國立陽明交通大學
智慧系統與應用研究所
碩士論文
初稿

Institute of Intelligent Systems
National Yang Ming Chiao Tung University
Master Thesis

組合類別標籤到圖像擴散模型

Compositional Class label-to-image Diffusion Model

研究生 : 賴世倫 (Shih-Lun Lai)
指導教授 : 馬清文 (Ching-Wen Ma)

中華民國 一一三年五月
May 2024

組合類別標籤到圖像擴散模型

Compositional Class label-to-image Diffusion Model

研 究 生：賴世倫
指導教授：馬清文

Student: Shih-Lun Lai
Advisor: Dr.Ching-Wen Ma

國立陽明交通大學
智慧系統與應用研究所

碩士論文 初稿

A Thesis Draft
Submitted to Institute of Intelligent Systems
College of Artificial Intelligence
National Yang Ming Chiao Tung University
in partial Fulfilment of the Requirements
for the Degree of
Master
in
Intelligent Systems

May 2024

Taiwan, Republic of China

中華民國 一一三年五月

組合類別標籤到圖像擴散模型

學生：賴世倫　　指導教授：馬清文 教授

國立陽明交通大學 智慧系統與應用研究所

摘　　要

在一般的機器學習任務中，測試資料集通常和訓練資料集有同樣的分布，在圖像生成領域也是如此，生成模型往往只能產生跟訓練資料集具有同樣分布的資料，我們透過一個很簡單的例子，利用不同的方法去實現 unseen data generation，在現有的方式沒辦法做到的情況下，我們將 unseen data generation 進一步拆解成 unseen compositional image generation，並利用組合式標籤來引導生成模型。不同與以往只使用一種類別標籤來引導生成模型，我們的模型使用多種類別標籤來控制生成過程，像是屬性類別，物件類別等等，透過模型訓練學習到不同種類標籤的特徵，來實現組合式的零樣本圖像生成。

關於零樣本生成，大語言模型，如 GPT-4，與大型文生圖模型，如 DALLE-2，也具有類似零樣本生成之能力。其不同之處爲，我們研究在特定領域內實踐零樣本生成之可能性與必備條件，並以中小模型來完成。爲佐證組合式零樣本影像生成乃爲當代機器學習技術有潛力能完成之任務，我們設計一系列由簡單到複雜的組合式零樣本生成任務。在簡單的任務中，我們提出的方法已可準確生成新組合樣本。在較爲複雜的任務中，由我們的模型所生成之新組合樣本，經過挑選，亦可得到合理新樣本。

關鍵詞: 圖像生成、擴散模型、組合式零樣本圖像生成、組合式零樣本學習。

Compositional Class label-to-image Diffusion Model

Student: Shih-Lun Lai Advisor: Dr. Ching-Wen Ma

Institute of Intelligent Systems
National Yang Ming Chiao Tung University

Abstract

In typical machine learning tasks, the test dataset usually shares the same distribution as the training dataset. Similarly, in the domain of image generation, the generation models often produce data with the same distribution as the training dataset. We addressed this limitation by employing various methods to achieve unseen data generation using a simple example. When existing methods failed to accomplish this, we further decomposed unseen data generation into unseen compositional image generation. We guided the generation model using compositional class labels. Unlike traditional approaches that use a single class label to guide the generation process, our model uses multiple category labels, such as attribute categories and object categories, to control the generation process. Through training, the model learns features from different types of labels to achieve compositional zero-shot image generation.

Regarding zero-shot generation, large language models like GPT-4 and large-scale image generation models like DALL-E 2 also possess similar zero-shot generation capabilities. However, our research focuses on implementing zero-shot generation within specific domains and using smaller models. To demonstrate the potential and prerequisites of compositional zero-shot image generation as a task achievable with contemporary machine learning techniques, we designed a series of tasks ranging from simple to complex compositional zero-shot generation tasks. In the simpler tasks, our proposed method accurately generates new compositional samples. In the more complex tasks, the newly generated compositional samples from our model, when selected appropriately, also yield reasonable new samples.

Keyword: Image generation, Diffusion Model, Compositional zero-shot image generation, Compositional zero-shot learning

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter primarily aims to elucidate the motivation and objectives of our research. It provides a brief description on the proposed approach.

1.1 Background

In the realm of computer vision, the task of image synthesis plays a pivotal role in various applications, ranging from data augmentation for machine learning models to the generation of high-fidelity images for creative content creation. This field has gained significant traction with advancements in deep learning and generative models, there are several common ways of image synthesis, such as Generative Adversarial Network(GAN) (1), Variational AutoEncoders(VAEs) (2) or Diffusion Models.

Recently, diffusion models have achieved significant success in the field of image synthesis. Not only can they produce high-quality images, but they also exhibit excellent properties such as mode coverage and easy scalability. These models achieve the restoration of real images by gradually reducing the ratio of noise in the signal.

1.2 Motivation

1.2.1 Initial idea

In typical machine learning tasks, the test dataset usually has the same distribution as the training dataset. Figure shows a 2D point dataset with 24 clusters. During the testing phase, we aim to generate a complete set of 25 clusters. This includes generating data that does not belong to the distribution of the training dataset, and it cannot be randomly generated; it must be

generated based on the conditions we provide. We considered several methods for the scenario described above.

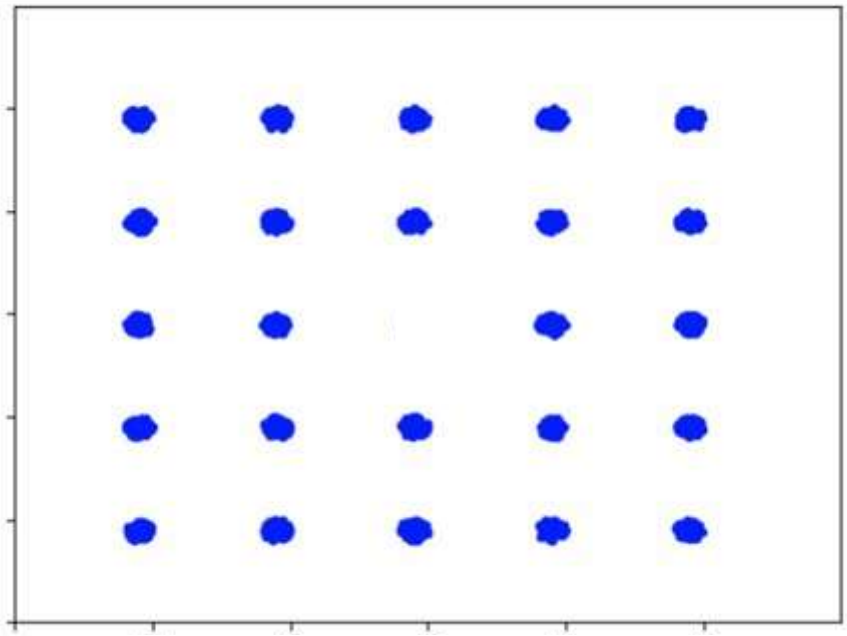

Figure 1.1: 2D points dataset with 24 clusters

1.2.2 Different approach

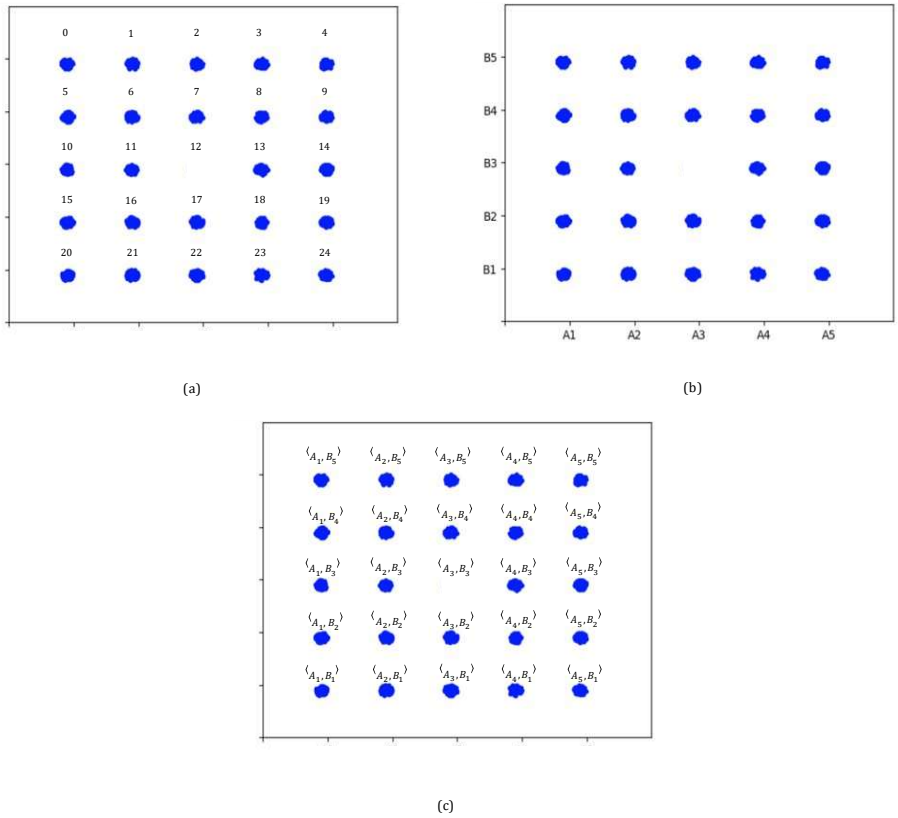

Figure 1.2: Illustration of different approach on 2D point dataset

1. **Directly labeling each cluster**: We annotate each cluster, with each cluster corresponding to a label, as shown in figure 1.2.(a). We trained a diffusion model conditioned on class labels $c = 0, 1, 2, ..., 25$ each label corresponds to a cluster. During the testing phase, we generate each clusters based on the class labels, cluster corresponding to label 12 is not present in the training data. We aim for the generation model to produce it during the testing phase..

2. **Compositional visual generation with energy based model** (3): We referenced the content of this paper to decompose the problem, approaching image generation from a compositional generation perspective. We annotated each row and column of clusters in the image, as shown in figure 1.2.(b) below, and trained different energy models based on the class labels $c = A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5$. In this example, we train ten

energy-based models. During the testing phase, we combine any two energy-based models to generate clusters. We aim for the combination of the energy-based models with labels A_3 and B_3 to generate the missing cluster in the center.

3. **Compositional visual generation with composable diffusion models**(4): We referenced the content of this paper to decompose the problem, approaching image generation from a compositional generation perspective. We annotated each row and column of clusters in the image, as shown in figure 1.2.(b) below, and trained different energy models based on the class labels $c = A_1, A_2, A_3, A_4, A_5, B_1, B_2, B_3, B_4, B_5$. In this example, we train ten diffusion models. During the testing phase, we combine any two diffusion models to generate clusters. We aim for the combination of the diffusion models with labels A_3 and B_3 to generate the missing cluster in the center.

4. **Compositional generation with compositional class label(ours)**: For each cluster in the image, we assigned a corresponding compositional class label composed of two class labels, as shown in figure 1.2.(c). We trained our model conditioned on compositional class labels $c, c =< A_1, B_1 >, < A_1, B_2 >, ..., < A_5, B_5 >$.During the testing phase, we aim for the cluster generated with the condition $< A_3, B_3 >$ to appear in the center of the plot.

1.2.3 Experiment results

We use MultiLayer Perceptron(MLP)(5) in our experiment. Detailed hyperparameters are provided in table 1.1.

	CCDM (MLP as denoising model)
Diffusion steps	50
Noise Schedule	linear
Number of Fully Connected Layer	3
Batch Size	512
Epochs	100
Learning Rate	2×10^{-4}
Embedded Dimension	128

Table 1.1: Hyperparameters for CCDM(without attention) trained on the 2D Point Dataset.

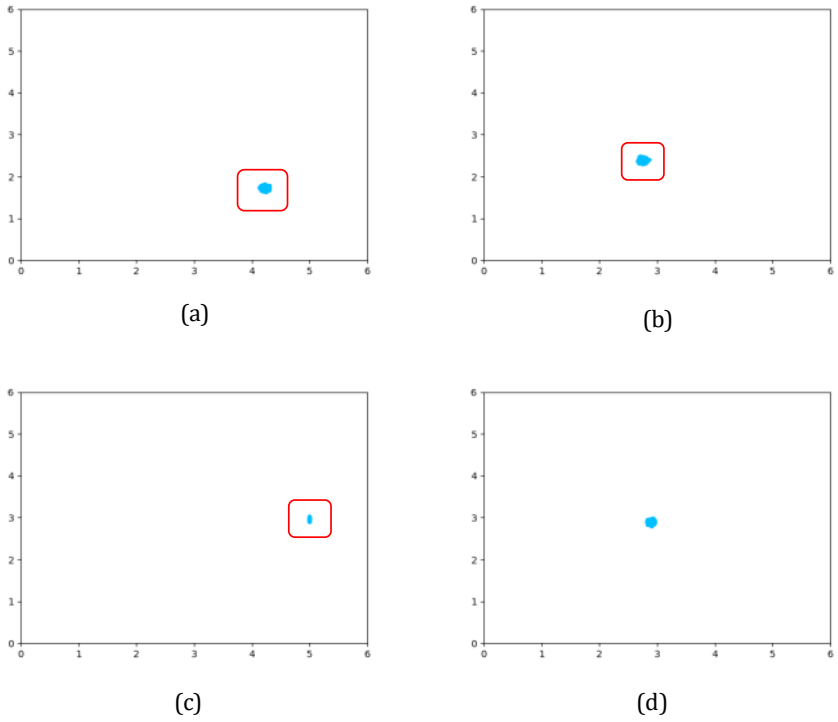

Figure 1.3: The unseen clusters generated by the four different methods mentioned above

As shown in figure 1.3，the unseen cluster generated from first three methods fail were not at the correct position and all shifted during the testing phase. On the contrary, our method can accurately generate the unseen cluster in the center. With this success on this toy dataset, we

have gained more confidence in the compositional class label-to-image diffusion model.

1.3 Goal

Let $C_1, C_2, ..., C_m$ be set of category labels for each condition, $|C_1| = N_1, |C_2| = N_2, |C_m| = N_m$. The set of all compositions $U, |U| = N_1 \times N_2 \times \cdots \times N_m, S \subset U$ is the set of available compositions. Our goal is using the available composition $c_i, c_i \in S$ to generate all compositions $c_j, c_j \in U$.

Our research aims to applying compositional cognitive abilities to image generation emphasizes "learning from old compositional concepts and generalizing to new compositional concepts", achieving compositional zero-shot image generation.

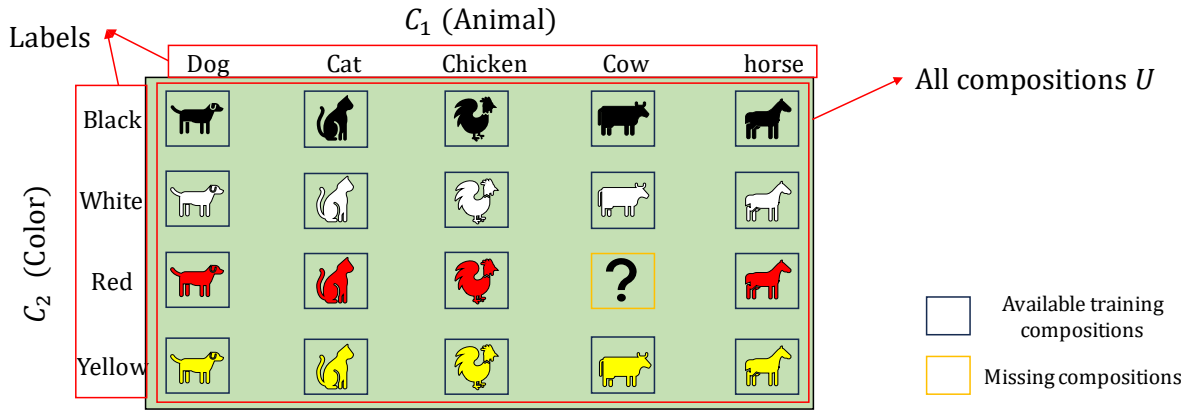

Figure 1.4: Illustraion of Compositional Zero-shot image generation

As illustrate in figure1.4 , C_1 represents the species of animals, and C_2 represents colors. They respectively have the categories "Dog", "Cat", "Chicken", "Cow", "Horse", "Black", " White", "Red", and "Yellow". In total, there can be 20 compositions formed from these categories. However, our training dataset only contains 19 of these compositions. Our goal is to generate all compositions using the limited training compositions.

1.4 Contribution

The contributions of this paper are summarized in the followings:

- We propose a novel method for class label-to-Image generation, utilizing various categories of class labels and employing compositional class labels to generate the desired images. This enhances the controllability of the Diffusion model when using class labels as conditions, enabling the generation of more precise images and even producing images that have never been seen in the original dataset.

- On the application scenario, our model utilizes multiple categories of class labels to generate images. Compared to models that generate images from text, the process of image generation is simpler and faster. There is no need for meticulous design of inputs to produce the desired images. Additionally, our approach eliminates the necessity for large pretrained language models and the substantial amount of data and data preprocessing typically required for training language models, such as prompt engineering.

- Our method offers an efficient way to expand and enrich the original dataset, addressing issues related to data imbalance or absence. Since our model is trained using the original dataset, it effectively addresses the augmentation of the dataset by generating various combinations of data, filling gaps in missing or underrepresented classes. When utilizing compositional class labels as inputs, the generated images exhibit diversity, preventing an overly uniform representation and promoting a more varied dataset. Importantly, this approach avoids the generation of out-of-distribution data. Following the generation of composite images, a classification model is employed to filter the images. This framework serves as an efficient and effective way to expand and enhance the original dataset.

The rest of this thesis is organized as follows. Chapter 2 briefly describes related work. Chapter 3 describes more details about diffusion models. Chapter 4 details the proposed method and architecture. Chapter 5 shows our experiment results. Chapter 6 concludes this thesis.

Chapter 2

Related Work

2.1 Diffusion Model on Image Synthesis

In the field of image synthesis, diffusion models [(6), (7), (8), (9)] have achieved state-of-the-art performance. They have broken the long-standing dominance of GAN (1) in image synthesis domain. Diffusion model have demonstrated outstanding performance in various domain of image synthesis, from the unconditional diffusion models (6) to conditional diffusion models (10). Diffusion models have a wide range of applications, such as Image Super-Resolution [(11), (12), (13)], Inpainting [(13), (14)], Image translation [(15), (14)] and Editing [(16)]. These methods all implement the diffusion model, employing different conditional inputs and modifications to the architecture based on the conditional input. This allows the diffusion model to generate corresponding images based on different conditions. Text-to-Image generation is currently one of the most frequently discussed applications of diffusion models. Diffusion models need to generate corresponding image from a descriptive text. (13) utilized a pretrained text encoder CLIP (17) and used cross-attention to combine text information with diffusion model, generating real and diverse image based on text description. On the other hand, DALLE-2 (18) proposed a two-state approach, involving a prior model capable of generating a CLIP-based image embedding based on a text caption, and a diffusion-based decoder that can generate an image conditioned on the produced image embedding. Imagen (19) employed a different text encoder T5 (20) which is a text-to-text transformer, they explored that a large language model trained on only text data is still very effective on text-to-image generation. In order to train with high-resolution images, (13) shifted the diffusion process to latent space to reduce computation resources. They first employed a pretrained autoencoder to encode images into latent space, then the diffusion process and the sampling process are conducting in the latent spaces. This method effectively

reduces computational resources and can be applied to various tasks.

2.2 Classifier-free guidance

The original diffusion models were unconditional, meaning they couldn't be controlled by text or categories to generate images.(10) proposed the classifier guidance method. During the training of the diffusion model, an additional classifier is trained on noisy samples. When sampling images, the diffusion model can be guided and generate images according to the class by leveraging the gradients from the classifier. This allows the diffusion model to generate images based on the specified class. This method need to train an additional classifier, the sampling quality is highly correlated with the classifier. (21) proposed the proposed classifier-free guidance, allowing the diffusion model to generate images according to categories without the need for an additional classifier. By combining unconditional input and conditional input with a guidance scale ω, the diffusion model can generate images based on categories without relying on an additional classifier.

2.3 Compositional Zero-shot Learning (CZSL)

Traditional Zero-Shot Learning (ZSL) [(22), (23), (24), (25), (26), (27)] investigates classification tasks when there are no training samples available, but the semantics of new classes are known. One branch of research within ZSL is Compositional Zero-Shot Learning (CZSL) [(28), (29), (30), (31)]. Given a known object concept, people can combine combine this object with various attribute to create different compositions. CZSL represents all attributes and objects as concept are available during training phase, the model needs to recognize the unseen compositions based on seen concepts and correctly classify them during validation or testing. (32) trained a linear classifier of seen compositions and inferred unseen compositions form observed classifier using Bayesian framework. (33) propose a simple approach which decomposed a complex concepts into multiple primitives and trained a binary linear classifier on each of these primitive. This method demonstrates our ability to infer a complex concept composed of two or more

primitives, which motivate us on image selection framework. (34) propose a model based on the concept of 'attributes as operators,' representing attribute-object composition. This model conditions transformations on attributes, incorporating it into an embedding learning model for identifying unseen compositions. (29) create a model that learn three spaces for the object, attribute and composition classifications.

Chapter 3

Preliminary

In this chapter, we will explain the process and mathematical principles of the denoising diffusion probabilistic model (diffusion model for short) (6), providing a basic understanding of diffusion models for better understanding the rest of this thesis.

3.1 Diffusion model

Diffusion model is a class of generative models that simulate a stochastic diffusion process where a simple initial distribution is transformed into a more complex distribution that represents the data distribution. They define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise. Given a data point sampled from a real data distribution $x_0 \sim q(x)$, we add small amount of Gaussian noise ϵ to the original sample in T steps, where T is the total diffusion steps, $T \in \{1, 2, ..., T\}$, this produce a sequence of noisy sample $x_1, x_2, ..., x_T$ with different noise level and they all share the same dimension as x_0. The posterior $q(x_1, ..., x_t|x_0)$, also known as the **forward diffusion process**, which converts the original data distribution $q(x_0)$ into a tractable prior distribution $q(x_t)$, typically Gaussian distribution. This process is set as a Markov chain, we can factorize the posterior as follows:

$$q(x_1, ..., x_t|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}), \tag{3.1}$$

where $\beta_t \in (0, 1)$ is the variance schedule controlling the step size and can be regarded as a constant hyperparameter. Let $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, then x_t at arbitrary timestep t can

be expressed in a closed form:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathbf{I}), \tag{3.2}$$

Equation 3.2 can be further reparameterized as

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{3.3}$$

The **reverse diffusion process** gradually remove noise from noisy sample x_t, producing a slightly denoised sample x_{t-1}, since we cannot estimate $q(x_{t-1}|x_t)$, we need to learn a model p_θ to approximate $q(x_{t-1}|x_t)$.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3.4}$$

Here, θ denotes the model parameters and the mean $\mu_\theta(x_t, t)$ and the variance are modeled using deep neural networks. With this "denoising model", we can generate a data sample x_0 from a noise sample x_t $q(x_t)$ sampled from prior distribution.

When training the denoising model, (6) try to maximize the variation lower bound (ELBO) on negative log-likelihood and utilize the Kullback-Leibler (KL) divergence.

$$\begin{aligned}\mathbb{E}_q[-\log p_\theta(x_0] \leq L \coloneqq \underbrace{E_q[D_{KL}(q(x_T|x_0)||p_\theta(x_T))}_{L_T} + \\ \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{L_{t-1}} - \underbrace{\log p_\theta(x_0|x_1)]}_{L_0}\end{aligned} \tag{3.5}$$

L_T can be considered as a constant since q does not have any parameters and x_T is a Gaussian noise. L_t can be considered as bringing two Gaussian distributions: $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$ and $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, x_0), \Sigma_\theta)$ closer. Let:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha}_t(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}, \tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t \tag{3.6}$$

Based on the solving the KL divergence on the multivariate Gaussian distribution, L_{t-1} can be rewritten as:

$$L_{t-1} = \mathbb{E}_q[\frac{1}{2\|\Sigma_\theta(x_t,t)\|_2^2}\|\tilde{\mu}_t(x_t,x_0) - \mu_\theta(x_t,t)\|^2] + C \tag{3.7}$$

where C is a constant, $\mu_\theta(x_t, x_0)$ is what we train to predict $\tilde{\mu}_t(x_t, x_0)$.

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t,t)) \tag{3.8}$$

ϵ_θ is the denoising model intended to predict ϵ_t from x_t. Since x_t is available at training phase, we can try predict Gaussian noise term ϵ_t from the input x_t as time step t instead of predicting $\tilde{\mu}_t(x_t, x_0)$. The training objective of the denoising model becomes to predict the noise vector ϵ_t given noisy sample x_t and timestep t. Bring Eq.(3.6), Eq.(3.8) into Eq.(3.7), Eq. (3.7) can be simplified to:

$$L_{t-1} = \mathbb{E}_{x_0,\epsilon}[\frac{\beta_t^2}{2\alpha_t(1-\overline{\alpha}_t\|\Sigma_\theta\|_2^2}\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t)\|^2] \tag{3.9}$$

Through experiment results, (6) discovered that beneficial to sample quality to train on a simpler objective function that ignored the weighting term:

$$L_{simple} = \mathbb{E}_{x_0,\epsilon}[\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t)\|^2] \tag{3.10}$$

Note that (6) does not explicitly consider the Σ_θ in both training and inference;instead, it is set to β_t or $\tilde{\beta}_t$. They found that predict Σ_θ could lead to training instability.

3.2 Denoising Diffusion Implicit Models

In practical applications, achieving high-quality images requires a larger timestep T, which can result in a time-consuming process for reverse diffusion process. To accelerate the reverse diffusion process, (35) introduced an approach that sacrifices diversity in exchange for faster

inference. They discovered that the objective loss function in DDPM only relies on the marginal distribution $q(x_t|x_0)$, rather than on the joint distribution $q(x_{1:T}|x_0)$. They choose to use an alternative non-Markovian forward process, indexed by a real vector $\sigma \in \mathbb{R}^{\mathbb{T}}$ the forward process is defined as:

$$q_\sigma(x_{1:T}|x_0) = q_\sigma(x_T|x_0)\prod_{t=2}^{T} q_\sigma(x_{t-1}|x_t, x_0) \tag{3.11}$$

where $q_\sigma(x_T|x_0) = \mathcal{N}(\sqrt{\alpha_T}x_0, (1-\alpha_T)\mathbf{I})$ for all $t > 1$,

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \sqrt{\alpha_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 I) \tag{3.12}$$

The forward process can be derived from Bayes' rule:

$$q_\sigma(x_t|x_{t-1}, x_0) = \frac{q_\sigma(x_{t-1}|x_t, x_0)q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)} \tag{3.13}$$

In the reverse diffusion process, we can use following equation to generate x_{t-1} from x_t

$$x_{t-1} = \sqrt{\alpha_{t-1}}(\underbrace{\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}}}_{\text{predicted } x_0}) + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(x_t, t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}} \tag{3.14}$$

Let $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$, we can consider η as a hyperparameter to control the sampling stochasticity. When $\eta = 0$, the forward process becomes deterministic, the reverse process becomes a fixed procedure while the random noise term in Eq. 3.14 becomes zero.

Because DDIM has no specific forward process, we can use a forward process with a smaller number of steps to accelerate the model's generation process. We sample a subsequence $[\tau_1, \ldots, \tau_S]$ with length S from the original sequence $[1, \ldots, T]$, then:

$$q(x_{\tau_i}|x_0) = \mathcal{N}(\sqrt{\alpha_{\tau_i}}x_0, (1-\alpha_{\tau_i})\mathbf{I}) \tag{3.15}$$

The reverse process now generates samples according to τ, because the length S is much smaller than T, we can significantly accelerate the generation process.

In the following chapters, we will utilize the above equations to elucidate our approach.

Chapter 4

Proposed Method

4.1 Compositional Class label-to-Image Diffusion model (CCDM)

Our model is a diffusion model conditioned on a compositional class label c, where $c = [c_1, \ldots, c_n]$ is composed of a combination of labels from different categories. We use $n = 2$ and $n = 3$(two conditions and three conditions) for our model. In this chapter, we further decomposed our model into several components to introduce our approach: the denoising model , the conditioning module, and the compositional class encoder. As mentioned in Section 3.1, the denoising model is used to predict noise based on a compositional class label c, a timestep t and a noisy sample x_t, with this model, we can use Eq. 3.14 to restore x_t to x_0. The conditioning module is how we add conditional information into our denoising model, we employ four different architectures to integrate the information of compositional class labels. The compositional class encoder is used to encode the compositional class label c, mapping the class label to a higher-dimensional space.

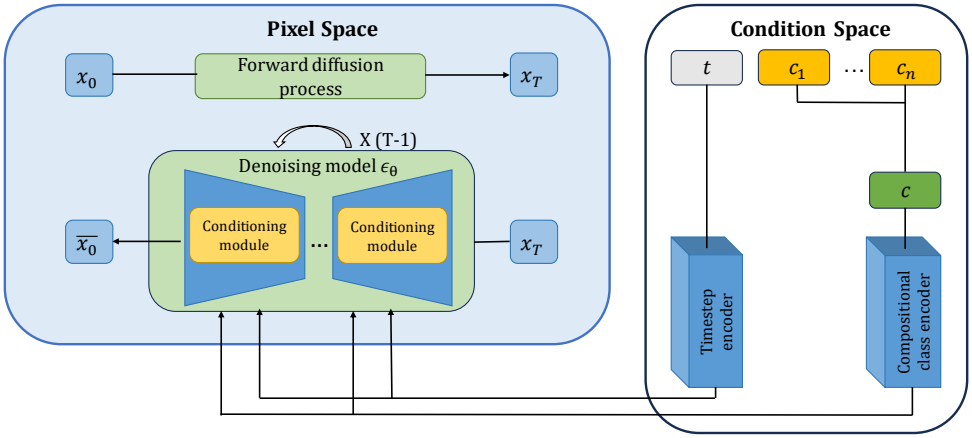

Figure 4.1: Block diagram of CCDM

Figure 4.1 illustrates the architecture of our model, the compositional class label will be

encoded by the class encoder and integrate with the conditioned denoising model through the conditioner, and the conditioned denoising model will predict the noise term based on conditional information.

4.2 Compositional class encoder

In order to guide the diffusion model with compositional class information, we first use one-hot encoding to encode the compositional class label for different categories. Given $c = (c_1, c_2, c_3, ...)$ is the compositional class label, the compositional class encoder is illustrated as the follows:

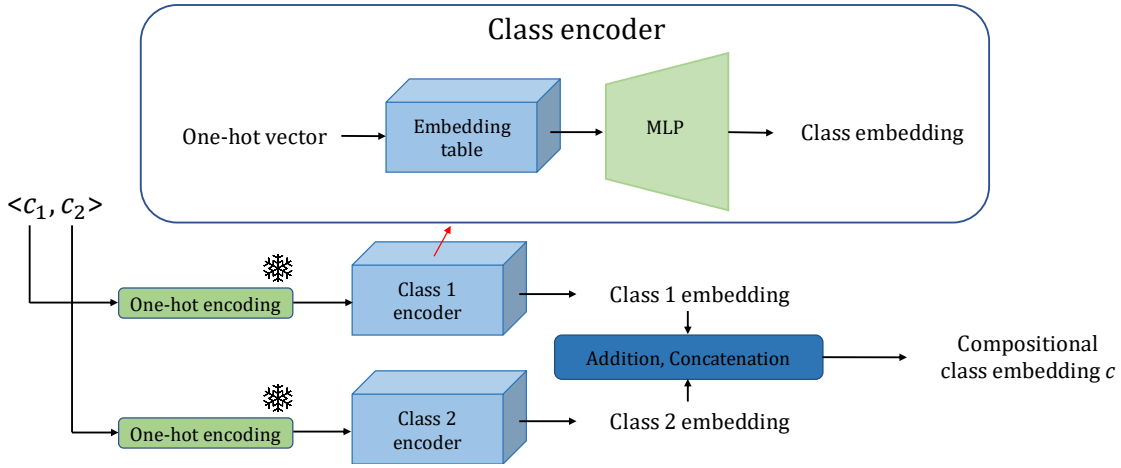

Figure 4.2: Compositional class encoder

For each category's class label, we first encode them using one-hot encoding. Then, we use an embedding table and MLP to encode the one-hot vector. Finally, we sum or concatenate these class embeddings based on different conditioning modules. Figure 4.2 illustrates how we encode compositional class labels using two category class labels.

4.3 Denoising model

According to Chapter 3, we added a compositional class label c to guide our denoising model. We modify Eq.3.10 to cooperate with compositional class label c and sampled image according

to the compositional label c. The modified loss function will be:

$$L_{simple} = \mathbb{E}_{x_0,\epsilon}[\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t, c)\|^2] \tag{4.1}$$

Our denoising model ϵ_θ now has three inputs, which are the noisy sample x_t, timestep t and the compositional label c instead of only x_t and t. The training and sampling procedure are illustrated as the follows:

Algorithm 4.1 Training CCDM

repeat:
  $(x_0, c) \sim p(x, c)$
  $c \leftarrow \varnothing$ with probability p_{uncond}
  $t \sim Uniform(\{1, \ldots, T\})$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon$
  Take a gradient step on $\nabla_\theta\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t, c)\|^2$
until converged

Algorithm 4.2 Sampling

Require: ω : guidance scale
Require: c : Compositional class label for compositional generation
Require: η : hyperparameter controlling sampling stochasticity

$x_t \sim \mathcal{N}(0, \mathbf{I})$
for t = $S, \ldots, 1$ **do**
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $\epsilon = 0$
  $\sigma_t = \eta \cdot \tilde{\beta}_t$
  $\tilde{\epsilon}_t = (1+\omega)\ \underbrace{\epsilon_\theta(x_t, t, c)}_{\text{conditional noise}}\ -\omega\ \underbrace{\epsilon_\theta(x_t, t, \varnothing)}_{\text{unconditional noise}}$
  $x_{t-1} = \sqrt{\alpha_{t-1}}(\frac{x_t - \sqrt{1-\alpha_t}\tilde{\epsilon}_t}{\sqrt{\alpha_t}}) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\tilde{\epsilon}_t + \sigma_t\epsilon_t$
end for
return $\overline{x_0}$

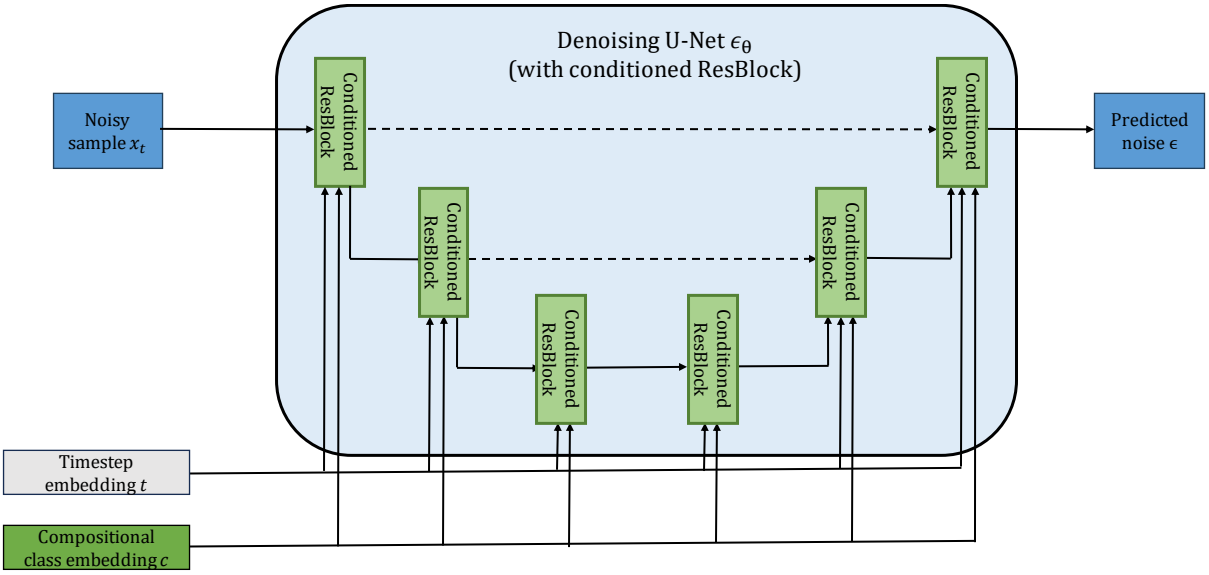

Figure 4.3: Conditioned denoising U-Net with conditioned ResBlock

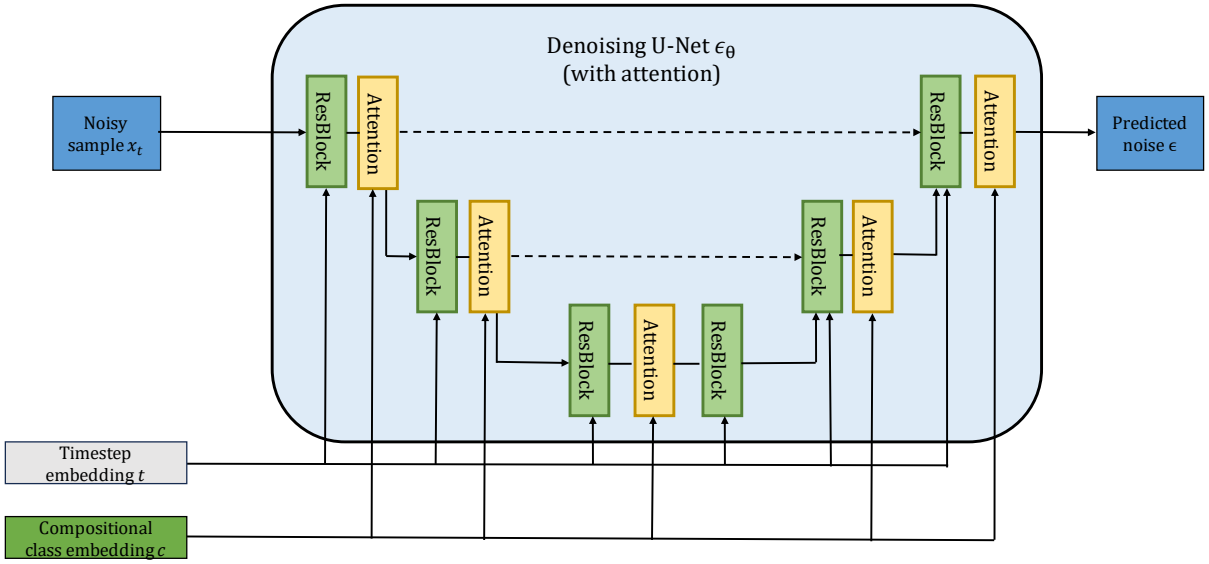

Figure 4.4: Conditioned denoising U-Net with attention

Figure 4.3 and figure 4.4 shows how to integrate conditional information with each residual block in the conditioned denoising U-Net. The difference between these two U-Net architectures lies in the location where condition information is incorporated.

4.4 Conditioning module

There are various methods to incorporate conditional information into the denoising model. Drawing inspiration from numerous papers, we implemented four different conditioning modules in our model. In the following schematic diagram, we will illustrate how we use the information from compositional class labels to guide our model. Before introducing these conditioning modules, we need to establish definitions for some symbols. Let h be the hidden state, t be the timestep embedding vector, c be the compositional class embedding vector and $proj$ be the linear projection layer.

4.4.1 Addition (Add)

Follow DDPM(6) implementation, we preform element-wise addition on projected timestep embedding and compositional class embeddings into each residual block. We define the conditioning module as Add(h, t, c), then

$$\text{Add}(h, t, c) = h + proj(t) + proj(c) \tag{4.2}$$

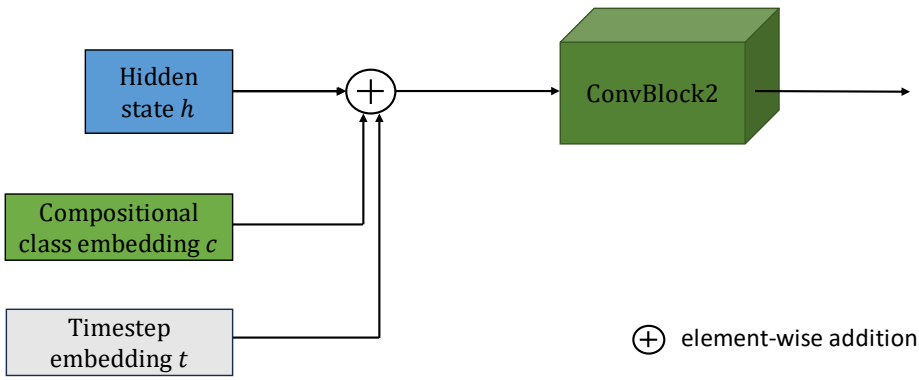

Figure 4.5: Addition

Figure 4.5 shows how we integrate hidden state h and conditional information. In each residual block, we add projected timestep embedding and projected compositional class embedding into h, where h is the output of the first convolution block. We call this conditioning module

Add for the rest of this thesis.

4.4.2 Adaptive Group Normalization (AdaGN)

Follow (36) implementation, we also implement this module on our model, let $y' = proj(y + t)$ and $y' = [y_s, y_b]$, which we split y' into two vectors with the same dimension, then the conditioning module is illustrated as the follows:

$$\text{AdaGN}(h, t, c) = (1 + c_s)\text{GroupNorm}(h) + c_b \tag{4.3}$$

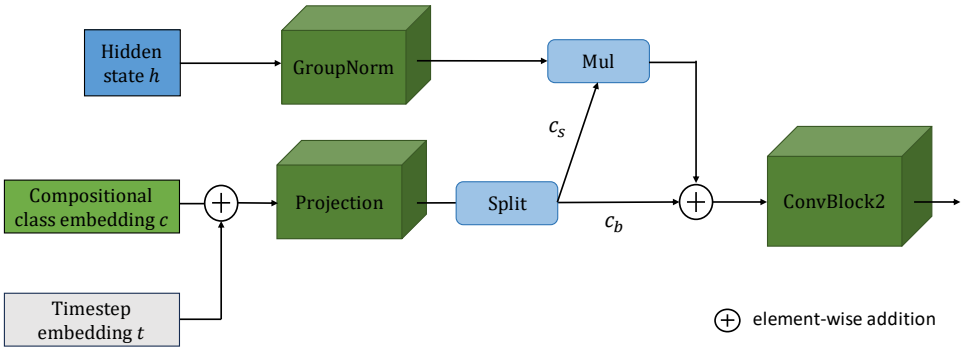

Figure 4.6: Adaptive Group Normalization

Figure 4.6 illustrates the procedure of adaptive group normalization, the projection layer maps the original dimension, which is 512, to 512 * 2, while the split operation divides the vector, originally of dimension 1024, into two 512-dimensional vectors.

4.4.3 Image Class concatenation (IC)

We concatenate compositional class embedding y with hidden state h to add conditional information guidance, then we use a bottleneck layer to reduce dimension after concatenation, details are illustrated as the follows:

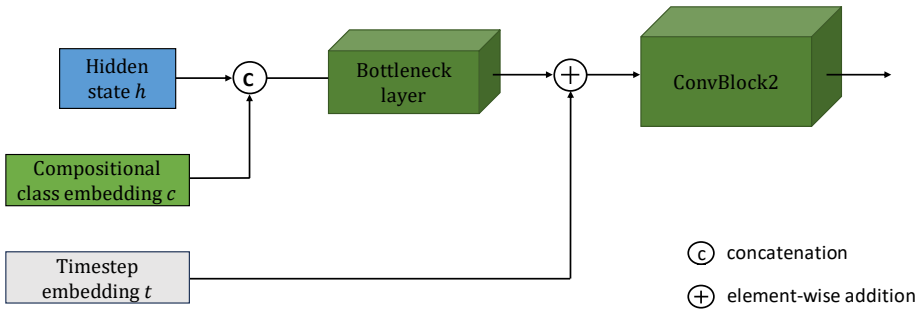

Figure 4.7: Image class concatenation

Follow (13) implementation of super-resolution and inpainting diffusion model. We concatenate concatenate compositional class embedding c with hidden state h in diffusion process.Figure 4.7 illustrates how we integrate timestep embedding t and compositional class embedding y with hidden state h. We first combine the hidden state and compositional class information using concatenation. Then, we use a bottleneck layer for dimensionality reduction, restoring it to the original dimension. Finally, we incorporate the timestep information through element-wise addition.

4.4.4 Cross Attention (CA)

Follow (13) implementation, we use an shallow spatial transformer composed by N transformer blocks, each block consists of a multi-head self-attention layer, a position-wise feedforward network and a multi-head cross-attention layer.

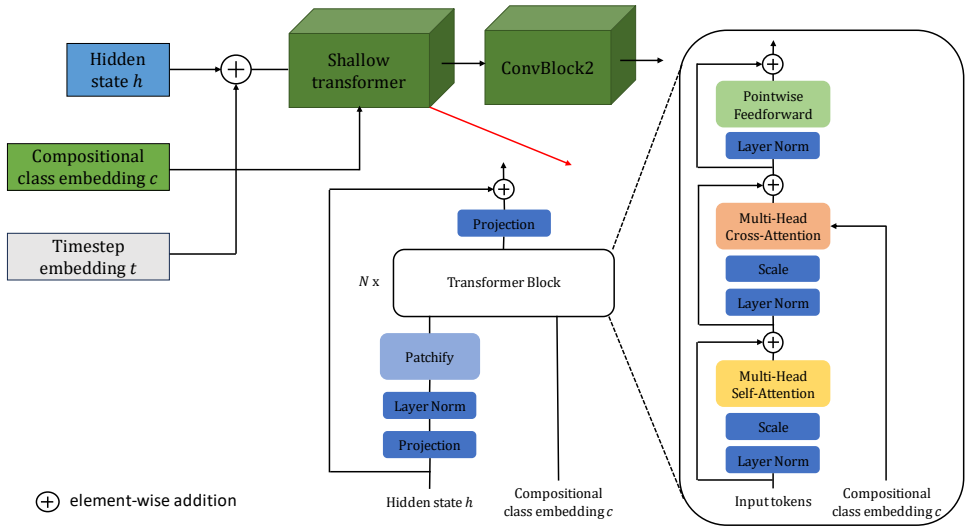

Figure 4.8: Cross attention

Figure 4.8 illustrates the architecture of spatial transformer and transformer block, note that the compositional class embedding $y \in \mathbb{R}^{B\times 1\times D_{emb}}$ is extended from the original compositional class embedding $y \in \mathbb{R}^{B\times D_{emb}}$ by adding a new dimension.

4.5 Image Selector

Although CCDM has the capability of compositional zero-shot image generation, not every generated image aligns perfectly with the unseen compositional class label. Sometimes, the images produced by the model may exhibit deviations. Therefore, we require a framework to assess the performance of the model.

We introduced an image selection framework to assess the model. Following (33), we separately trained binary classifiers for the object and attribute of unseen compositions. Using "Heel Slipper" as an example, we would train a "Heel" binary classifier and a "Slipper" binary classifier. These classifiers are then utilized to filter the images generated by CCDM. To ensure the reliability of these classifiers, we first generated N unseen composition images using CCDM. Subsequently, we utilized these classifiers to select images that match the unseen compositions. We then manually verified whether the selected images indeed matched the criteria.

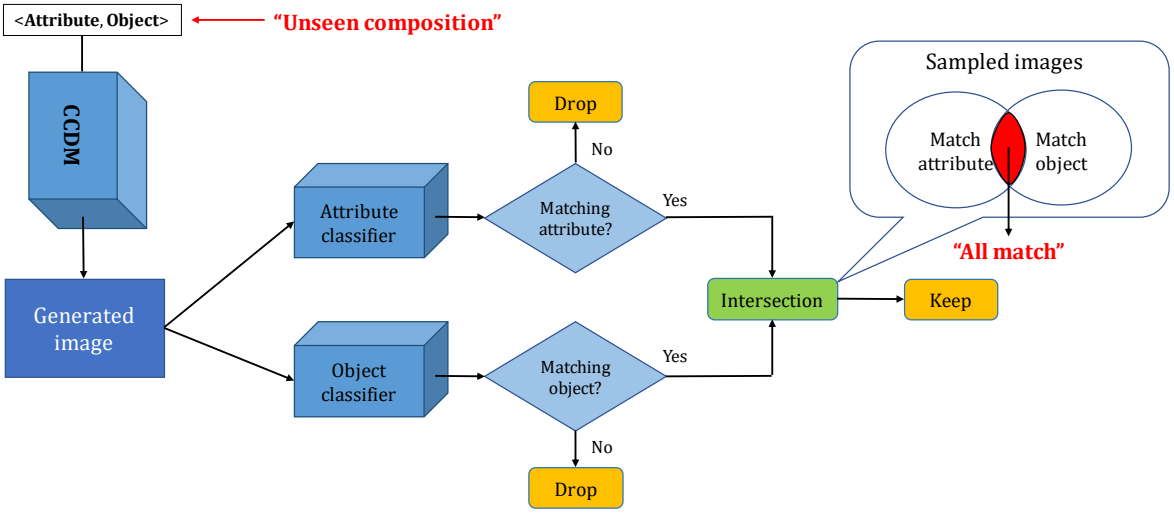

Figure 4.9: Image selector on unseen composition

Figure 4.9 illustrates how we evaluate a generated image. After generating images using CCDM based on unseen compositional class labels, each generated image undergoes evaluation by multiple binary classifiers to determine whether it matches the description of the class label.

Chapter 5

Experiment Results

5.1 Dataset preparation

For conducting experiments on Compositional zero-shot image generation, we utilized several datasets, including publicly available datasets and ones we generated ourselves. In the initial stages to validate the feasibility of our ideas, we started with a very simple dataset that we generated ourselves. After achieving success on this dataset, we proceeded to use publicly available datasets. In the following sections, we will provide detailed introductions for each dataset.

5.1.1 Shape Color Dataset

This dataset is generated by the python OpenCV library, includes various geometric shapes of different colors.

- **Two condition setting:** We use geometric shapes as the object category. There are a total of 6 object classes: Circle, Oval, Triangle, Rectangle, Pentagon, and Hexagon. We use color as the attribute category, with a total of 6 attribute classes: Red, Green, Blue, Yellow, Purple, and Black, resulting in a total of 36 compositions, As illustrated in table 5.1. Each of them has 1000 training samples. Each training sample is a 64 * 64 pixel image, the size and the location of the geometric object are randomized.

	Condition 1 (Color)	Condition 2 (Geometric shape)
Labels	Circle, Hexagon, Oval, Pentagon, Rectangle, Triangle	Black, Blue, Green, Purple, Red, Yellow

Table 5.1: Two condition setting of Shape Color dataset

- **Three condition setting:** Continuing with the two-condition setting, we use the size of the objects as a new category, with a total of 3 classes: Big, Medium, Small. This results

in a total of 108 compositions, As illustrated in table 5.4. Each of them has 1000 training samples. Each training sample is a 64 * 64 pixel image.

	Condition 1 (Size)	Condition 2 (Color)	Condition 3 (Geometric shape)
Labels	Big, Medium, Small	Circle, Hexagon, Oval, Pentagon, Rectangle, Triangle	Black, Blue, Green, Purple, Red, Yellow

Table 5.2: Three condition setting of Shape Color dataset

5.1.2 UT-Zappos50K

The UT-Zappos50K dataset comprises a total of 50,025 images of footwear, categorized into four main types: Shoes, Boots, Slippers, and Sandals. These images are further divided based on their functional types. We have restructured the dataset and removed some ambiguous data in the dataset for our experiments, leaving us with 47,917 images for training.

- **Two condition setting:** We retain the original four major categories as object category, introducing a new attribute category based on the presence or absence of heels, dividing it into two attribute classes: Heel and Flat. This results in a total of 8 compositions. As illustrated in table 5.3.

	Condition 1 (Shoe height)	Condition 2 (Shoe type)
Labels	Heel, Flat	Boot, Shoe, Sandal, Slipper

Table 5.3: Two condition setting of UT-Zappo50K dataset

- **Three condition setting:** The images in the dataset were initially captured from a consistent perspective, so we horizontally flipped half of the data to introduce different shooting angles as a new category, with a total of 2 classes: Left, Right. Continuing with the two-condition setting, resulting a total of 16 compositions. As illustrated in table 5.4.

	Condition 1 (Filming angle)	Condition 2 (Shoe height)	Condition 3 (Shoe type)
Labels	Left, Right	Heel, Flat	Boot, Shoe, Sandal, Slipper

Table 5.4: Three condition setting of UT-Zappo50K dataset

5.1.3 CelebFaces Attributes Dataset

CelebFaces Attributes Dataset(CelebA) is a large-scale face attributes dataset with 202599 face images, each with 40 binray attribute annotations.

- **Two condition setting:**We use two of these attributes as our object category: Male and Female. For the attribute category, we use 4 of these attributes: Black Hair, Brown Hair, Blond Hair, Gray Hair. This results in a total of 8 compositions. As illustrated in table 5.5.

	Condition 1 (Hair color)	Condition 2 (Gender)
Labels	Black hair, Gray hair, Blond hair, Brown hair	Male, Female

Table 5.5: Two condition setting of CelebA dataset

- **Three condition setting:**Continuing with the two-condition setting, we use hair style as a new category, dividing it into 2 classes: Wavy Hair and Straight Hair. This results in a total of 16 compositions. As illustrated in table 5.6

	Condition 1 (Hair style)	Condition 2 (Hair color)	Condition 3 (Gender)
Labels	Straight hair, Wavy hair	Black hair, Gray hair, Blond hair, Brown hair	Male, Female

Table 5.6: Three condition setting of CelebA dataset

5.2 Experiment setup

5.2.1 Training Diffusion model

To train our model, we refer to the common hyperparameter settings of the stable diffusion model to configure the hyperparameters of our model. We used the following default hyperparameters: the training epoch was set to 100, the initial learning rate was set to 6×10^{-5} with 10 warm-up epochs and the cosine learning rate scheduler was used for more stable training. For the optimization, we used AdamW optimizer with weight dacay values 10^{-4}.

Diffusion process related hyperparameters are descride as follows. For the forward diffusion process, we set $T = 50$ for experiments with 2D Point Dataset, for the rest of experiments, we set $T = 1000$.For all experiments, the variance β_T was set to 10^{-4} increasing linearly to 0.02. For the reverse process, we used classifier-free DDIM to sample images, the guildance strength ω to 1.8, η was set to 0 and the samples were generated in 100 steps.

As for the hyperparameters related to the denoising models and the conditioner, they are described as the follows: For the denoising model used in 2D Point Dataset related experiments, we implemented the denoising model as a MultiLayer Perceptron (MLP) with three fully connected layers. For rest of the experiments, we implemented U-Net as our denoising model, we set the number of residual blocks to 2 in each downsample or upsample block. We set the base channel to 128 and the channel multiplier to [1, 2, 2, 2]. For the conditioner of cross attention , we used the same architecture in stable diffusion, which is a unmasked transformer, the number of attention head was set to 4 and the attention resolutions were set to [32, 16, 8].

The training images were all resized to 64 x 64 with RGB channels. All experiments were conducted on two NVIDIA Tesla V100 GPU. The hyperparameter configuration for each architecture are illustrated with the following table.

	CCDM (with Conditioned ResBlock)
Diffusion steps	1000
Noise Schedule	linear
Base Channels	128
Number of Residual Blocks	2
Channel Multiplier	1,2,2,2
Batch Size	64
Epochs	100
Learning Rate	6×10^{-5}
Embedded Dimension	512

Table 5.7: Hyperparameters for CCDM(without attention) trained on the Shape Color Dataset, UT-Zappo50K, CelebA.

	CCDM (with attention)
Diffusion steps	1000
Noise Schedule	linear
Base Channels	128
Number of Residual Blocks	2
Channel Multiplier	1,2,2,2
Batch Size	64
Epochs	100
Learning Rate	6×10^{-5}
Embedded Dimension	512
Number of Attention Heads	4
CA-resolutions	32, 16, 8
Transformer Depth	1

Table 5.8: Hyperparameters for CCDM(with attention) trained on the Shape Color Dataset, UT-Zappo50K, CelebA.

5.2.2 Sampling

During the generation phase, we utilize DDIM (6) to accelerate the generation process. For each dataset, there is an unseen composition. We employ the trained CCDM to generate 1000 unseen compositional images using the unseen compositional class label. The hyperparameters used are illustrate in table 5.9

Hyperparameters	
Diffusion steps	100
Batch size	200
Guidance scale ω	1.8
Sampling stochasticity η	0
Number of samples	1000

Table 5.9: Hyperparameters in sampling phase

5.3 Evaluation Metrics

Deep learning tasks require specific evaluation metrics that are tailored to the particular problem at hand. To evaluate the performance of CCDM, our focus is on the model's ability to generate unseen composition images. Therefore, we did not utilize the traditional Frechet Inception Distance(FID) (37) for evaluation. During testing, the unseen composition images generated by CCDM are denoted as N, the unseen composition images selected by unseen image selector are denoted as N_{select}.

- **Classification accuracy of binary classifier:** In order to assess whether the generated images by the model align with the description of unseen compositional class labels, we train a supervised binary classifier for each category class label of the unseen compositional class label. We evaluate the performance of each classifier based on its accuracy on the test dataset of each compositional class label.

- **Sample accuracy:** Follow (4) evaluation metric, the images generated by CCDM are filtered by the image selector to choose those that match the unseen compositions. Sample accuracy is employed to assess CCDM's ability to generate unseen compositions, which includes whether the images generated by CCDM match the unseen compositional class

label and the class label of each condition.

$$\text{Sample accuracy} = \frac{N_{select}}{N} \tag{5.1}$$

5.4 Experiment results

We designed various architectures for testing on each dataset, including both two-condition and three-condition settings. In this comparison, we evaluated different conditioning mechanisms and class encoder structures using sample accuracy. The detailed dataset settings are provided section 5.1 in The detailed conditioning mechanisms are provided in figure 4.5, figure 4.6, figure 4.7, figure 4.8.

5.4.1 Experiment results on shape color dataset

After the successful experimentation with the 2D point dataset, we proceeded to test our model with more complex and higher-dimensional data, which is a major application of modern diffusion models:image generation. We began with our self-created shape color dataset 5.1.1, employing various architectures for the generation of unseen composition images. We designed different scenarios to assess the performance of CCDM

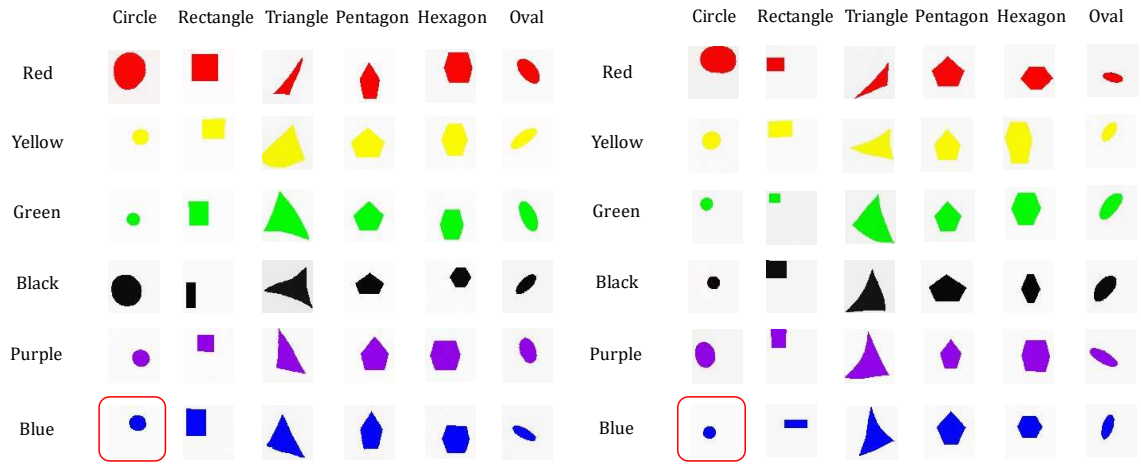

Figure 5.1: Under two condition setting of shape color dataset. Samples generated from CCDM, the unseen composition images are represented in red boxes.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	All Corrects accuracy ↑
Add	0.54	0.55	0.31
AdaGN	0.46	0.76	0.35
IC	0.87	**0.80**	**0.70**
CA	**0.99**	0.61	0.61

Table 5.10: Architecture comparison on Shape Color Dataset (Two conditions setting)

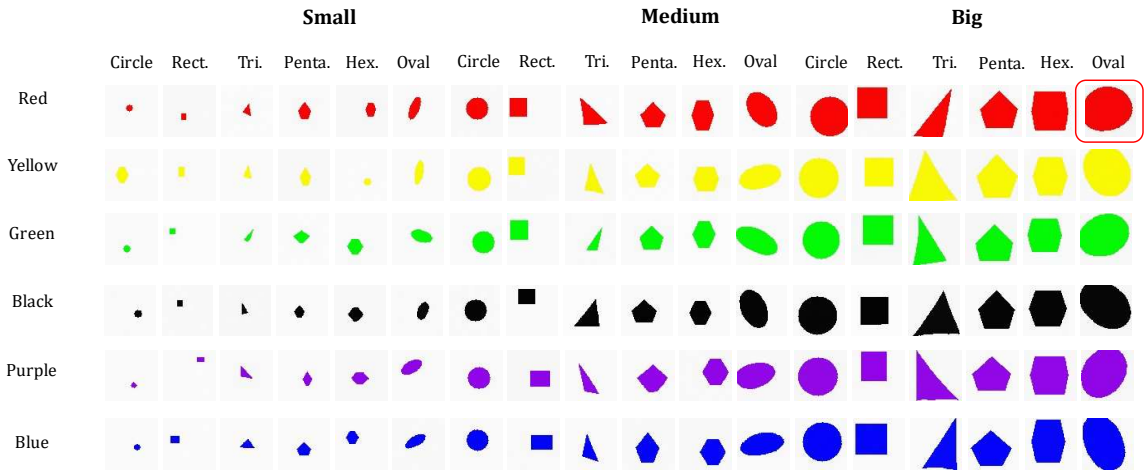

Figure 5.2: Samples of seen and unseen composition images generated from different architecture of CCDM, the unseen composition is represented by images enclosed in red boxes.

From the experimental results in figure 5.1 and figure 5.2, CCDM were able to successfully generate images of compositions that were not present during training. The results for seen compositions were also very promising across the board. These results also boosts our confidence in CCDM's ability to generalize to unseen compositions.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	c_3 accuracy ↑	All Corrects accuracy ↑
Add	**1.0**	**1.0**	0.98	0.98
AdaGN	0.99	0.99	0.96	0.95
IC	**1.0**	**1.0**	**1.0**	**1.0**
CA	**1.0**	0.99	**1.0**	0.99

Table 5.11: Architecture comparison on Shape Color Dataset (Three conditions setting)

5.4.2 Experiment results on UT-Zappo50K

After confirming that CCDM possesses the capability to generate unseen compositions, we proceeded to test CCDM on real-world datasets, specifically public datasets. We conducted tests on UT-Zappo50K, a common dataset in CZSL (Zero-Shot Learning), and made some modifications to the dataset structure. We examined CCDM's performance under different structures and scenarios. However, we encountered a challenge—instability in unseen composition generation. CCDM's ability to generate unseen composition images was not consistently stable under certain circumstances in these datasets. To address this, we employed an image selector to mitigate the issue. We will compare various architectures and evaluate their ability to generate unseen compositions. Detailed parameters of each architecture are provided in table 5.7, table 5.8.

Figure 5.3: Under two condition setting of UT-Zappo50K. Samples generated from CCDM, the unseen composition images are represented in red boxes.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	All Corrects accuracy ↑
Add	0.17	**0.9**	0.07
AdaGN	0.28	0.71	0.06
IC	0.31	0.86	**0.18**
CA	**0.62**	0.50	0.16

Table 5.12: Architecture comparison on UT-Zappo50K Dataset (Two conditions setting)

Figure 5.4: Under three condition setting of UT-Zappo50K. Samples generated from CCDM, the unseen composition images are represented in red boxes.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	c_3 accuracy ↑	All Corrects accuracy ↑
Add	**1.0**	0.32	0.93	0.27
AdaGN	**1.0**	0.75	0.5	**0.29**
IC	**1.0**	0.25	**0.94**	0.21
CA	**1.0**	**0.99**	0.06	0.06

Table 5.13: Architecture comparison on UT-Zappo50K Dataset (Three conditions setting)

Figures 5.3 and figure 5.4 demonstrate CCDM's performance on the UT-Zappo50K dataset. It accurately generates unseen composition images, showcasing varied styles similar to real shoes. This illustrates CCDM's applicability to real-world datasets.In the results of table 5.12, where c_1 represents "Heel," a subtle attribute, similar to the observations in table 5.15, we notice similar trends. In table 5.13, where c_2 still represents "Heel" and c_3 represents "Sandal," we observe that Slipper and Sandal have many similarities in the UT-Zappo50K dataset. This leads to CCDM generating incorrect images, with many resembling slippers. Through experimentation, we confirm that CCDM possesses the capability of compositional zero-shot image generation. However, CCDM struggles with generating features that are subtle or have low separability, resulting in lower accuracy when generating such samples.

5.4.3 Experiment results on CelebA

To demonstrate that CCDM's ability to generate unseen compositions is not limited to a single case, we chose to conduct tests on the CelebA dataset, a well-known dataset in the field of image synthesis. CelebA Dataset is renowned for its comprehensive annotations, which facilitated a smoother experimental process. We selected several attributes from the original CelebA dataset to serve as different conditions, as detailed in 5.1.3. We conducted various tests on CCDM using different architectures and scenarios on the CelebA dataset.Detailed parameters of each architecture are provided in table 5.7, table 5.8.

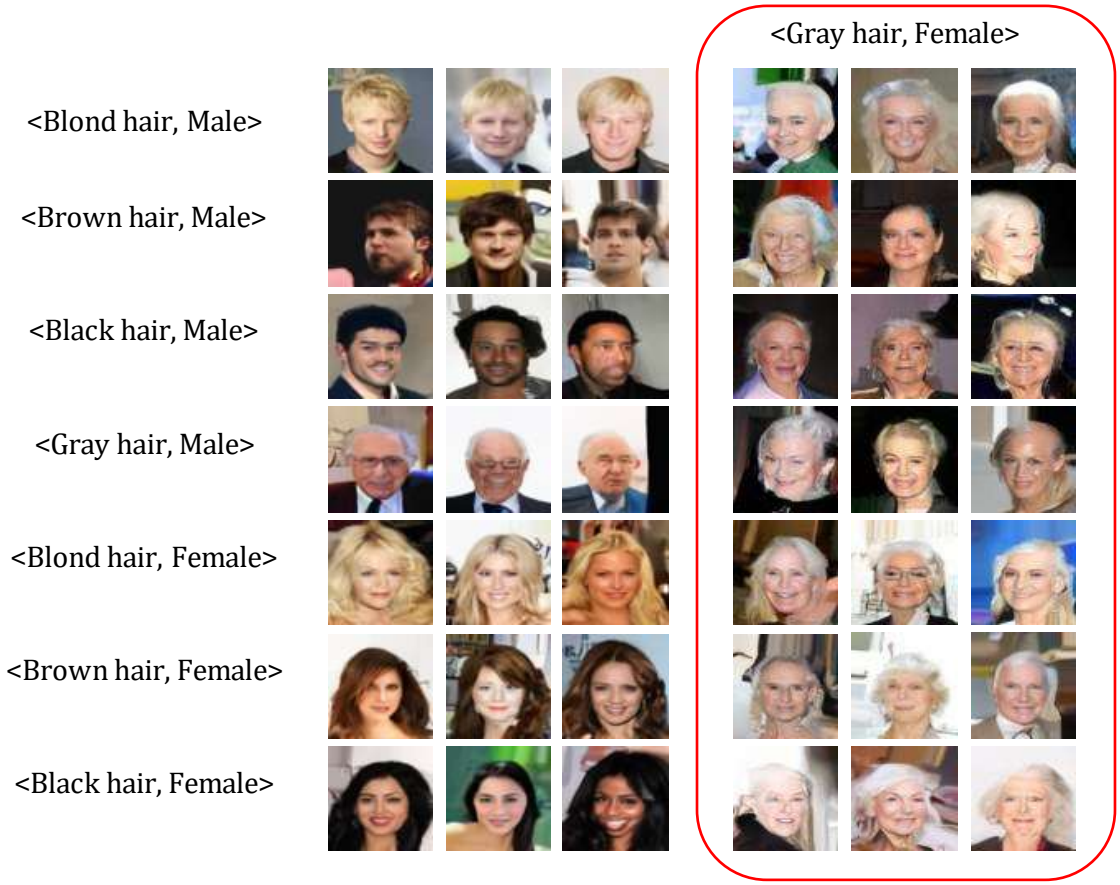

Figure 5.5: Under two condition setting of CelebA. Samples generated from CCDM, the unseen composition images are in red boxes.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	All Corrects accuracy ↑
Add	0.36	0.82	0.20
AdaGN	**0.58**	0.74	**0.32**
IC	0.49	0.73	0.27
CA	0.35	**0.93**	0.29

Table 5.14: Architecture comparison on CelebA Dataset (Two conditions setting)

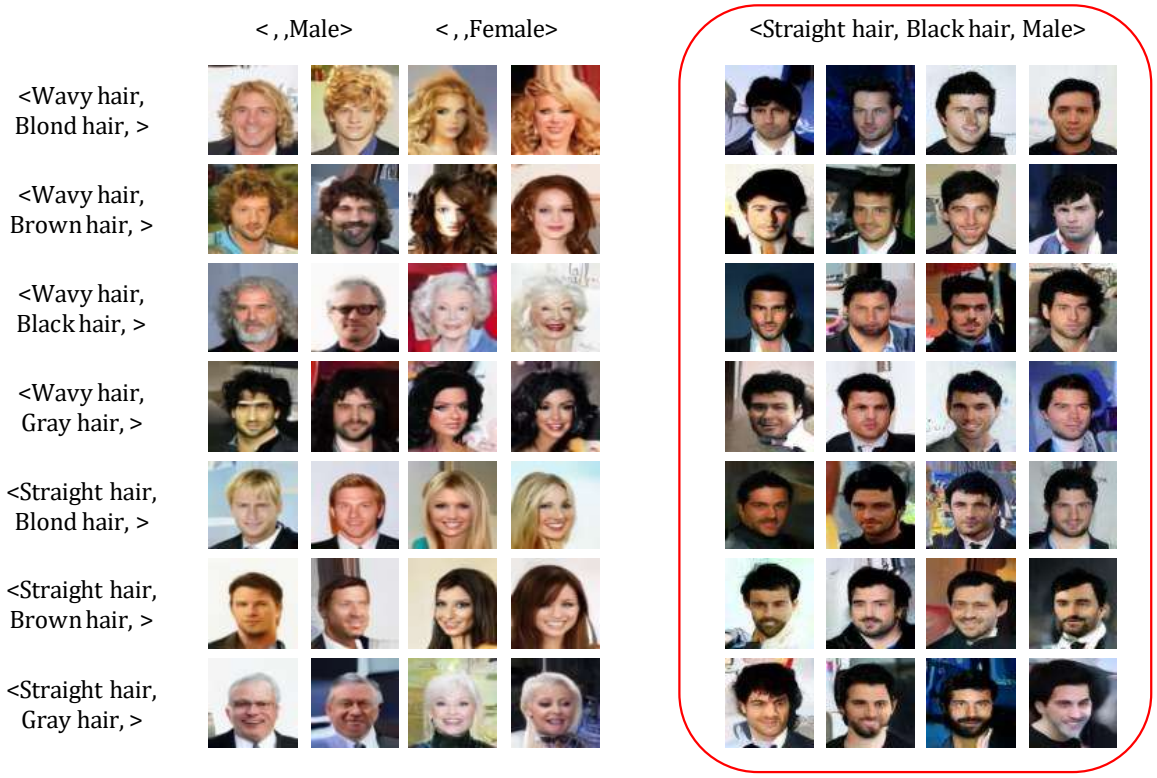

Figure 5.6: Under three condition setting of CelebA. Samples generated from CCDM, the unseen composition images are in red boxes.

Architecture	c_1 accuracy ↑	c_2 accuracy ↑	c_3 accuracy ↑	All Corrects accuracy ↑
Add	**0.20**	0.99	0.99	**0.20**
AdaGN	0.11	**1.0**	**1.0**	0.11
IC	**0.20**	0.99	**1.0**	0.19
CA	0.06	**1.0**	**1.0**	0.06

Table 5.15: Architecture comparison on CelebA Dataset (Three conditions setting)

In figure 5.5 and figure 5.6, CCDM performs well on the CelebA dataset under the two-condition setting and three condition setting. However, the results generated by CCDM are not perfect. In table 5.14 and table 5.15, we compare the accuracy of generating unseen composition images using four different architectures. Although all can generate the unseen composition images correctly, the accuracy is not ideal. In the two-condition setting, where c_1 represents " Gray hair," this attribute has only a few samples in the dataset. In the three-condition setting

c_1 represents ”Straight hair”, distinguishing between ”Straight hair” and ”Wavy hair” in male hair has low separability. We believe these factors contribute to the relatively low accuracy of CCDM in generating unseen composition images.

5.5 Experiment analysis

Architecture	Accuracy ↑
Add	0.34
AdaGN	0.34
IC	**0.42**
CA	0.36

Table 5.16: Overall architecture comparison

Table 5.16 illustrates that among the four conditioning modules we used, Image Class Concatenation performs the best for compositional zero-shot image generation. This closely aligns with our expectations. For the diffusion model with compositional class labels as conditions, element-wise addition in the Addition module may aggregate information from multiple class labels, potentially hindering the model's ability to generate correct samples. The same rationale applies to AdaGN. Cross attention, when using compositional class label embeddings as keys and values, suffers from limited dimensions in both keys and values. Combined with the relatively insufficient data, it fails to fully exploit the power of cross attention. Concatenation, on the other hand, maximally preserves the information from compositional class embeddings. Additionally, concatenation avoids mixing the class embeddings of different conditions using addition, making it less prone to confusion for the model.

CCDM can achieve compositional zero-shot image generation across various datasets using the four architectures we employed. However, the quality of generated images is not consistently stable. We attribute this to not having found the most suitable conditioning module for compositional class labels yet. Nonetheless, the success on these datasets gives us confidence

in using compositional class labels to guide the diffusion model. In future research, we aim to identify the optimal conditioning module for compositional class labels.

Chapter 6

Conclusion

We proposed a diffusion model conditioned on compositional class labels for compositional zero-shot image generation.Starting from the most basic dataset, we employed various methods to achieve Unseen data generation. Eventually, we adopted the approach of using compositional class labels as conditions to guide the generation model. By succeeding with the simplest dataset, we further validated that our model can achieve compositional zero-shot image generation through different datasets. Applying compositional cognitive abilities to the domain of image generation, our model can achieve ”learning from old compositional concepts and generalizing to new compositional concepts.”

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[3] Y. Du, S. Li, and I. Mordatch, "Compositional visual generation and inference with energy based models," *arXiv preprint arXiv:2004.06030*, 2020.

[4] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, "Compositional visual generation with composable diffusion models," in *European Conference on Computer Vision*. Springer, 2022, pp. 423–439.

[5] H. Taud and J. Mas, "Multilayer perceptron (mlp)," *Geomatic approaches for modeling land change scenarios*, pp. 451–455, 2018.

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[7] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[9] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[10] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[11] H. Li, Y. Yang, M. Chang, S. Chen, H. Feng, Z. Xu, Q. Li, and Y. Chen, "Srdiff: Single image super-resolution with diffusion probabilistic models," *Neurocomputing*, vol. 479, pp. 47–59, 2022.

[12] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4713–4726, 2022.

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[14] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–10.

[15] M. Özbey, O. Dalmaz, S. U. Dar, H. A. Bedel, Ş. Özturk, A. Güngör, and T. Çukur, "Unsupervised medical image translation with adversarial diffusion models," *IEEE Transactions on Medical Imaging*, 2023.

[16] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," *arXiv preprint arXiv:2108.01073*, 2021.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[18] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[19] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 479–36 494, 2022.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[21] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[22] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[23] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C. P. Lim, X.-Z. Wang, and Q. J. Wu, "A review of generalized zero-shot learning methods," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[24] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[25] S. Rahman, S. Khan, and F. Porikli, "A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, 2018.

[26] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5542–5551.

[27] Z. Han, Z. Fu, S. Chen, and J. Yang, "Contrastive embedding for generalized zero-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2371–2381.

[28] G. Xu, P. Kordjamshidi, and J. Y. Chai, "Zero-shot compositional concept learning," *arXiv preprint arXiv:2107.05176*, 2021.

[29] M. Yang, C. Xu, A. Wu, and C. Deng, "A decomposable causal view of compositional zero-shot learning," *IEEE Transactions on Multimedia*, 2022.

[30] X. Lu, S. Guo, Z. Liu, and J. Guo, "Decomposed soft prompt guided fusion enhancing for compositional zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23 560–23 569.

[31] N. Saini, K. Pham, and A. Shrivastava, "Disentangling visual embeddings for attributes and objects. 2022 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 648–13 657.

[32] C.-Y. Chen and K. Grauman, "Inferring analogous attributes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 200–207.

[33] I. Misra, A. Gupta, and M. Hebert, "From red wine to red tomato: Composition with context," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1792–1801.

[34] T. Nagarajan and K. Grauman, "Attributes as operators: factorizing unseen attribute-object compositions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 169–185.

[35] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” *arXiv preprint arXiv:2010.02502*, 2020.

[36] A. Q. Nichol and P. Dhariwal, “Improved denoising diffusion probabilistic models,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 8162–8171.

[37] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *Advances in neural information processing systems*, vol. 30, 2017.