樣性**：確保模型在訓練過程中見到多樣化的數據，促使其使用更多的碼字。
2. **正則化技術**：在損失函數中添加正則化項，鼓勵模型使用更多的碼字。例如，增加一個鼓勵均勻使用碼字的熵正則化項。
3. **更新策略**：改進碼書更新策略，如使用 EMA（指數移動平均）來平滑碼書更新。
4. **增強訓練過程**：通過訓練過程中的探索策略，強制模型嘗試使用未被經常使用的碼字。
5. **調整超參數**：調整向量量化中的超參數（如碼書大小、學習率等），以平衡碼字的使用。

GAN



GAN



生成對抗網絡 (Generative Adversarial Network, GAN) 是一種強大的生成模型，由 Ian Goodfellow 於 2014 年提出。GAN 由兩個神經網絡組成：生成器 (Generator) 和判別器 (Discriminator)，這兩個網絡通過對抗訓練的方式共同進化。以下是對 GAN 的詳細解釋：

結構和工作原理

1. **生成器 (Generator)：**
   - 生成器 G 接收隨機噪聲向量 z 作為輸入，並生成假數據 $G(z)$（例如假圖像）。
   - 目標是騙過判別器，使其無法區分生成數據和真實數據。
2. **判別器 (Discriminator)：**
   - 判別器 D 接收數據作為輸入，並判斷該數據是真實數據還是生成數據。
   - 目標是最大化正確分類真實數據和生成數據的概率。
3. **對抗訓練：**
   - 在訓練過程中，生成器和判別器進行博弈。生成器嘗試生成越來越逼真的數據，以騙過判別器；同時，判別器則不斷提升其識別能力，以更準確地區分真實數據和生成數據。
   - 這種對抗過程可以看作是一個零和博弈，生成器和判別器互相對抗，共同進步。

GAN

數學表示

GAN 的訓練目標是使生成器生成的數據分佈 p_g 盡可能接近真實數據分佈 p_{data}。其損失函數如下：

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

其中：

- $\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$ 是對真實數據的期望，目標是最大化判別器對真實數據的判斷概率。
- $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$ 是對生成數據的期望，目標是最小化判別器對生成數據的判斷概率。

訓練過程

1. **初始化**：隨機初始化生成器和判別器的權重。
2. **訓練判別器**：
   - 使用一批真實數據進行訓練，更新判別器的權重，使其能夠識別真實數據。
   - 使用生成器生成一批假數據，更新判別器的權重，使其能夠識別假數據。
3. **訓練生成器**：
   - 使用生成器生成一批假數據，更新生成器的權重，使其能夠騙過判別器，使判別器認為這些假數據是真實的。
4. **重複步驟 2 和步驟 3**，直到生成數據的質量達到期望水平。

VQ-GAN

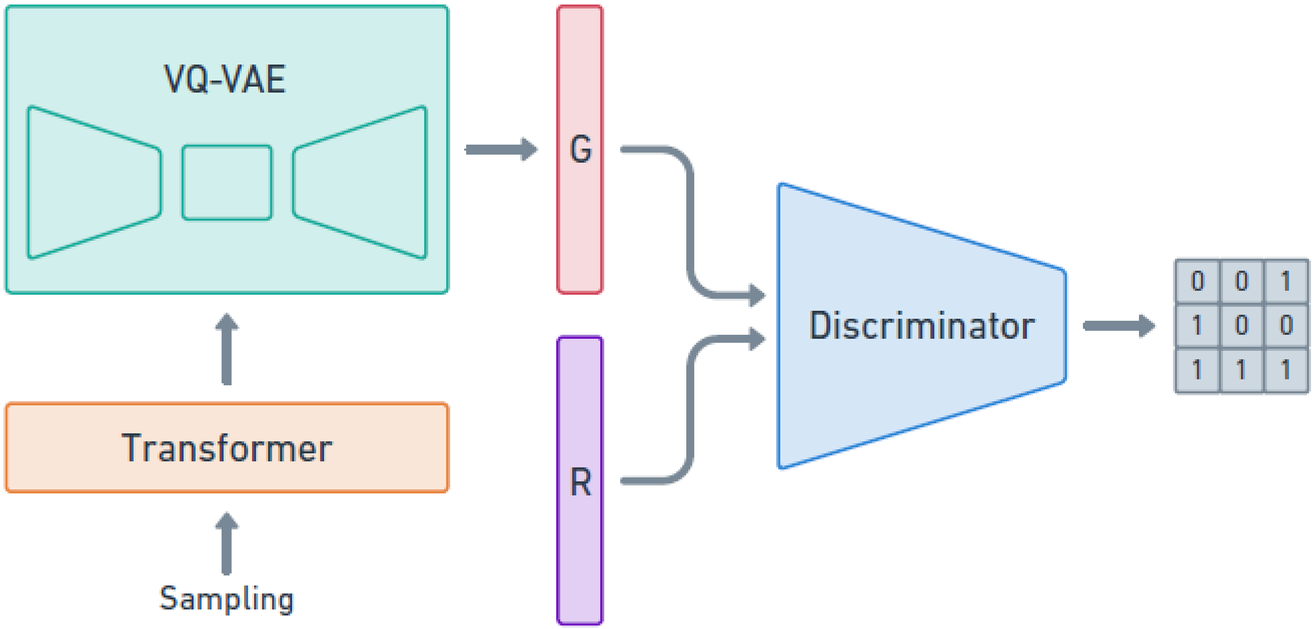


Figure 3.8.: The VQ-GAN design.

VQ-GAN; ViT-VQ-GAN

The VQ-VAE was further extended to the VQ-GAN [28]. Apart from VQ-VAE functioning as a generator, the authors leverage a transformer as a learnable prior, and to address the challenge of processing very long sequences, they introduce a sliding attention window. The VQ-GAN loss function includes a patch-based adversarial loss. The discriminator attempts to distinguish fragments of the generated image from the corresponding fragments of the real image, instead of evaluating the entire output. Consequently, a grid of values ranging from 0 to 1 is produced, as depicted in Figure 3.8.

Instead of $l2$ loss as a reconstruction loss, VQ-GAN employs a perceptual loss [47]. This method evaluates the similarity between two images, leveraging feature representations extracted from a pre-trained convolutional neural network (CNN) to better capture textural and structural differences. The concept is depicted in Figure 3.9, where CNN embeddings of real and generated images are compared using the $l2$ loss.

Vision transformer-based VQ-GAN (ViT-VQ-GAN) [49] extends the VQ-GAN design by incorporating ViT into the generator and enhancing the utilization of the codebook (as detailed in Section 3.5.2). Regarding the loss function, ViT-VQ-GAN integrates logit-laplace loss [50], $l2$ loss, perceptual loss, and adversarial loss in the reconstruction loss term.

Diffusion Model

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I), \tag{3.25}$$

where $\alpha_t = 1 - \beta_t$ and β_t is a variance schedule. The variance schedule controls the amount of noise added at each step, ensuring a gradual and stable diffusion, which is essential for the model's performance and the stability of the reverse process. The data point any time step t can be derived analytically. Given $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, the expression is:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I). \tag{3.26}$$

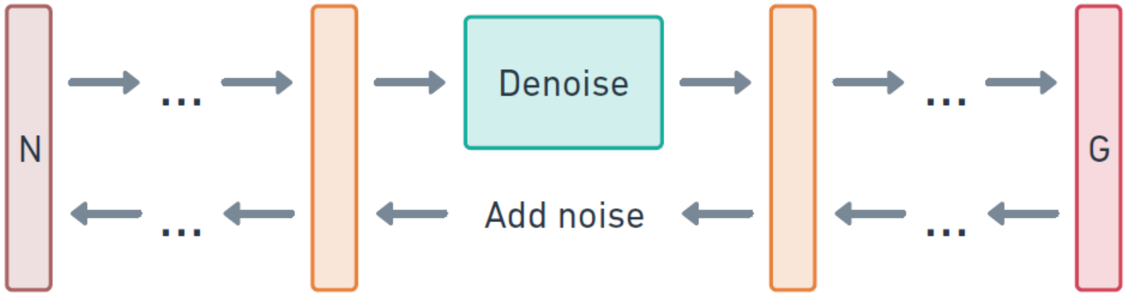
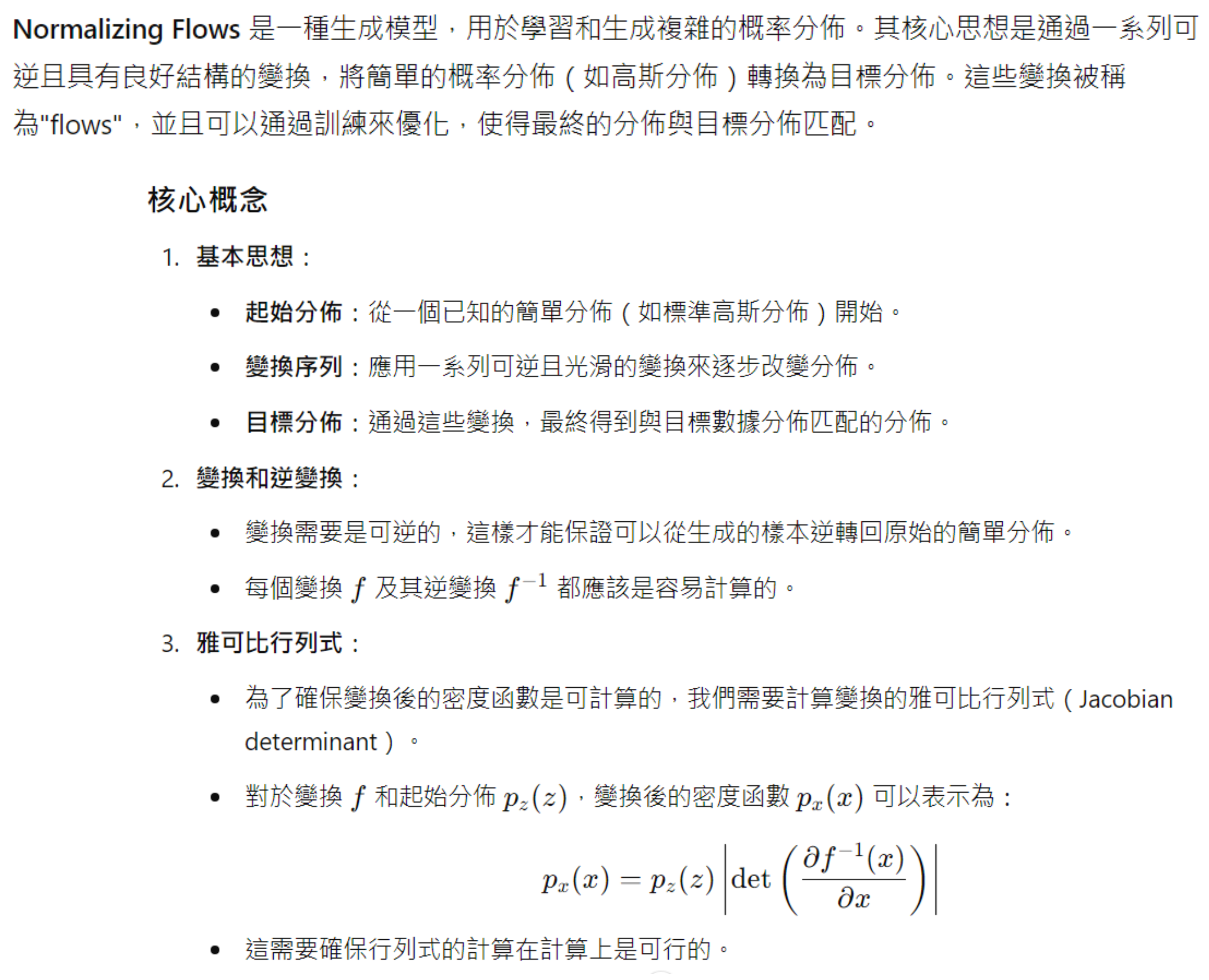


Figure 3.10.: The diffusion models design.

Normalizing flows (NFs)



Normalizing Flows 是一種生成模型，用於學習和生成複雜的概率分佈。其核心思想是通過一系列可逆且具有良好結構的變換，將簡單的概率分佈（如高斯分佈）轉換為目標分佈。這些變換被稱為"flows"，並且可以通過訓練來優化，使得最終的分佈與目標分佈匹配。

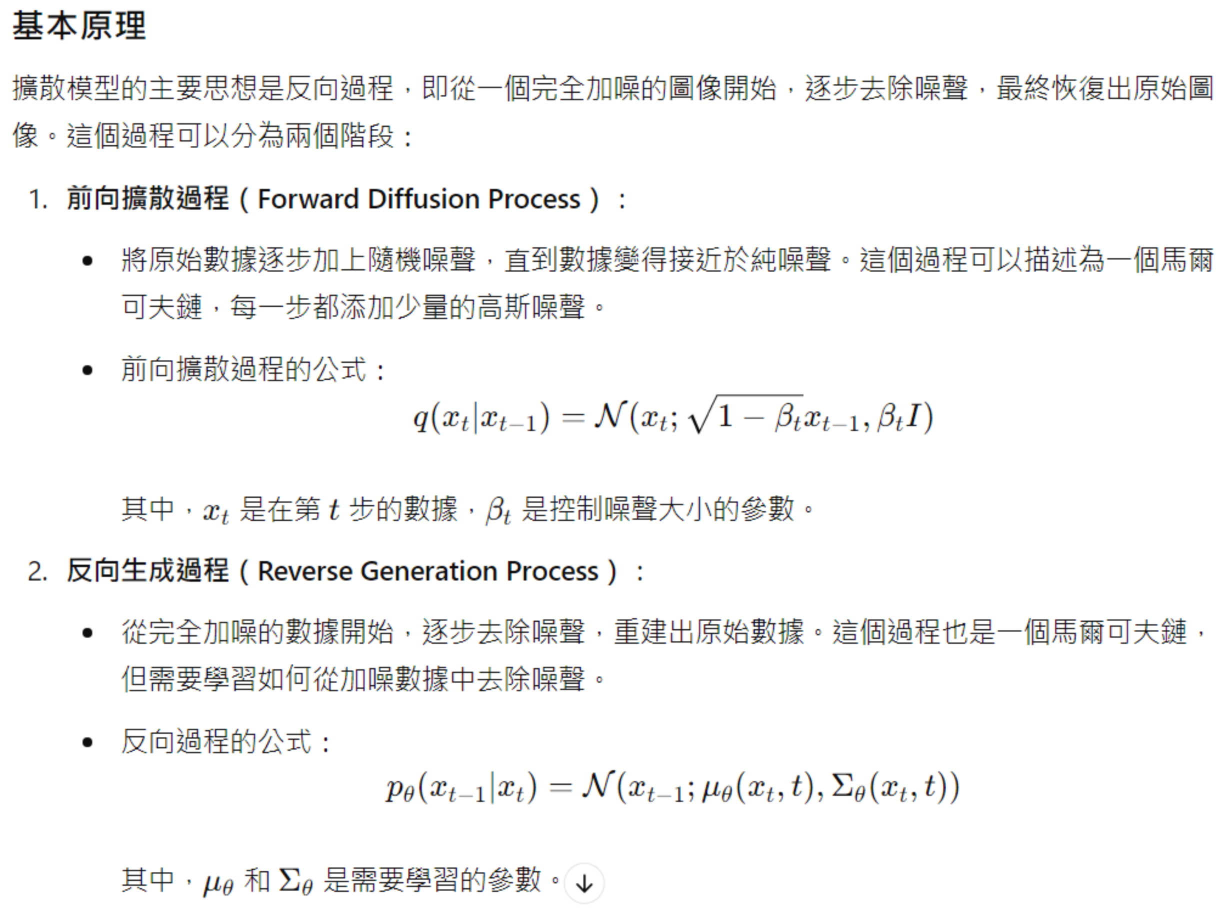
核心概念

1. **基本思想**：
   - **起始分佈**：從一個已知的簡單分佈（如標準高斯分佈）開始。
   - **變換序列**：應用一系列可逆且光滑的變換來逐步改變分佈。
   - **目標分佈**：通過這些變換，最終得到與目標數據分佈匹配的分佈。
2. **變換和逆變換**：
   - 變換需要是可逆的，這樣才能保證可以從生成的樣本逆轉回原始的簡單分佈。
   - 每個變換 f 及其逆變換 f^{-1} 都應該是容易計算的。
3. **雅可比行列式**：
   - 為了確保變換後的密度函數是可計算的，我們需要計算變換的雅可比行列式（Jacobian determinant）。
   - 對於變換 f 和起始分佈 $p_z(z)$，變換後的密度函數 $p_x(x)$ 可以表示為：

$$p_x(x) = p_z(z) \left| \det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

   - 這需要確保行列式的計算在計算上是可行的。

Normalizing flows (NFs)



基本原理

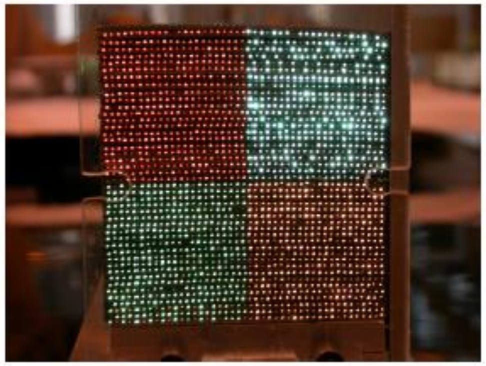
擴散模型的主要思想是反向過程，即從一個完全加噪的圖像開始，逐步去除噪聲，最終恢復出原始圖像。這個過程可以分為兩個階段：

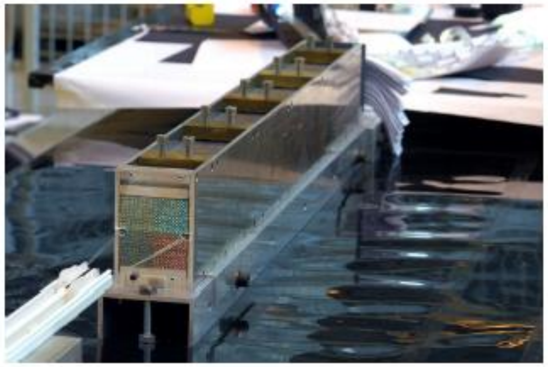
1. **前向擴散過程（Forward Diffusion Process）**：
   - 將原始數據逐步加上隨機噪聲，直到數據變得接近於純噪聲。這個過程可以描述為一個馬爾可夫鏈，每一步都添加少量的高斯噪聲。
   - 前向擴散過程的公式：
   $$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$
   其中，x_t 是在第 t 步的數據，β_t 是控制噪聲大小的參數。
2. **反向生成過程（Reverse Generation Process）**：
   - 從完全加噪的數據開始，逐步去除噪聲，重建出原始數據。這個過程也是一個馬爾可夫鏈，但需要學習如何從加噪數據中去除噪聲。
   - 反向過程的公式：
   $$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$
   其中，μ_θ 和 Σ_θ 是需要學習的參數。

Detector

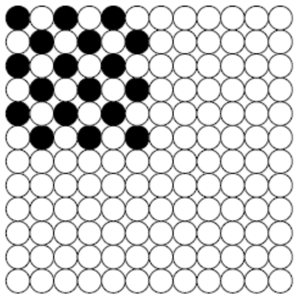
The Zero Degree Calorimeter (ZDC) [7] in the ALICE experiment measures particle showers to determine the centrality of collisions, helping to understand their dynamics [8]. Comprising four calorimeters, the system includes two dedicated for proton detection (ZP) and two for neutron detection (ZN). An image of ZN is provided in Figure 1.2.

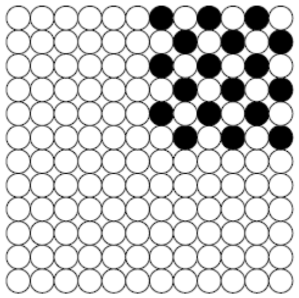
(a) The quartz fibers divided into channels.

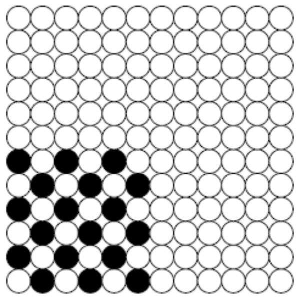

(b) The matrix of quartz fibers.

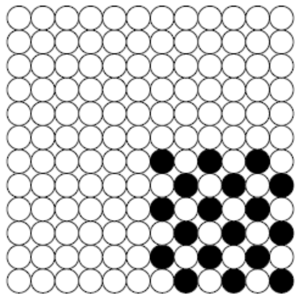

(c) The ZN detector.

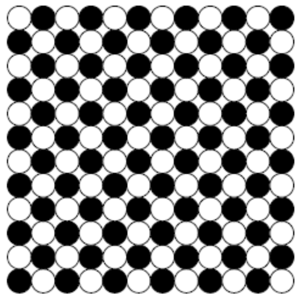
Analysis

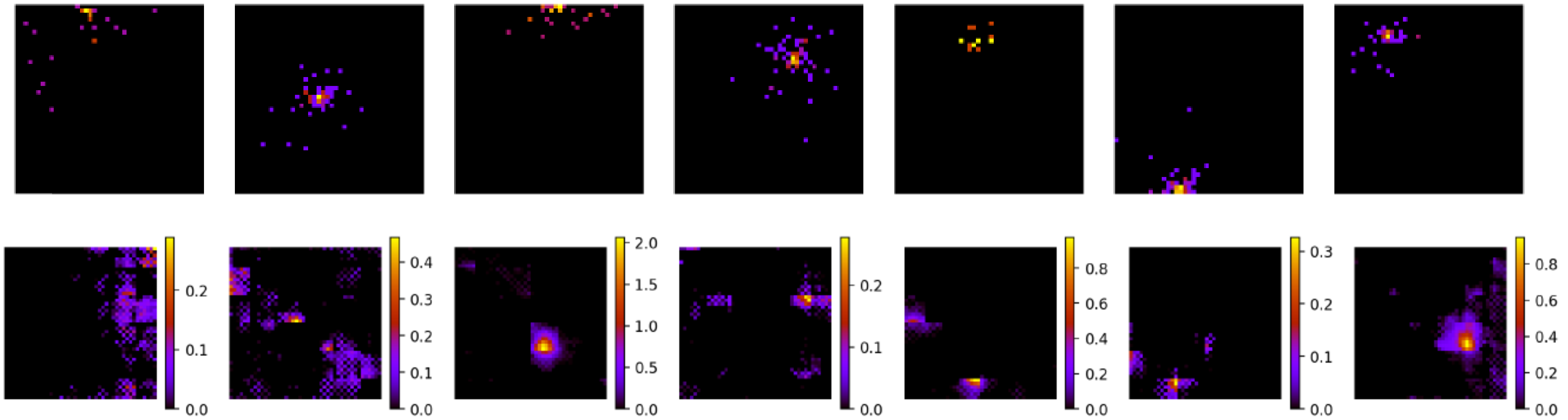

(a) Channel 1.


(b) Channel 2.


(c) Channel 3.


(d) Channel 4.


(e) Channel 5.



The following columns are the responses to different particles, the first from the left is $\pi+$, followed by γ particles (the most common in this dataset), and the last is K^0_S. The rows show independent runs for the same particles.

Dataset

The dataset consists of two parts: particle properties, which serve as conditional variables, and ZN responses, which have dimensions $44 \times 44 \times 1$, making them suitable to be treated as images by generative models. The particle properties are represented as 9-dimensional vectors that contain information on energy, primary vertex positions in the x, y, and z dimensions, momenta in the x, y, and z dimensions, mass, and charge. Furthermore, PDG identifiers can be included to specify the type of particle.

The dataset contains 306780 samples, with all detector responses having at least 10 photons (filtering out zero responses is a separate problem and is not covered in this work, but can be found in the literature [90]). Among all examples, 99695 have a different sum of photons, and there are only 1805 unique feature vectors, indicating that the dataset might not cover the entire space of possible particle features. Additionally, the dataset is highly imbalanced – 60.83% are photons (γ), 23.34% are neutrons (n), 5.30% are protons (p), 3.08% are lambda baryons (Λ), 2.09% are short-lived kaons (K_S^0), 1.79% are long-lived kaons (K_L^0), 1.30% are pions ($\pi+$), and the others represent less than 1% of the dataset size. In general, the dataset includes 21 different types of particles.

Dataset

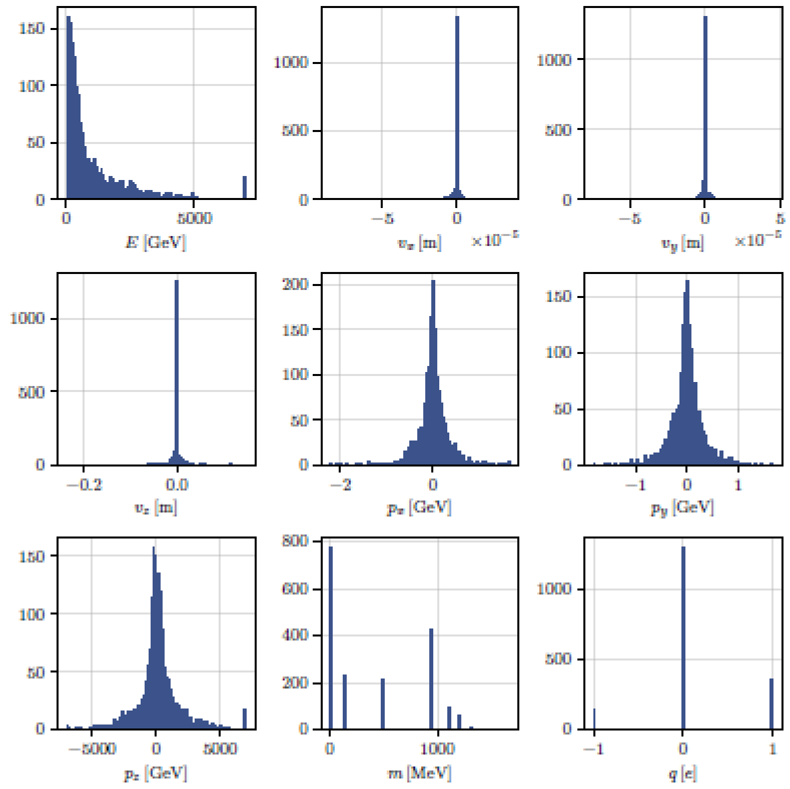
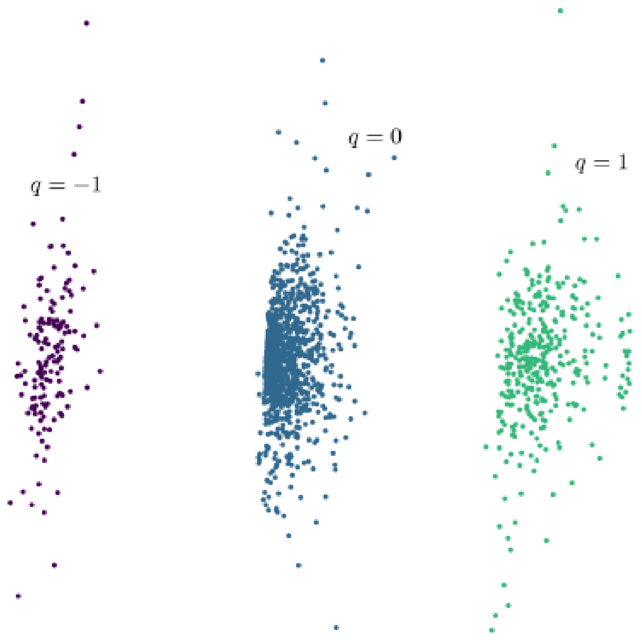


Table 4.1.: The features importance for particle diversity.

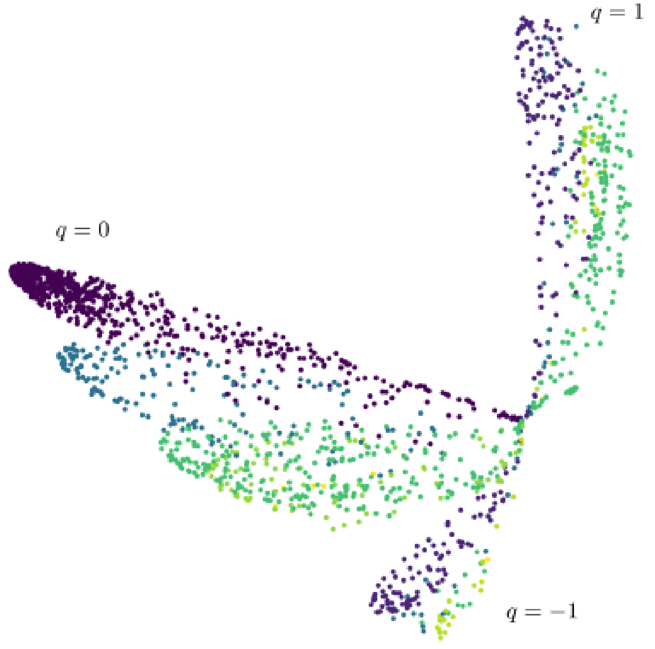
Feature	Importance
E	0.324
v_x	0.001
v_y	0.005
v_z	0.001
p_x	0.050
p_y	0.054
p_z	0.013
m	0.506
q	0.023
PDGID	0.021

Dataset visualizations

principal component analysis (PCA)

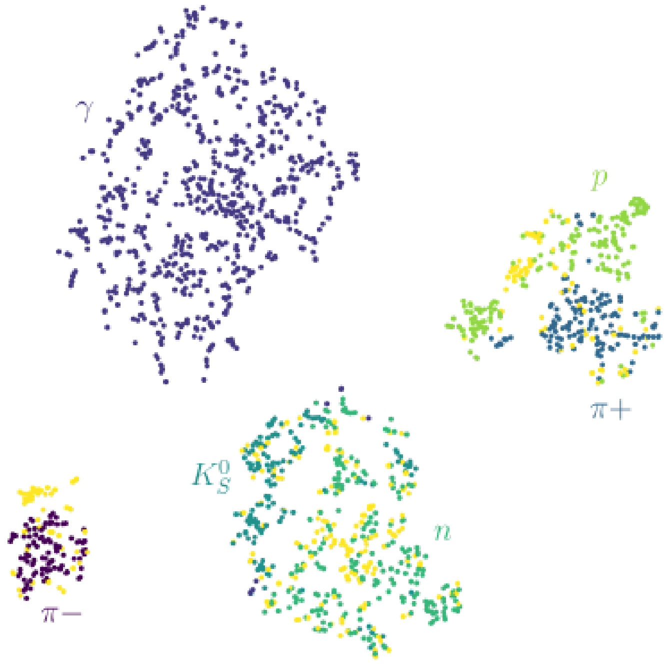


(a) Particle features visualization using linear PCA. The colors indicate the charge.



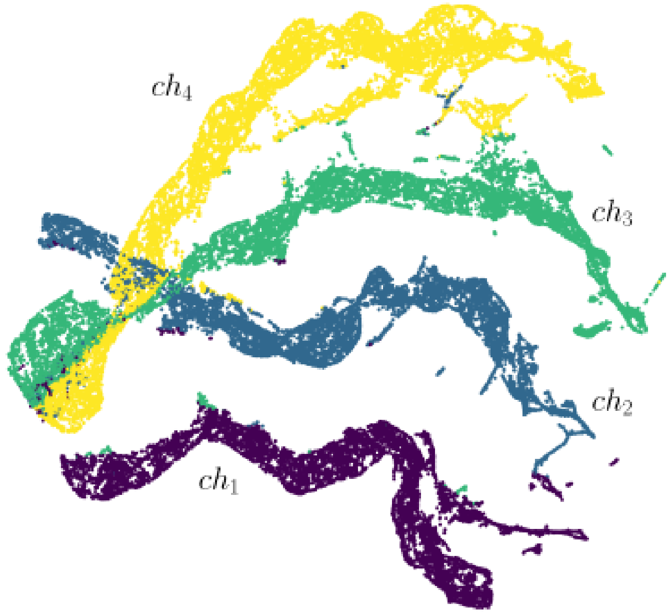
(b) Particle features visualization using kernel PCA. The colors indicate the mass, and the labels identify clusters with different charges.

Dataset visualizations



(c) Particle features visualization using t-SNE. The colors denote the types of the particles.

principal component analysis (PCA)



(d) Response channels visualization using UMAP. The colors indicate the channel with the highest value.

Evaluation metrics

- **Wasserstein metric** is the most commonly used measure of model performance. This metric is calculated as the mean Wasserstein-1 distance between the original and generated histograms of the photon sums.
- **Mean absolute error (MAE),** which compares the l1 distance of the channel values. Unlike the Wasserstein metric, this is a local metric that directly compares the channel values of the original and generated samples.
- **Pixel-wise root mean square error (RMSE),** which directly compares pixels.

Evaluation metrics

$$\text{Wasserstein-1}(w, \hat{w}) = \frac{1}{5} \sum_{i=1}^{5} \int_0^1 \left| F_{w_i}^{-1}(z) - F_{\hat{w}_i}^{-1}(z) \right| dz, \tag{4.2}$$

$$\text{MAE}(w, \hat{w}) = \frac{1}{n} \sum_{k=0}^{n} \frac{1}{5} \sum_{i=1}^{5} |w_i^k - \hat{w}_i^k|, \tag{4.3}$$

$$\text{RMSE}(x, \hat{x}) = \sqrt{\frac{1}{n} \sum_{k=0}^{n} \frac{1}{44 \cdot 44} \sum_{i=1}^{44} \sum_{j=1}^{44} (x_{ij}^k - \hat{x}_{ij}^k)^2}, \tag{4.4}$$

where F_q^{-1} is the inverse cumulative distribution function of the distribution q, w_i denotes the distribution of the i-th channel, n refers to the number of evaluated examples, w_i^k represents the value of the i-th channel of the k-th response, x_{ij}^k is the value of the pixel with i and j coordinates of the k-th response, and $\hat{w}$ and $\hat{x}$ are the corresponding predicted values.

Training

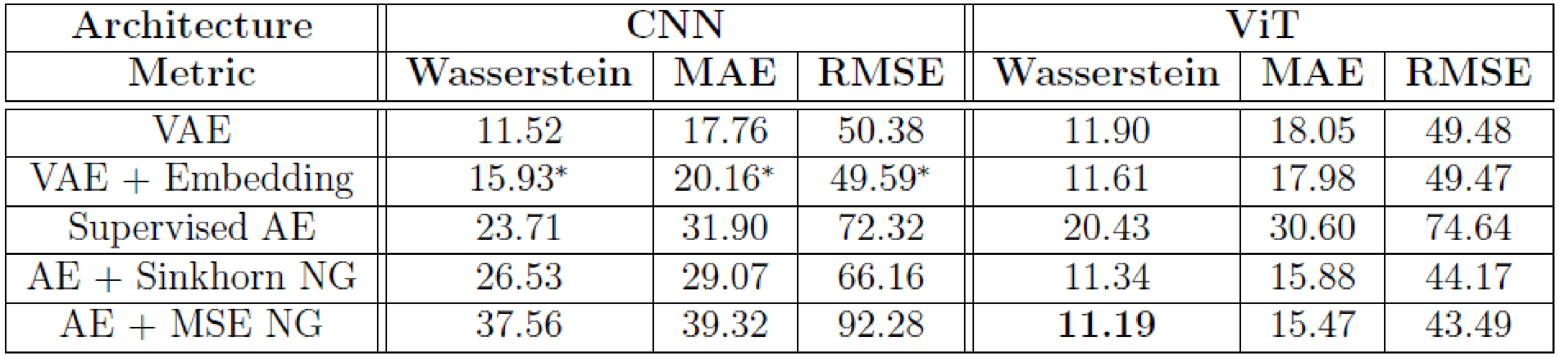
- The dataset was divided into training, validation, and testing subsets in proportions of 70%, 10%, and 20%, respectively.
- Training was carried out on the training set, the parameters were tuned on the validation set, and the final results were calculated on the test set.
- The AdamW optimizer was used, with hyperparameters fine-tuned through more than 100 trials per model with Optuna software optimizing the Wasserstein metric.
- The optimization procedure leveraged the tree-structured Parzen Estimator (TPE) sampler without pruning and considered the following parameters: learning rate, β1, β2, ϵ, and the use of the cosine learning rate schedule, weight decay, and Nesterov momentum.

Training

- The training was performed on a single NVIDIA A100 GPU with 40 GB of memory on the Athena supercomputer2. All models were trained for 100 epochs in batches of size 256.
- At the end of each epoch, the weights were stored to allow for reconstruction at any time during the training.

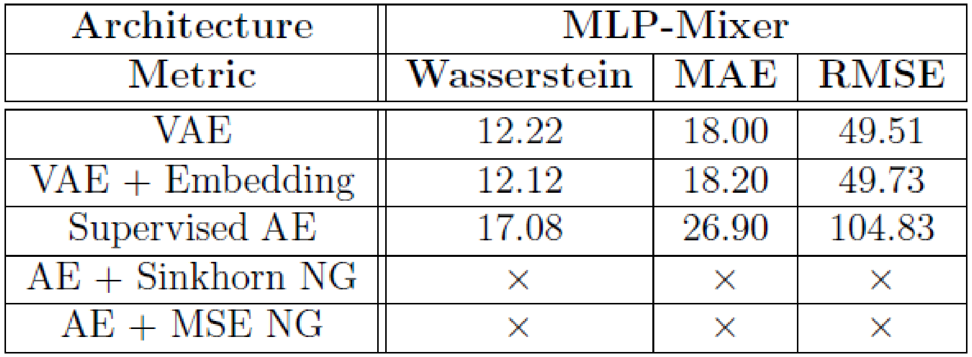
Performance

Table 4.3.: Performance of autoencoders across various architectures (part I).[3]



Architecture	CNN			ViT		
Metric	Wasserstein	MAE	RMSE	Wasserstein	MAE	RMSE
VAE	11.52	17.76	50.38	11.90	18.05	49.48
VAE + Embedding	15.93*	20.16*	49.59*	11.61	17.98	49.47
Supervised AE	23.71	31.90	72.32	20.43	30.60	74.64
AE + Sinkhorn NG	26.53	29.07	66.16	11.34	15.88	44.17
AE + MSE NG	37.56	39.32	92.28	**11.19**	15.47	43.49

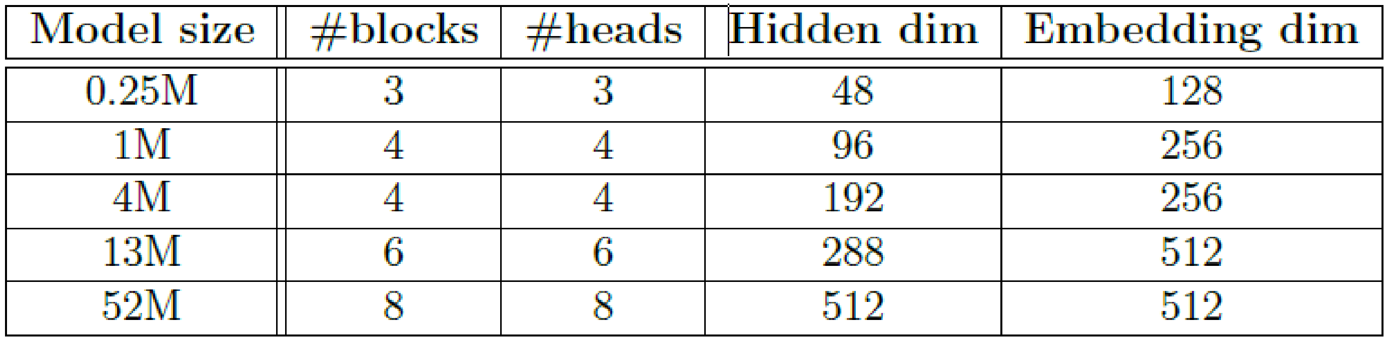
Table 4.4.: Performance of autoencoders across various architectures (part II).[4]



Architecture	MLP-Mixer		
Metric	Wasserstein	MAE	RMSE
VAE	12.22	18.00	49.51
VAE + Embedding	12.12	18.20	49.73
Supervised AE	17.08	26.90	104.83
AE + Sinkhorn NG	×	×	×
AE + MSE NG	×	×	×

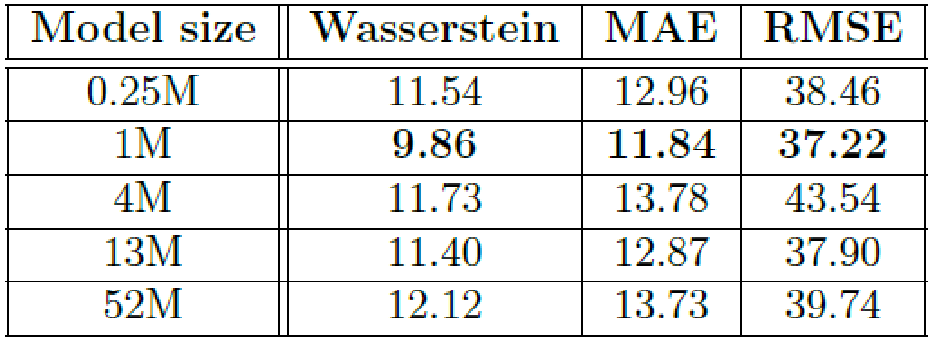
Model Size vs. Performance

Table 4.5.: VQ-VAE autoencoder specification depending on model size.



Model size	#blocks	#heads	Hidden dim	Embedding dim
0.25M	3	3	48	128
1M	4	4	96	256
4M	4	4	192	256
13M	6	6	288	512
52M	8	8	512	512

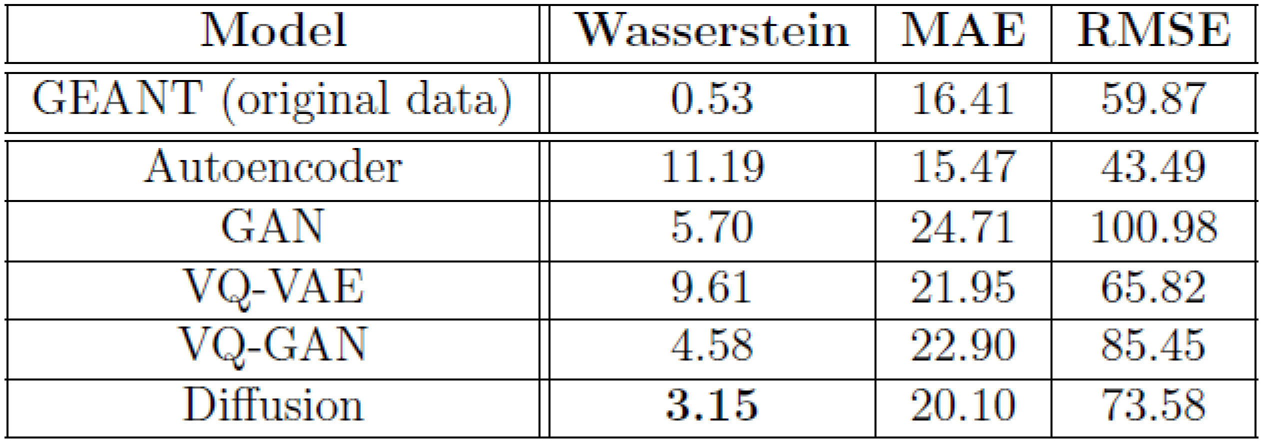
Table 4.6.: VQ-VAE reconstruction performance depending on model size.



Model size	Wasserstein	MAE	RMSE
0.25M	11.54	12.96	38.46
1M	**9.86**	**11.84**	**37.22**
4M	11.73	13.78	43.54
13M	11.40	12.87	37.90
52M	12.12	13.73	39.74

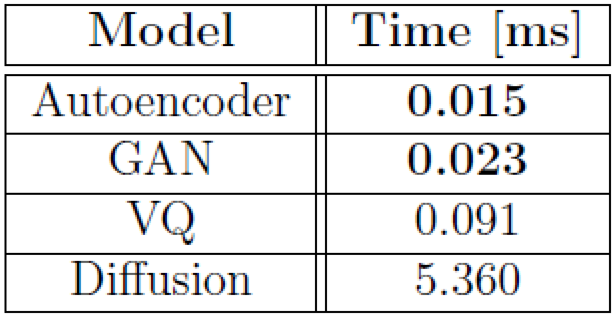
Performance vs. Framework

Table 4.7.: Performance comparison of generative frameworks.



Model	Wasserstein	MAE	RMSE
GEANT (original data)	0.53	16.41	59.87
Autoencoder	11.19	15.47	43.49
GAN	5.70	24.71	100.98
VQ-VAE	9.61	21.95	65.82
VQ-GAN	4.58	22.90	85.45
Diffusion	**3.15**	20.10	73.58

Table 4.8.: Generation time of various generative models.



Model	Time [ms]
Autoencoder	**0.015**
GAN	**0.023**
VQ	0.091
Diffusion	5.360

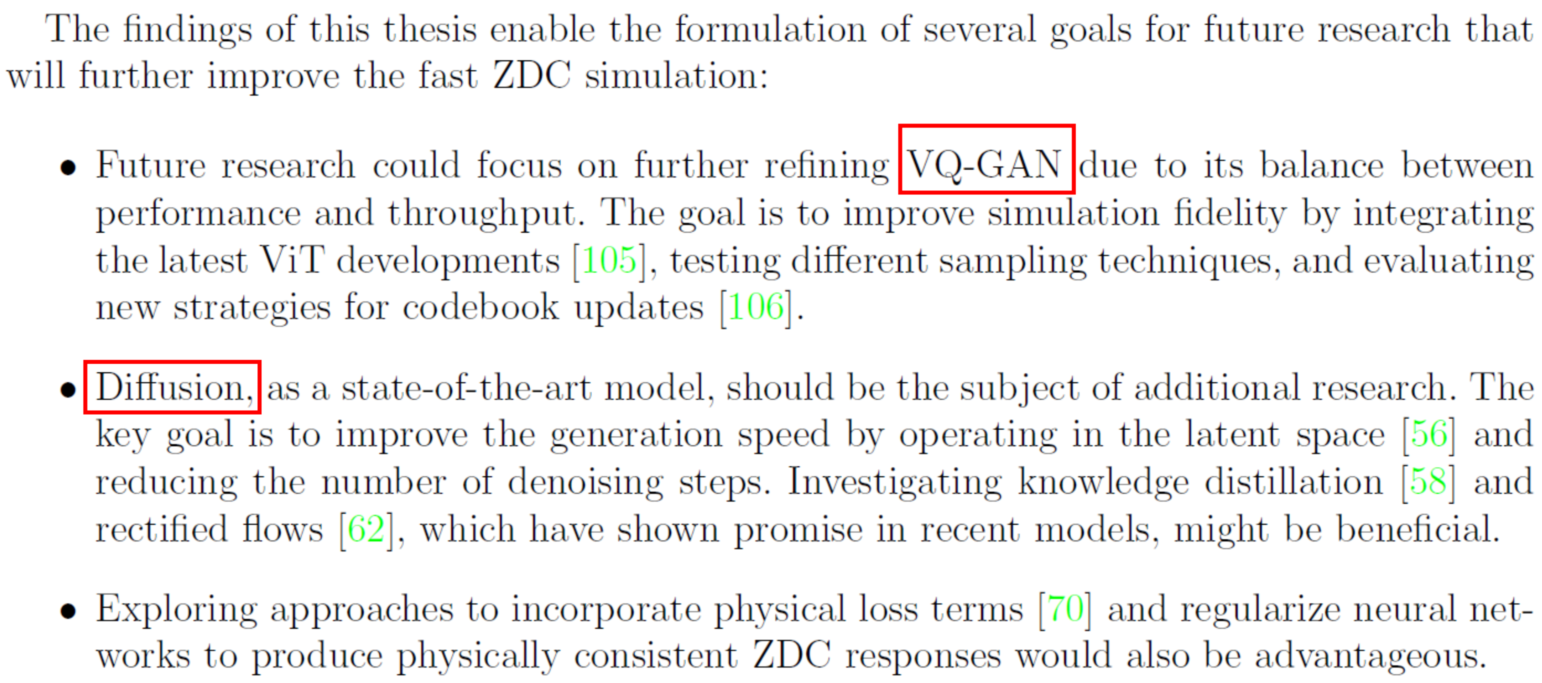
Systematic Study

- Autoencoder
  - Latent space size and posterior collapse
- Generative adversarial network
  - Improving training stability
  - Improving fidelity of simulations
- Vector quantization
  - Codebook utilization
  - Codebook size
  - Loss functions
  - Transformer settings
- Diffusion
  - Noise scheduler
  - Number of denoising steps
  - Denoising Diffusion Implicit Models

Conclusion (1)

- The **GAN** research showed that a classically formulated GAN, especially when combined with a postprocessing step, yields the best performance.
- **Diffusion models** outperform others, even with a limited number of denoising steps, and adjusting their parameters can further improve performance. They havea longer generation time.
- **VQ-GAN**, while slightly less accurate, offers a good compromise with a faster generation time, making them suitable for scenarios that involve extremely fast simulations. VQ-GAN is recommended.

Conclusion (2)

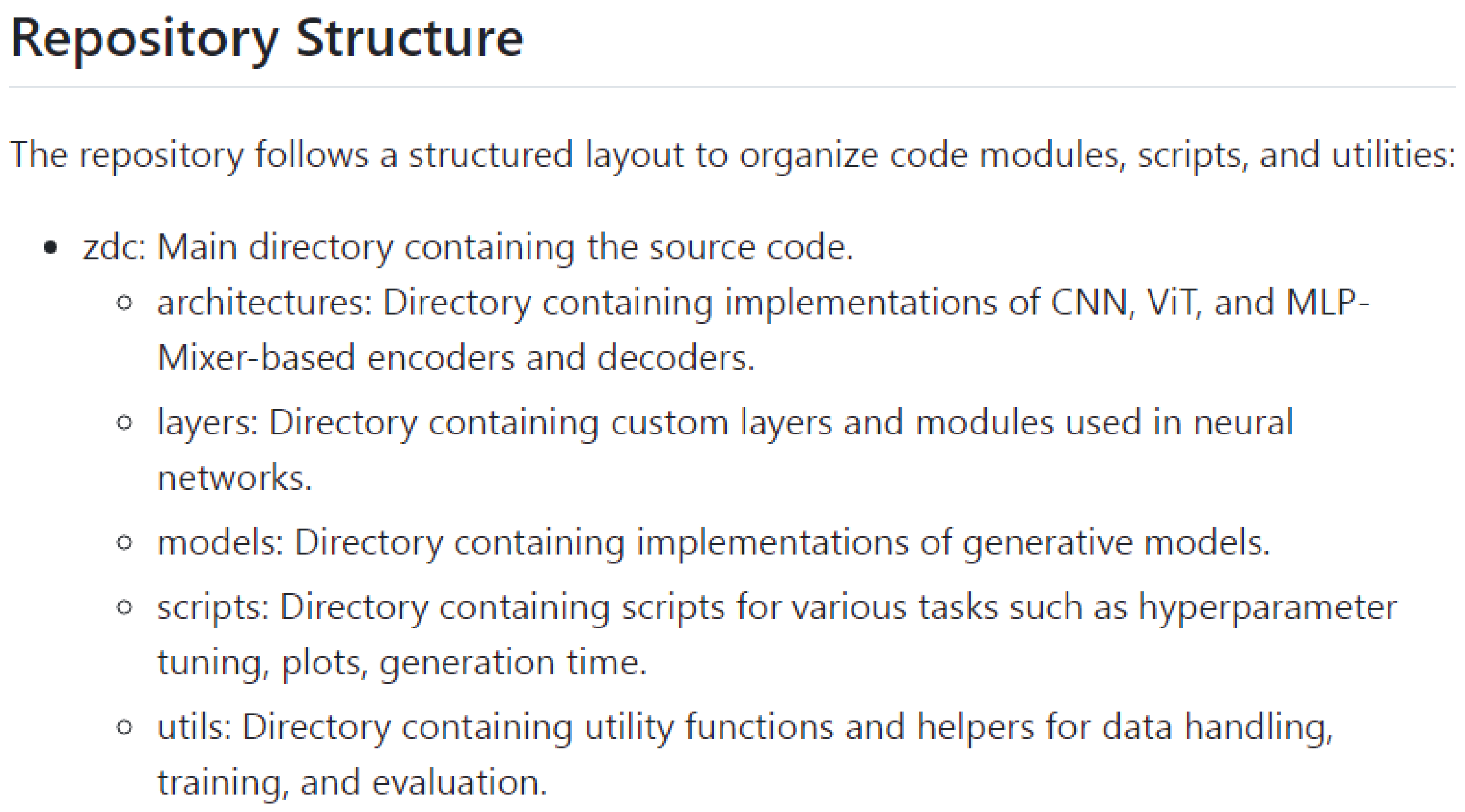


The findings of this thesis enable the formulation of several goals for future research that will further improve the fast ZDC simulation:

- Future research could focus on further refining VQ-GAN due to its balance between performance and throughput. The goal is to improve simulation fidelity by integrating the latest ViT developments [105], testing different sampling techniques, and evaluating new strategies for codebook updates [106].
- Diffusion, as a state-of-the-art model, should be the subject of additional research. The key goal is to improve the generation speed by operating in the latent space [56] and reducing the number of denoising steps. Investigating knowledge distillation [58] and rectified flows [62], which have shown promise in recent models, might be beneficial.
- Exploring approaches to incorporate physical loss terms [70] and regularize neural networks to produce physically consistent ZDC responses would also be advantageous.

Github Repository

https://github.com/m-wojnar/zdc



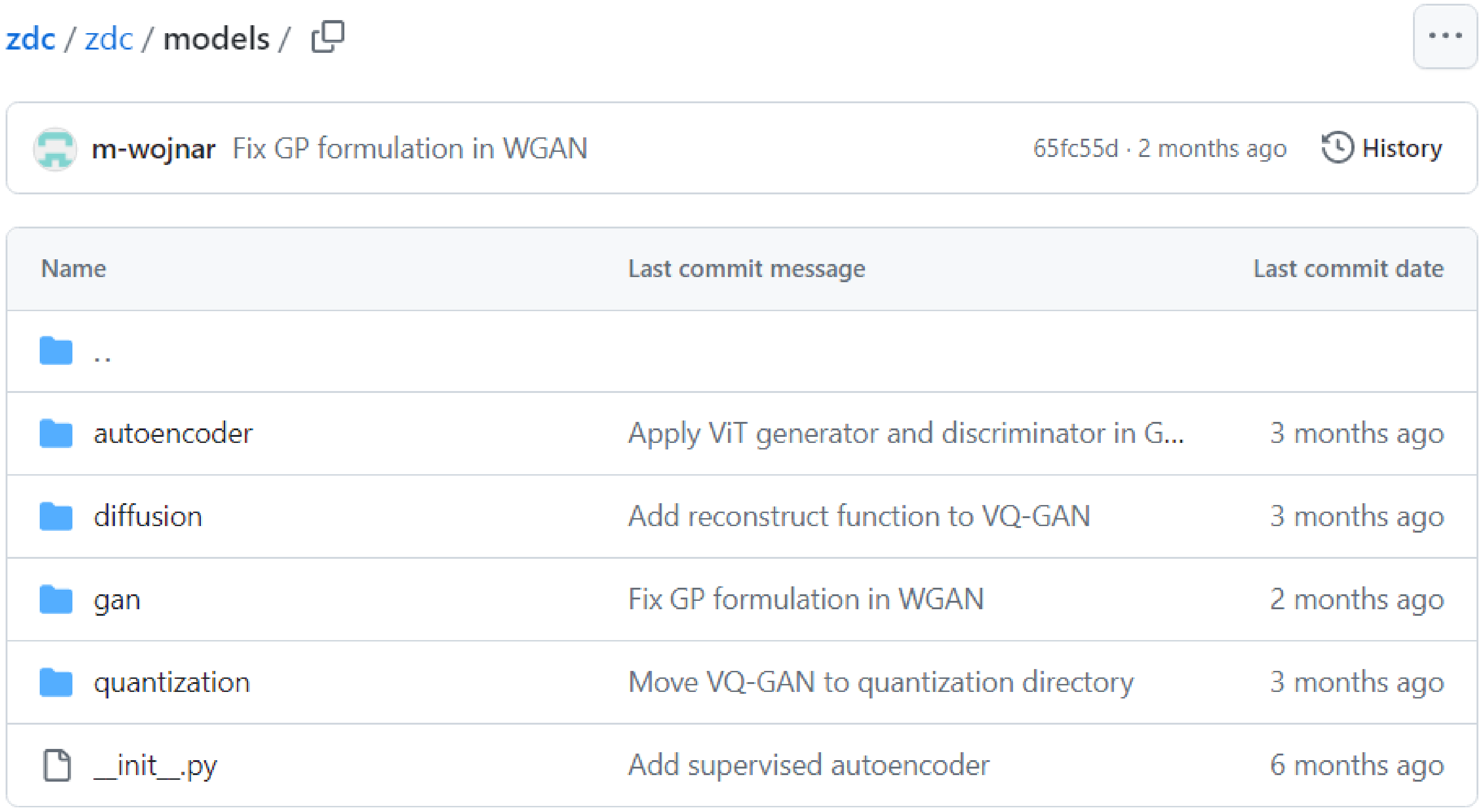
Repository Structure

The repository follows a structured layout to organize code modules, scripts, and utilities:

- zdc: Main directory containing the source code.
  - architectures: Directory containing implementations of CNN, ViT, and MLP-Mixer-based encoders and decoders.
  - layers: Directory containing custom layers and modules used in neural networks.
  - models: Directory containing implementations of generative models.
  - scripts: Directory containing scripts for various tasks such as hyperparameter tuning, plots, generation time.
  - utils: Directory containing utility functions and helpers for data handling, training, and evaluation.

Github Repository

https://github.com/m-wojnar/zdc



zdc / zdc / models /

m-wojnar Fix GP formulation in WGAN 65fc55d · 2 months ago History

Name	Last commit message	Last commit date
..		
autoencoder	Apply ViT generator and discriminator in G...	3 months ago
diffusion	Add reconstruct function to VQ-GAN	3 months ago
gan	Fix GP formulation in WGAN	2 months ago
quantization	Move VQ-GAN to quantization directory	3 months ago
__init__.py	Add supervised autoencoder	6 months ago