DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Does Feasibility Matter? Understanding In- and Out-of-Distribution Data Impact

Yiwen Liu

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Does Feasibility Matter? Understanding In- and Out-of-Distribution Data Impact

Spielt Machbarkeit eine Rolle? Verstehen des Einflusses von In- und Out-of-Distribution-Daten

Author:	Yiwen Liu
Supervisor:	Prof. Dr. Zeynep Akata-Schulz
Advisor:	M.Sc. Jessica Bader, M.Sc. Jae Myung Kim
Submission Date:	29.10.2024

I confirm that this master's thesis in informatics: robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 29.10.2024 Yiwen Liu

Abstract

With the development of increasingly photorealistic diffusion models, models trained in part or fully on synthetic data achieve progressively better results. However, diffusion models still routinely generate images that would not exist in reality, such as a cat floating above the ground or with unrealistic texture artifacts. We consider these types of images **infeasible**. Intuitively, training with infeasible images should be detrimental to a model's ability to generalize to real data; hence, infeasible images are typically treated as out-of-distribution (OOD) and removed from the training set whenever possible via filtering techniques. But does feasibility really matter? In this paper, we investigate the necessity of feasibility when generating synthetic training data for classifiers by using an LLM to define per-class in-distribution (ID) and OOD attributes realting to the three target categories: background, color, and texture. We introduce a minimal-change generation pipeline, VariReal, to create feasible and infeasible comparison pairs from real images. In this way, we isolate the target attribute from other information in the synthetic data. We show that feasibility of the synthetic data does not majorly affect performance on several fine-grained classification datasets when LoRA fine-tuning CLIP on synthetic data, showing less than 1 percentage point difference in top-1 accuracy between feasible and infeasible datasets across almost all test settings when evaluated on Oxford Pets, FGVC Aircraft, and Stanford Cars. More importantly, we show that mixing feasible and infeasible data within synthetic training datasets does not significantly impact performance when compared with models trained on only feasibles or infeasible synthetic images.

Kurzfassung

In Zeiten großskaliger vortrainierter Modelle erfordert der Mangel an hochwertigen Daten die Generierung von Datensätzen, die reale Verteilungen widerspiegeln. Generative Modelle, insbesondere Diffusionsmethoden wie Stable Diffusion, bieten eine Lösung, indem sie synthetische Daten erzeugen. Aufgrund von Machbarkeitsproblemen enthalten synthetische Daten jedoch häufig Out-of-Distribution (Out-of-Distribution (OOD)) Bilder, die Elemente einführen, die von plausiblen realen Szenarien abweichen. Diese Arbeit untersucht den Einfluss von machbarkeitsbezogenen OOD-Daten auf nachgelagerte Aufgaben, insbesondere Klassifizierungsaufgaben. Wir schlagen eine Pipeline zur Minimal-Änderung-Synthetischen-Daten-Generierung (Minimal-Change Synthetic Data Generation (MCSDG)) vor, die kontrollierte Attributvariationen in synthetischen Bildern wie Hintergrund, Farbe oder Textur gewährleistet. Mit diesem Ansatz entwickeln wir machbare In Distribution (ID)- und nicht machbare OOD-Datenpaare und erforschen die Auswirkungen machbarer Daten auf das Modelltraining. Unsere Experimente, die mit einem feinabgestimmten CLIP-Klassifikator unter Verwendung von LoRA-Modulen durchgeführt wurden, zeigen, dass Machbarkeit die Klassifikationsleistung nicht beeinflusst. Im Gegensatz dazu wirken sich Änderungen am Hauptobjekt oder am Hintergrund bei Klassifizierungsaufgaben erheblich auf die Leistung aus. Durch eine umfassende Analyse leisten wir einen Beitrag zu neuen Erkenntnissen über die Rolle der Machbarkeit und die strategische Nutzung von OOD-Daten zur Erhöhung der Modellrobustheit bei nachgelagerten Aufgaben.

Contents

1. Introduction

1.1. Background

In recent years, large-scale pre-trained models [1, 2, 3, 4, 5, 6] have significantly surpassed traditional deep learning and machine learning approaches in various tasks. However, as the scale of training data grows, access to high-quality data has become increasingly limited [7], posing challenges to further improving these large models' capabilities. With the popularity of generative models [8, 9] like Stable Diffusion [2], researchers are increasingly leveraging these models to generate high-fidelity synthetic data that closely resembles real-world data, offering a solution to data scarcity [**surveydpm**, **surveyhealth**].

1.2. Motivation

Prior studies have explored synthetic data generation under a limited few-shot real image setting [10, 11, 12, 13, 14, 15, 16, 17]. These works aim to create synthetic data that approximates the real-world data distribution while avoiding overfitting to the limited available examples. Some studies [11, 13] suggest that synthetic data can offer benefits beyond those of real data. However, the inherent randomness in the image generation process of diffusion process [1, 2] can introduce domain shifts [13] or implausible scenarios like "a dog floating in the sky" [12] that do not reflect realistic patterns, which might intuitively be counterproductive.

Interestingly, some studies [18, 19, 20] suggest that OOD data can positively impact downstream tasks when mixed with real data in certain proportions. A typical example is data augmentation [19], where some augment methods introduce OOD data relative to the original distribution yet still provide benefits. While the advantages of OOD data generally diminish as divergence from the original distribution increases [18], these findings demonstrate OOD data is not always harmful. Conversely, incorporating feasible content similar to the training domain is naturally beneficial. The ALIA method [14] augments datasets with "feasible backgrounds", demonstrating performance improvement with ID data. This raises a key question: does training data feasibility affect downstream tasks, and could control the incorporation of such OOD data improve performance?

1.3. Our Work

This work introduces an automatic minimal-change generation pipeline, **MCSDG**, based on Stable Diffusion [2]. MCSDG allows us to control object feasibility to create targeted synthetic comparison pairs, as the example shows in Figure **??**. We evaluate feasibility effectiveness

by employing the CLIP [21] classifier and fine-tuning it on the synthetic dataset generated using MCSDG. Precisely, we manipulate three key object attributes—background, color, and texture—to examine classifier performance under two conditions: (1) fine-tuning with synthetic data only and (2) mixed training with real and synthetic data. For each attribute, we consider feasible data as ID and infeasible data as OOD. For example, a black "Bombay" dog is a plausible real-world instance (ID), while a "Bombay" dog with white fur is infeasible and thus categorized as OOD.

Our experiments on three fine-grained datasets reveal several key insights. We also show that modifications similar to ALIA [14] do not necessarily need to select only feasible scenarios. Regardless of feasibility, changing the background can enhance the classifier's focus on the primary task, while foreground modifications for color and texture often challenge the classifier's learning process. We also demonstrate that mixing synthetic data can yield performance benefits when paired with real data.

In summary, our contributions are as follows:

- We propose MCSDG, an automated generation pipeline for producing minimal-change synthetic data by altering only one attribute at a time. This approach can be applied out-of-the-box to any object-centric classification dataset without additional fine-tuning.
- We generate and provide feasible (ID) and infeasible (OOD) dataset comparison pairs based on real images, covering three controlled attributes.
- To explore feasible and OOD data roles, we fine-tune CLIP with LoRA modules. Analyzing classification scores, we offer new insights into the impact of feasibility and the strategic use of OOD data for enhancing downstream task performance.

1.4. Thesis Overview

The thesis is organized into seven main sections. Section 2 introduces the fundamental knowledge of Diffusion Models. Building on this foundation, Section 3 presents related work, including current research on text-to-image models, few-shot image synthesis, OOD data, and other relevant algorithms. In Section 4, we detail our methodology, followed by experimental results in Section 5. Finally, Section 6 discusses future work, and the conclusions are presented in Section 7.

2. Fundamental

2.1. Introduction

Generative modeling is a core challenge in machine learning, with numerous applications in fields such as computer vision, audio synthesis, and natural language processing. Among the wide variety of generative models, diffusion-based approaches have gained substantial attention due to their theoretical foundation and high-quality results in image generation tasks. We present one example in Figure 2.1. Diffusion models rely on the principle of gradually adding noise to data and then learning how to reverse this process to recover the original data distribution. The Latent Diffusion Model (LDM) [2] extends this framework by applying diffusion in a compressed latent space rather than the original high-dimensional pixel space, significantly improving the computational efficiency.

We will firstly introduce the Denoising Diffusion Probabilistic Models (DDPM) in section 2.2, then Denoising Diffusion Implicit Models (DDIM) in 2.3, and finally Latent Diffusion Models (LDM) 2.4.

Figure 2.1.: The comparison of generated images for CelebAHQ [22] dataset for VAE, GAN and diffusion methods respectively. For VAE method, we need specialized method so that we could generate human face images very well [23, 24]. GAN [24] methods are not stable for training while DDPM [1] is a general process.

2.2. Denoising Diffusion Probabilistic Models

DDPM, proposed by Ho et al. [1], was one of the first diffusion models that demonstrated the potential of noise-based generative processes. The model operates on two phases: a

forward diffusion process that gradually corrupts the data and a reverse denoising process that attempts to recover the original data from noise.

2.2.1. Forward Diffusion Process

The forward diffusion process in DDPM is defined as a sequence of transformations that gradually add Gaussian noise to the original data. Starting with a data point $x_0 \sim q(x)$, the forward process generates a series of noisy samples $x_1, x_2, \ldots, x_T$ through the following transition distribution:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{2.1}$$

where $\beta_t \in (0,1)$ is a small positive constant that controls the variance of the noise added at each time step. As $t \to T$, the sample x_T approaches a pure Gaussian noise distribution, such that:

$$q(x_T|x_0) = \mathcal{N}(x_T; 0, I). \tag{2.2}$$

A notable property of this process is that it can be expressed in a closed form for any arbitrary time step t, allowing us to directly sample x_t from x_0 as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{2.3}$$

where $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$ and $\epsilon \sim \mathcal{N}(0, I)$ represents the Gaussian noise. This closed-form expression significantly simplifies training since it removes the need to compute every intermediate step sequentially.

2.2.2. Reverse Denoising Process

The reverse process aims to gradually remove noise from a noisy sample x_T to recover a high-quality approximation of the original data x_0. The reverse process is modeled by a neural network, typically parameterized by θ, which predicts the noise at each time step t. The reverse transition distribution is formulated as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{2.4}$$

where $\mu_\theta(x_t, t)$ is the mean predicted by the neural network, and $\Sigma_\theta(x_t, t)$ is the variance (which is either fixed or learned). The model is trained by optimizing the following loss function, which encourages the neural network to predict the noise added at each step:

$$L_{simple} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]. \tag{2.5}$$

This formulation is known as a denoising score-matching loss, and it is designed to minimize the difference between the actual noise ϵ and the noise predicted by the model $\epsilon_\theta(x_t, t)$. Over time, the network learns to reverse the forward diffusion process by progressively removing the noise, resulting in high-quality samples.

2.2.3. Methods for Predicting Noise

One of the key design choices in diffusion models is to predict the noise added at each time step rather than the raw pixel values directly. The reasons for this are as following:

- 1. Simplicity of Modeling Noise: Predicting noise in Gaussian processes is often simpler than directly predicting pixel values. This is because the distribution of noise in a Gaussian process is well-characterized, whereas pixel values can be highly complex and multimodal.
- 2. Stochasticity of Data Generation: Noise prediction aligns with the stochastic nature of the generative process, making it easier to learn a smooth approximation of the reverse process. The neural network essentially learns a denoising function real distribution that removes structured noise from the data.

When we are training the diffusion model, we sample the time-step t and add corresponding noise to the input images. Then we use a U-Net model to predict the added noise for the specific time-step t.

The U-Net consists of an encoder, a bottleneck, and a decoder. The encoder progressively downsamples the input, extracting high-level features, while the decoder upsamples the feature map back to the original resolution. Skip connections between corresponding layers of the encoder and decoder allow the network to retain high-resolution details, which is crucial for accurate noise prediction in diffusion models.

In the context of diffusion models, the input to the U-Net is the noisy image (or latent representation) at time step t, and the output is the predicted noise $\epsilon_\theta(x_t, t)$. The skip connections help the network retain fine details from the input, improving the quality of the generated samples.

2.3. Denoising Diffusion Implicit Models

DDIM [25] introduces a deterministic alternative to the sampling process in DDPM, achieving a trade-off between sample quality and computational efficiency. While DDPM uses a Markovian reverse process, where each step is stochastic, DDIM modifies the process to make it deterministic.

2.3.1. Deterministic Reverse Process

In DDIM, the forward process remains the same as in DDPM. However, the reverse process is modified to remove the stochasticity, allowing for deterministic sampling while retaining high-quality results. The reverse process in DDIM is given by:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(x_t, t). \tag{2.6}$$

This formulation guarantees a consistent mapping from the noise space to the data space, making DDIM particularly useful when faster sampling is required. The deterministic nature of the process also enables fewer sampling steps, reducing the time required for generation.

2.3.2. Relation Between DDPM and DDIM

While DDIM and DDPM share a similar forward diffusion process, their key difference lies in the reverse process. DDIM's reverse process eliminates the randomness associated with DDPM, but this also introduces a trade-off between flexibility and sampling speed. DDIM typically requires fewer sampling steps (as low as 10-20 steps) compared to DDPM, which may need hundreds of steps.

2.4. Latent Diffusion Models

Latent Diffusion Models (LDM) [2] build upon the principles of DDPM and DDIM by moving the diffusion process from pixel space to a lower-dimensional latent space.

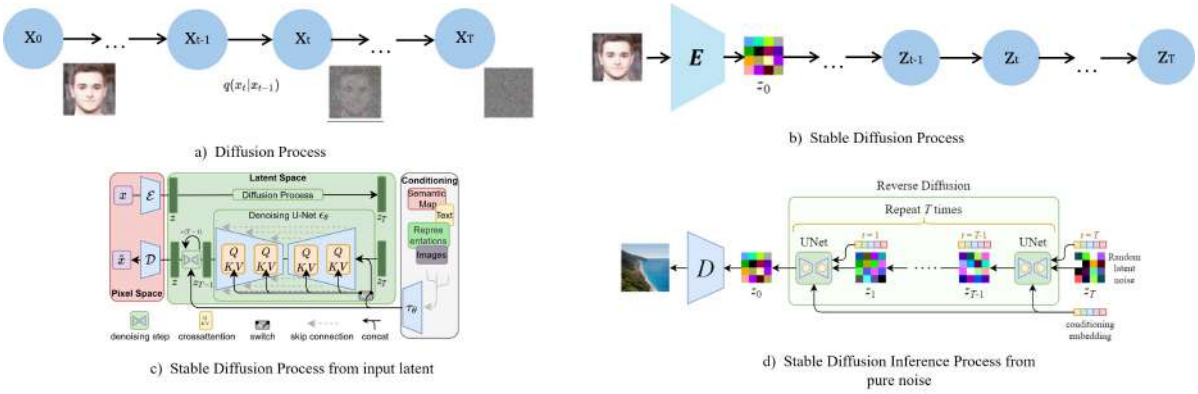

Figure 2.2.: The forward diffusion process are present in sub-figure a [1] and sub-figure b. Sub-figure c [2] shows the stable diffusion prior injecting process from encoding a input image, while sub-figure d [2] shows the inference process from a pure noise.

2.4.1. Motivation for Latent Space Diffusion

Operating in pixel space poses significant challenges for diffusion models due to the high dimensionality of images. For example, an image with a resolution of 1024×1024 contains over a million pixels, making the forward and reverse processes computationally expensive. So traditionally, most DM model operates images on a lower resolution. LDM addresses this issue by first encoding the input images into a lower-dimensional latent space using a pre-trained encoder such as a Variational Autoencoder (VAE).

Once in the latent space, the diffusion process can be applied more efficiently, as the dimensionality is significantly reduced. After the reverse process is complete, the latent representation is decoded back into pixel space using the VAE decoder, recovering the original high-resolution image. The process is shown in Figure 2.2.

2.4.2. Latent Diffusion Process

Let $E(x_0)$ represent the encoder of the VAE that maps a high-dimensional image x_0 into a lower-dimensional latent space z_0. The forward diffusion process in this latent space is similar to the pixel-based diffusion process:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \tag{2.7}$$

where z_t is the noisy latent representation, and ϵ is the Gaussian noise added at each time step t. The reverse process is modeled using a neural network, typically a U-Net architecture, that predicts the noise added to the latent representation at each time step to minimize the loss:

$$L_{LDM} := \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0,1), t} \left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2 \right]. \tag{2.8}$$

Once the reverse process is complete, the latent representation z_0 is decoded back into pixel space using the decoder $D(z_0)$ of the VAE:

$$x_0 = D(z_0). \tag{2.9}$$

2.4.3. Efficiency of Latent Diffusion

By moving the diffusion process to the latent space, LDM significantly reduces the computational burden. Since the latent space has a much lower dimensionality compared to pixel space, both the forward and reverse processes become more efficient. This allows for faster sampling and enables the generation of high-resolution images without the need for extensive computational resources.

3. Related Work

3.1. Text-to-Image Diffusion Models

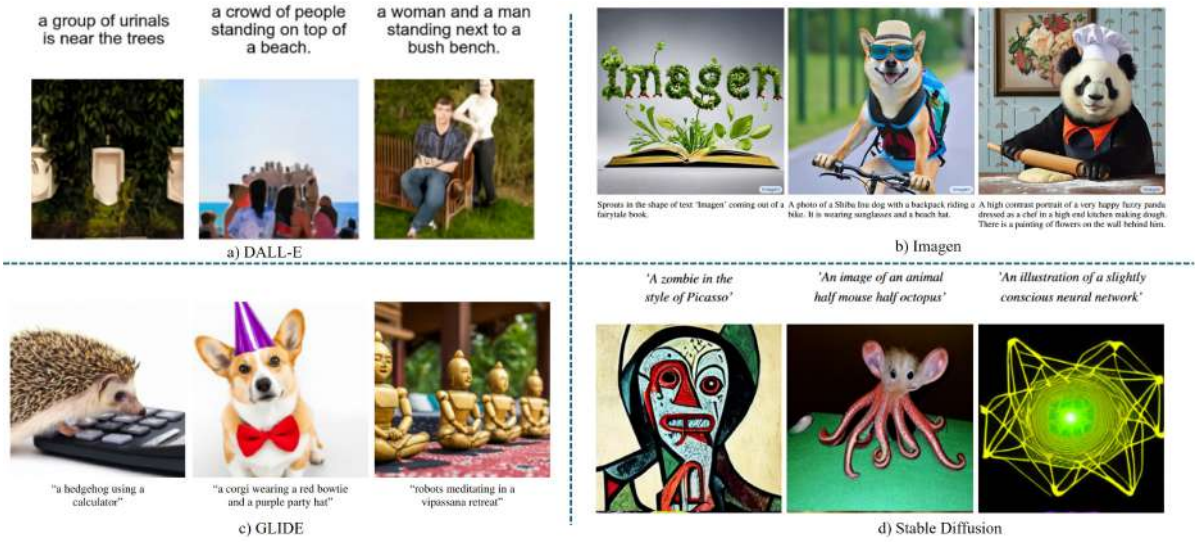

Figure 3.1.: The generated image examples from representative T2I diffusion models [26, 27, 28, 2].

3.1.1. DALL-E

DALL-E [26] is a text-to-image generation model. Its core approach involves converting text features into image features, utilizing two main components: a diffusion-based Diffusion Prior module and a diffusion-based Diffusion Decoder module. The Diffusion Prior uses a CLIP text encoder to extract text features to iteratively update the randomly initialized image embeddings.

The Decoder uses the image embeddings along with the text captions as inputs. The text encoding and image embeddings guide the diffusion process in the Decoder to generate the final image. DALL-E excels in complex visual generation tasks by integrating multimodal training on text and image data. The generated images are shown as a) Figure 3.1.

3.1.2. Imagen

Imagen [28] is a text-to-image generation model that combines the capabilities of large-scale language models and diffusion models to generate high-resolution images. It uses a pre-trained T5 model to extract features from the input text, enhancing the semantic understanding of the text and making the generated images more aligned with the descriptions.

A Text-to-Image Diffusion Model then generates a 64×64 image from random noise based on the text embedding. The generated image is upscaled to 256×256 using a Super-Resolution module, and then to 1024×1024 using another Super-Resolution module, both guided by the text embedding. A new, efficient U-Net architecture is also introduced, providing higher computational efficiency, better memory usage, and faster convergence.

The authors also demonstrated that extending the text encoder significantly impacts performance more than extending the diffusion model itself. The generated images are shown as b) Figure 3.1.

3.1.3. GLIDE

GLIDE [27] introduces a simple yet effective diffusion model for generating images conditioned on text. Before GLIDE, Guided Diffusion [29] is to classify the generated image at each step of the reverse process using a classification network, and then compute the gradient based on the cross-entropy loss between the classification score and the target class. This gradient is used to guide the next step of sampling.

Although Guided Diffusion does not require retraining the diffusion model, it incurs additional computation costs since each reverse process requires at least one pass through the network. Furthermore, the separate training of the guidance function and the diffusion model makes it difficult to scale the model efficiently.

To address these limitations, Classifier-Free Guidance [30] was introduced. The diffusion model is trained with both conditional and unconditional settings, and the outputs of these paths are combined to control the details and diversity of the generated image. The noise estimation in the generation process is defined as:

$$\hat{\epsilon}_\theta(x_t|c) = \epsilon_\theta(x_t) + w \cdot (\epsilon_\theta(x_t|c) - \epsilon_\theta(x_t)), \tag{3.1}$$

where $\epsilon_\theta(x_t|c)$ and $\epsilon_\theta(x_t)$ are the conditional and unconditional noise predictions, respectively, and w is the guidance weight controlling the influence of the condition.

By adjusting w, an optimal balance can be achieved between enhancing the alignment with the conditioning input (e.g., a text description) and preserving diversity.

GLIDE is based on Classifier-Free Guidance, uses a larger-scale diffusion model trained on a large dataset. It replaces the previous class conditions with text descriptions, resulting in higher-quality image generation and better control over the image details. The generated images are shown as c) Figure 3.1.

3.1.4. Stable Diffusion

In the text-to-image stable diffusion [2] (T2I) model, text conditioning is achieved by incorporating a text encoder, such as CLIP [21], to map the input text $\mathbf{y}$ into a latent representation. This representation is then used to guide the diffusion process at each step. Specifically, the text embedding $\mathbf{z}_y$ is concatenated or injected into the intermediate layers of the U-Net model using cross attention [31], which predicts the noise ϵ_θ for denoising. The conditional probability $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_y)$ is optimized, where $\mathbf{z}_y$ provides the semantic information from the text that steers the model towards generating an image that matches the input prompt. The model learns to associate the text embedding with corresponding image features, ensuring that the generated image reflects the text description. The text input could be weighted by modifying the attention weights of specific token [32], like using the library Compel [33]. The generated images are shown as d) Figure 3.1.

Stable Diffusion XL

Stable Diffusion XL [34] is an enhanced version of Stable Diffusion, achieving higher-quality image generation through a deeper architecture and larger training dataset. The improvements include using a more complex UNet with additional parameters and employing a larger text-conditioning encoder, specifically OpenCLIP ViT-bigG [35], as well as an extra text encoder, CLIP ViT-L [21]. These additions enrich the conditioning process with complementary textual features, producing more refined image outputs.

Stable Diffusion XL also introduces an additional diffusion model specialized in handling high-quality, high-resolution data. The model is trained on a larger and more diverse image dataset, which significantly improves its generalization capability and image quality. This larger dataset allows Stable Diffusion XL to generate images with higher fidelity and more realistic details.

3.1.5. Expanding the Generation Domain from Pretrained Models

To generate images in a new domain beyond the training data, various generation techniques can be applied.

Textual Inversion

Textual Inversion [36] is a method for custom text-to-image generation by learning embeddings for specific terms. When introducing a new concept to a pretrained model, a placeholder token not present in the original training data is chosen, and a learnable embedding is associated with it. You could check the Figure 3.2 a).The original weights of the SD model are not modified; instead, the placeholder token's embedding is replaced with the learned embedding during inference. This approach requires only a few examples to introduce new concepts.

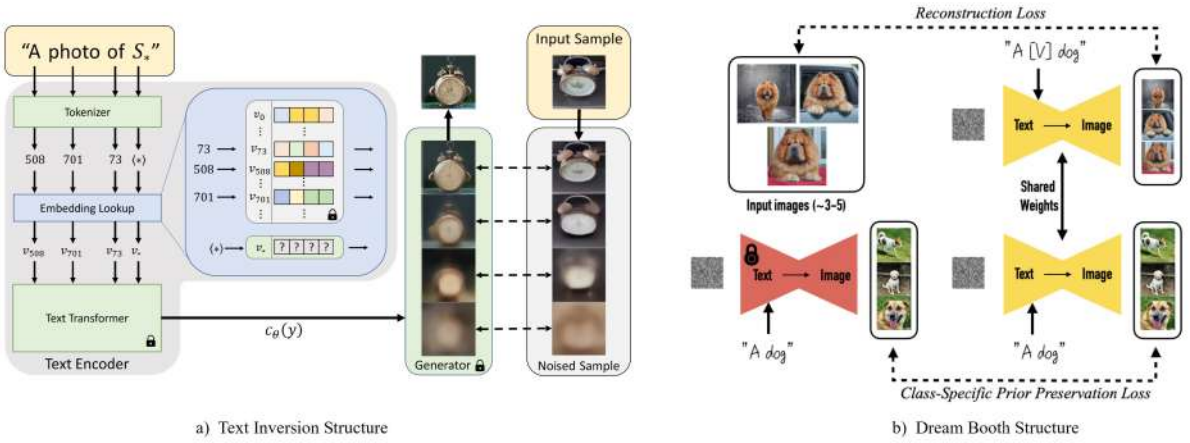

Figure 3.2.: The architecture illustration for Textual Inversion [36] and DreamBooth [37].

DreamBooth

DreamBooth [37] is a technique for personalized image generation, primarily used to fine-tune large-scale diffusion models with a few images to generate images with personalized characteristics. When attempting to learn new concepts using a small number of images (> 3), directly training on the limited data can lead to language drift or overfitting. The authors propose introducing a new token along with corresponding image examples and fully fine-tuning the SD model. DreamBooth enables the generation of high-quality images for specific objects while retaining the original generalization capabilities of the model.

To preserve the diversity of the original SD model's output, an additional mean squared error (MSE) loss is applied between the fine-tuned and original model outputs, maintaining the diversity during the fine-tuning process.

The methods described above are commonly used to introduce new concepts into pretrained SD models. There are also other similar techniques. For example, InterpretDiffusion [38] introduces a trainable vector in the UNet bottleneck to achieve the desired modifications. Another approach [39] removes the background from the main object in the new image and encodes it using an image encoder, incorporating cross-attention with the text information. However, this method requires fine-tuning the original SD model on new images. The method is demonstrated in Figure 3.2.

ControlNet

ControlNet [40] provides more precise constraints for text-to-image generation processes. The authors propose creating a trainable copy of a pretrained diffusion model while keeping the original weights fixed. The trainable copy takes the conditioning information c as input, with the input connected to the model using zero convolution to prevent altering the original training weights. ControlNet supports controlling the image generation process using external inputs, such as poses, depth maps, or other features, ensuring that the generated image meets

the user's requirements.

To enhance multi-level constraints, Uni-ControlNet [41] introduces a method to combine different types of constraints, such as edge maps, depth maps, and segmentation maps, for more comprehensive control.

Similar methods include:

- **ILVR [42]**: This approach modifies the DDPM sampling method to enable the generation of new concept images without fine-tuning.
- **Context Diffusion [43]**: This model extends ControlNet by incorporating reference images to influence not only the structure but also the style, color, and texture of the generated images. The method adds the image features as a condition in the cross-attention mechanism of the text condition, changing the key-value sources from text-only features to a concatenation of image and text features. However, this model requires the reference image to be similar to the ControlNet input image.
- **ColorSD [44]**: This technique uses an image or text reference for color style transfer, allowing for style and color adjustments without fine-tuning, simply by adjusting the text prompt.
- **Paint by Example [45]**: This approach performs inpainting on specified masked areas using a reference image, retaining the main subject and injecting it into the SD model via cross-attention.

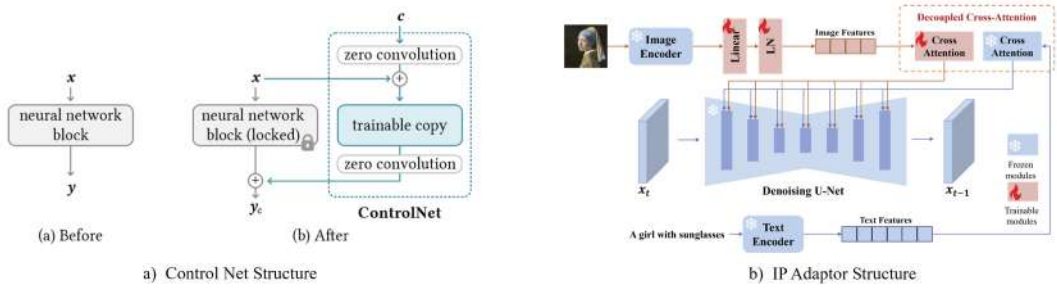

Figure 3.3.: The illustration of ControlNet [40] and IP-Adaptor [46].

IP-Adapter

The IP-Adapter [46] is a model adapter designed for image generation that enhances the generative capabilities of the original model without modifying its architecture. By introducing decoupled attention, features from a reference image can be added to the pretrained model's text attention, enabling additional conditioning based on the reference image. This is illustrated in Figure 3.3. The underlying principle involves summing the results of two cross-attention operations. It has parameter controls the scale of the adaptation; a higher scale makes the output more closely resemble the reference image.

3.2. Synthetic Data for Image Recognition

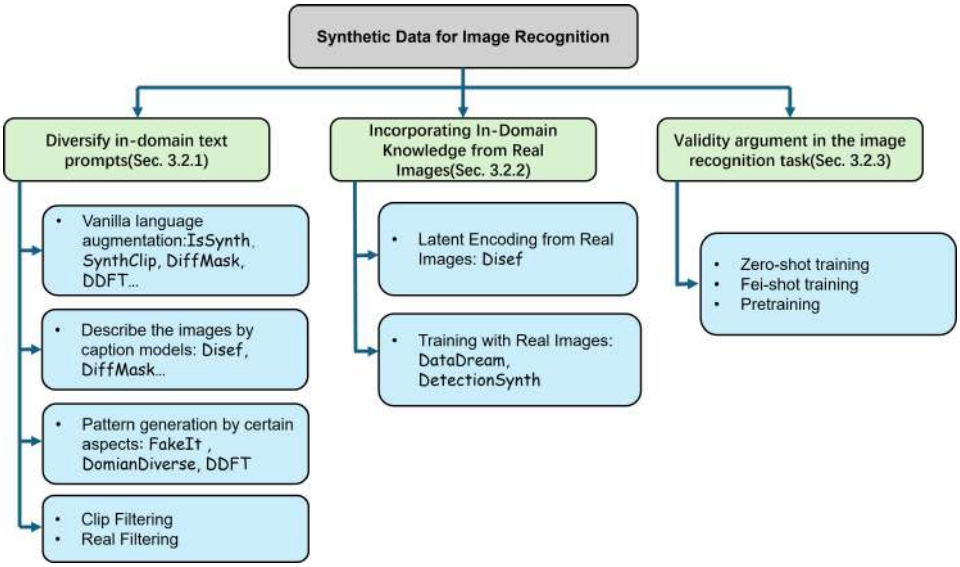

Figure 3.4.: The overview of Section 3.2.

Synthetic data has become a crucial resource in computer vision, particularly for creating large-scale, high-quality labeled datasets to reduce the burden of manual annotation [13]. With the rise of large language models (LLMs) and the extensive use of internet-based data for pre-training foundation models [7], the scarcity of available data has prompted growing interest in generating high-quality synthetic data.

There are two primary methods for generating synthetic data: simulation-based rendering pipelines and generative models. The first approach [47, 48] involves using 3D scenes rendered by a physical graphics engine. While this method offers flexibility for manual or semi-automated editing, it has several drawbacks:

- The rendered scenes require manual or semi-automatic design by individuals with expertise in physical engines, and numerous factors must be considered to minimize the domain gap between real-world and simulated environments.
- The model sources are often restricted, limiting the range of objects that can be used to build synthetic scenes. Additionally, creating new models for real-world objects is costly, especially for specific scenarios.
- Scaling the dataset's size and diversity requires redesigning the source scenes, making the process resource-intensive and inefficient for creating large datasets.

In contrast, using generative models to produce synthetic images is far more efficient. These models can generate high-fidelity, photo-realistic images, with the potential to produce an unlimited variety of outputs. Existing generative models, such as those based on Variational

Autoencoders (VAEs) [8] and Generative Adversarial Networks (GANs) [9], have been widely used. The generated synthetic images are widely used in tasks such as image recognition [11, 15, 14, 12], object detection [49], and semantic segmentation [17]. Since image recognition is a foundational task in computer vision, the challenge of generating diverse, in-distribution data to ideally solve this base task has become increasingly important. Recent works have identified several key methods for addressing this challenge, which can be summarized as follows:

3.2.1. Diversify In-domain Text Prompts

Vanilla Language Augmentation.

Synthetic data generated using only class labels with basic prompts often lacks the diversity found in real-world scenes, sub-objects within one super class, and semantic visual details. A basic prompt might be structured as "a photo of [CLS]," where "[CLS]" represents the corresponding class name. The simplest approach to generating more diverse images involves using a language model to enhance the basic prompts with the rich knowledge learned for specific classes during pretraining. For example, IsSynth [13] employs the T5 model [50] to increase linguistic diversity, enriching the range of generated images.The generated image examples are shown in Figure 3.5. Similarly, SynthClip [16] uses a large language model (LLM) as a text generator, focusing on prompt engineering to condition the model on particular classes, leading to more varied text prompt for a given class. DiffMask [17] addresses the monotony of generated images by incorporating K "[sub-class]" entries into basic prompts. For example, instead of using the simple prompt "Photo of a bird," which covers a broad class, they introduce sub-classes from sources like Wikipedia, resulting in prompts such as "Photo of a [sub-class] bird," where [sub-class] is one of $\{Sub_1, Sub_2, ..., Sub_K\}$.

In cases where class names have multiple meanings (e.g., "Crane" as both a bird and a machine), it can be difficult to predict which meaning a language model will generate without manual checks. To address this ambiguity, [10] propose a method to generate ImageNet domain synthetic images by first using a large language model (LLM) to generate K possible interpretations for ambiguous class names. Then, using CLIP, they extract features from the phrases and the original image, calculating their similarity to resolve the correct meaning.

However, augmenting text prompts can introduce noise, such as non-existent details or unrelated classes, into the generated images. To mitigate this, filtering or resampling methods[16] are often applied to ensure better alignment with the target real-world images.

Describe the Images by Caption Models

To generate more realistic and semantically rich details, caption models can be used to describe real images, helping to ensure that the generated content aligns with real-world data distributions. This method takes into account the kinds of content found in actual images, allowing the text prompts to better reflect real-world scenes. For instance, Disef [15] uses an Image Captioning Model to augment text input with detailed descriptions. Similarly,

DiffMask [17] uses the ClipRetrieval [51] model to retrieve the top K real captions from an image-caption dataset, ensuring that the text prompts are as realistic and diverse as possible.

Pattern Generation by Certain Aspects

The augmentation process involving language or caption models can be seen as a distillation of the knowledge contained within these models. However, beyond relying solely on the capabilities of these models, we can also hand-craft key aspects to better guide the generation of synthetic images.

For instance, FakeIt [14] addresses issues related to semantics and domain by using hypernyms and definitions from the WordNet graph. Hypernyms represent the parent class names of a given class in the graph, while definitions provide sentence-length descriptions of the semantics of each synset. Their experiments show that incorporating hypernyms and definitions improves classifier accuracy, suggesting that these techniques can help generate more realistic and precise synthetic images. However, despite the more expressive prompts, the generated images still lack diversity in pose and viewpoint. To tackle this, they employ classifier-free diffusion guidance and diversify the backgrounds using class-agnostic scenes. Although this approach may introduce domain shifts or unrealistic background-object combinations, the authors argue that such diversity, even if inconsistent with the data distribution, can serve as beneficial data augmentation.

DDFT [10] introduces the concepts of Contextualized Diversification (CD) and Stylized Diversification (SD) to enhance how LLMs describe real images. CD focuses on maximizing diversity in the context by combining elements like foreground objects, backgrounds, lighting conditions, and camera angles. SD, on the other hand, applies 60 different art styles to generate images. Using text prompts enriched with CD and SD, the model generates images with a broader range of visual diversity. Additionally, DomianDiverse [14] highlights the importance of domains such as location, weather, and time of day, which are unrelated to classification but add valuable context. Due to the limitations of using a caption model alone—which may not fully capture the nuances of input images—they further employ an LLM to summarize and filter out general elements, such as scenes, actions, camera poses, and zoom levels, that are unrelated to the primary class.

As discussed above, even with various techniques aimed at maximizing the realism and diversity of generated images, automatic generation pipelines can still produce unsuitable content or fail to capture all details from the text input. To address these issues, CLIP Filtering [15] and Real Filtering (when few-shot examples are available) [13] are used. The core idea is to leverage the zero-shot feature extraction capabilities of CLIP to obtain both image and text embeddings. The similarity between these embeddings is then compared, with high similarity indicating that the generated images closely match the text prompt or real images.

3.2.2. Incorporating In-Domain Knowledge from Real Images

Latent Encoding from Real Images

To encode more comprehensive information from real images beyond textual descriptions, Disef [15] proposes a method where noise is introduced into the image embeddings starting from a specific step. This process utilizes Stochastic Diffusion (SD) to disrupt low-level details while preserving high-level semantic features. Moreover, the approach uses the final caption from one image as a condition for generating the next image, ensuring diversity in the output. The primary goal is to generate images that belong to the same semantic category, maintaining similar visual patterns such as perspective or distinguishing characteristics. The generated image examples are shown in Figure 3.5.

Training with Real Images

The generation techniques discussed above focus on various aspects of image synthesis. We can retrain or fine-tune the Stable Diffusion model to capture the distributional information from real few-shot images more effectively. For example, DetectionSynth [49] aims to generate a synthetic detection dataset, which is more complex than a standard recognition dataset due to the need for additional bounding-box annotations. In this case, the images must contain multiple objects, and backgrounds should be diversified to vary detection difficulty. They argue that SD models, when guided by corresponding text prompts, often generate simplistic backgrounds with one or two objects, leading to reduced robustness in the detection model. To address this, they fine-tune the Stable Diffusion model using real images with SD objective loss, aiming to extend the dataset rather than focusing on few-shot learning.
Similarly, DataDream [11] utilizes the LoRA method to fine-tune the Stable Diffusion model using few-shot images and SD objective loss. Their work compares the impact of fine-tuning with few-shot images from each class individually versus using images across classes collectively. They conclude that fine-tuning with data from across classes helps the model learn the general distribution of the entire dataset more effectively. In this fine-tuning context, complex text prompts are not necessary—simple prompts like "a photo of [CLS]" are sufficient.

3.2.3. Validity Argument In the Image Recognition Task

The methods discussed above offer various approaches to generate high-fidelity and photorealistic synthetic data. IsSynth [13] conducts an in-depth analysis of the utility of synthetic data, reaching a counterintuitive conclusion. In zero-shot settings, synthetic data proves to be less effective for training compared to real data—50k synthetic images are equivalent to only 9k real images in terms of effectiveness. When training a CLIP image encoder from scratch, the performance of the model is significantly worse compared to the original pre-trained model. The authors attribute this performance gap to the domain differences between real and synthetic images. They further examine the effect of domain shifts in real images, such as switching tasks, and observe similar performance degradation. Moreover, increasing

Figure 3.5.: Our generate synthetic images using Disef [15] and IsSynth [13] methods on DTD dataset [52]. We could see the Disef method will generate some non-realistic images given few shots.

the volume of synthetic images does not result in continued performance improvement, as performance saturation is observed.

Figure 3.6.: Initial experiments using the Disef [15] method as an attribute editing technique on Tiny-ImageNet [53] demonstrated that Disef can adjust the "noise step" parameter in SD to control the level of noise added to the real latent representation. The results show that a larger "noise step" leads to greater divergence in the output. However, since the SD model has biases toward certain colors for specific attributes, it is challenging to directly alter colors accurately.

Another experiment shows that incorporating few-shot real images helps align the output more closely with the real data distribution. However, the more real images included, the less beneficial synthetic images become. On a positive note, synthetic data can be comparable to real images for model pretraining.

A key takeaway from this analysis is that the pretraining process used by models like SD [2] or GLIDE [27] often introduces biases, resulting in domain gaps when applied to different datasets. We could also experiment this in Figure 3.6. This means that poorer results on certain datasets may be attributed to the pretraining process not adequately learning the distribution of that specific data.

This work uses only basic text prompt augmentations, but concurrent research [16, 11, 12, 15, 10, 14] shows that synthetic data can have a positive impact, especially when combined with few-shot real images. In their experiments, retrained or fine-tuned classifier performance can surpasses models trained solely on real data, offering a valuable solution when training data is limited.

3.3. Value of Out-of-Distribution Data

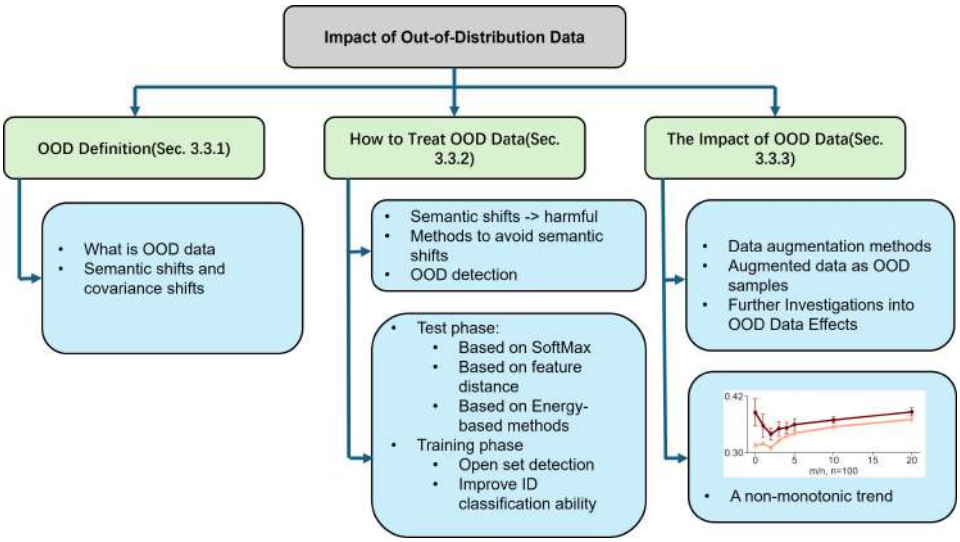

Figure 3.7.: The overview of Section 3.3.

3.3.1. OOD Definition

Out-of-distribution (OOD) data is defined with the in-distribution (ID) data range, which corresponds to a distribution shift. We first need to establish ID as the reference point for discussing OOD data. Typically, the distribution on which a model is trained is referred to as the source distribution, while the distribution of the data used during inference is called the target distribution [54]. The core issue with OOD data arises from these two distributions'

differences. Various factors contribute to this discrepancy. For example, real-world data is dynamic, making it challenging to ensure that the training data remains aligned with the test data distribution over time. Expanding the training dataset by adding samples from different distributions can introduce further variability. Ensuring the newly incorporated training data is aligned with original data is essential. This concept also underlies the motivation to generate synthetic images with a distribution similar to the real dataset, as discussed in Section 3.2.

OOD data can be broadly classified into two categories: semantic shifts and covariance shifts [55]. A semantic shift occurs when the label distribution differs between the training and test datasets. In this scenario, the model encounters test labels not present during training, leading to incorrect predictions where the model assigns an unseen sample to a known class [54]. We should avoid semantic shifts, and the methods are introduced in Section 3.3.1. On the other hand, covariance shifts arise when the input data distribution varies between training and testing due to biased sampling or inconsistencies during data collection [56]. An example would be a domain gap between the training and test data. For instance, if a model is trained on synthetic data for a segmentation task and the training set lacks real-world factors such as lighting variations or texture reflections, the model's performance in real-world conditions may suffer. The data feasibility focus of our paper also means this kind of covariance shifts OOD data.

Traditionally, OOD data has been intuitively regarded as harmful for training because it is assumed that the training data should be consistent with the test data. This assumption holds, especially in semantic shifts as previously mentioned. The existing method includes OOD detection and open set recognition to handle the semantic shifts in OOD data. For OOD detection, we could aim to eliminate OOD data from the test set by ensuring the training set covers as many scenarios as possible. However, given that real-world scenarios are infinite and dynamic, and our training datasets are constrained by time, resources, and potential biases in data collection, it is practically impossible to guarantee that the training distribution will perfectly match the test distribution—especially in real-world applications. This mismatch often results in performance degradation in machine learning algorithms. Various methods have been developed to detect OOD data and mitigate its harmful effects. Additionally, a model must recognize unseen classes and classify them as "unknown". As a result, open-set detection methods offer a potential solution to this challenge.

OOD Detection Methods

One primary approach is during the inference stage. Initially, the softmax operator [57] combined with a threshold was used to detect OOD data. The idea is that the minimum softmax value for ID data tends to be higher than the maximum value for OOD data. However, this method fails in cases where OOD data exhibits overconfidence. To address this, a temperature scaling factor [58, 59] is proposed to alleviate overconfidence and add small noise to input images to widen the score gap between ID and OOD data. The limitation of the softmax operator is that it is not proportional to data distribution and, therefore, may not capture input data effectively. In contrast, energy-based methods are more linearly related

to data distribution, prompting [60] to propose replacing softmax with an energy function. In addition to comparing the model's output, methods like calculating the Mahalanobis distance [61] between input data and selected prototypes using high-dimensional features have been proposed. This technique, however, often requires some known OOD samples. To avoid relying on such samples, activation values from neural networks can be leveraged to identify OOD data [62, 63]. Other approaches combine feature-level information and logits from the model output, using the raw data and its semantic content to improve OOD detection [64].

Another key approach occurs during the training phase. The central idea is that enhancing ID data detection improves OOD detection performance. For instance, [65] employs a Vision Transformer (ViT) backbone to better extract input features that help distinguish OOD data within the feature space. Another strategy cite zhang2023mixture, yang2024generalized involves exposing the model to OOD data by mixing ID data with unlabeled OOD data, allowing the model to learn how to detect OOD samples.

Models must be capable of identifying unknown classes rather than arbitrarily assigning them to known ones, as this can lead to critical failures. For instance, while accounting for every edge case in self-driving cars is impossible, the system must include failure-degradation mechanisms to handle OOD driving scenarios. If a self-driving car misclassifies an oddly shaped object as a vehicle on the highway, it could jeopardize passenger safety. To address this issue, [66] proposes a method for generating counterfactual images using a GAN model, which places data near the boundaries of real ID data in feature space. They generate near-boundary samples by introducing a new GAN training loss incorporating a K-class classifier score(where K represents ID classes). A classifier is trained with K+1 classes to help the model identify OOD data effectively. Furthermore, foundation models, which leverage vast amounts of pre-learned knowledge, can enhance the judgment of OOD data [55].

These techniques are effective in addressing semantic distribution shifts. However, we should adopt a more nuanced perspective when it comes to covariance shifts. Covariance shifts can offer certain advantages, which will be discussed in the next section.

3.3.2. The Impact of OOD Data

Augmented data as OOD samples

It is well-established that data augmentation is essential for training neural networks, especially when larger datasets are required to fine-tune the numerous parameters. Data augmentation methods such as flipping, cropping, scaling, and rotation are employed to overcome this limitation. Significantly, some advanced data augmentation methods like CutMix [67] will play more effect: The CutMix supposes we have two images, x_A and x_B. CutMix randomly selects a rectangular region from x_A and replaces it with the corresponding region from x_B.

Correspondingly, the labels y_A and y_B are mixed based on the area ratio of the mixed region. However, these data augmentation methods cause the training data to be out-of-distribution.

Yoshua et al. [20] were among the first to demonstrate that combining clean data with

perturbed data generated through data augmentation methods can help deep networks achieve human-level performance on the MNIST dataset. These perturbed data are typically generated using Gaussian smoothing or noise techniques. They argue that deep networks benefit significantly from these OOD examples. One possible explanation is that the lower layers of the network, which learn from both ID and OOD data, compute a hierarchy of features shared across tasks. This shared representation enhances the model's generalization ability.

Similarly, Geiping et al. [19] investigate the effectiveness of various data augmentation techniques. They use a metric to measure the amount of equivalent original data added by different augmentation methods. Their experiments show that for small-scale datasets (around 50,000 samples), data augmentation can effectively double the amount of valid data. However, data augmentation has a diminished effect on larger datasets (around 250,000 samples). Nonetheless, they acknowledge that some augmentation methods, such as rotation combined with random cropping, can introduce near-OOD data compared to the original dataset.

To investigate this, they trained a ResNet-18 architecture by randomly sampling rotations from each class to serve as training data and using a disjoint set of rotations as test data. They also employed horizontal flip and random crop augmentations to generate OOD samples, as these transformations cannot be produced through simple rotations. The experimental results show that OOD data improve classifier performance on rotated images. This improvement can be attributed to two factors: first, OOD data help the model learn invariant features, and second, they introduce stochastic factors during gradient descent, which can help the model avoid local minima by improving the optimization process during training.

Further Investigations into OOD Data Effects

However, this work provides only a preliminary conclusion regarding the benefits of OOD data, as it studies the effect of only one type of OOD data and does not explore its impact. Similarly, Silva et al. [18] conducted experiments to further investigate these effects. They studied two types of OOD data. Similar to [19], the first type involved using rotated or blurred samples as OOD data but with a quantitative evaluation of varying rotation angles and blur strengths. The second type used different classes, treating some as ID data and others as OOD data.

In their experiments, Silva et al. explored various OOD and ID data mixtures, keeping the amount of ID data constant while increasing the amount of OOD data. They found that if the domain shift was sufficiently small, the generalization error on the original test set consistently decreased as the ratio of OOD data to ID data increased. This finding aligns with the synthetic data generation goal and the [19] results, indicating that near-OOD data can positively influence the training process. However, the test error displayed a non-monotonic trend when rotation angles or blur strengths were increased. This trend is shown in Figure 3.7. In these cases, a small amount of OOD data was beneficial, but it became harmful as OOD data began to dominate the training distribution. This trend was also observed when using OOD data from different categories. Additionally, when OOD and ID mixture training was

applied to different domain tests, such as the DomainNet dataset, the OOD data from other domains followed a similar non-monotonic trend. They also experimented with the effect of different ratios of OOD data to ID data within a mini-batch. Their findings indicate that varying the proportion of OOD samples in each mini-batch influences the gradient optimization process, which in turn impacts the model's performance.

So far, we have discussed several works that examine the impact of OOD data in the training process. While OOD training data comes from different sources than the original training data—leading to distribution misalignment with the original training and test data—these studies show that if the degree of OOD data is controlled or kept within a certain ratio, it can improve a model's generalization ability. However, most of these experiments have been conducted using simple architectures, such as ResNet, and on essential datasets like MNIST, PACS, or CIFAR-10. In our work, we aim to expand the study of OOD data to more complex scenarios using diffusion models, and we will leverage more advanced architectures like the CLIP model to further explore the effects of OOD data.

3.4. Image Editing Techniques

3.4.1. Traditional Methods

Figure 3.8.: The image restoretion effect from the original paper [68, 69, 70]. We could see the MAT method could handle a very large missing area.

The following two approaches achieve effective outpainting without relying on inpainting techniques based on Stable Diffusion. They do not require any text prompts, only an input mask. LAMA [68] utilizes Fast Fourier Convolutions (FFC) [71] to maintain high efficiency while enlarging the receptive field, allowing for a global understanding of the inpainting region. An improved strategy for generating large masks during training activates the network's potential for better reconstruction.

Efficient long-range information interaction plays a crucial role in image restoration. Previously, methods capable of modeling long-range dependencies were limited to low-resolution images. MAT [69] is the first Transformer-based model that directly handles high-resolution image restoration. It employs dynamic masking to identify effective tokens for efficient long-range dependency modeling, making it suitable for repairing large missing areas. Our

tests show that MAT performs well even on datasets outside the training data.

There are follow-up works such as DiffIR [70] and DiffRIR [72], which combine diffusion models with LAMA for image restoration tasks. However, our tests indicate that their generalization performance on out-of-training data is not as strong as the aforementioned baseline models. Some generated samples could be found in Figure 3.8.

3.4.2. T2I SD Model for Image Editing

Inpainting is a fundamental technique in image editing, with outpainting as a related method. Outpainting extends the boundaries of an image to generate content beyond the original input, often used to complete or expand backgrounds. While some models [73] are specifically designed for outpainting tasks, they generally require retraining for new samples and are tailored solely for outpainting.

Diffusion Inpainting Models

Diffusion Inpainting [74], on the other hand, is an image restoration technique based on diffusion models that fills or repairs missing regions in an image. To enable image editing using the T2I SD model, we could:

1. **Fine-Tuning a Pretrained Stable Diffusion Model**: In this approach, the original image, the masked image (with the masked regions set to zero), and the corresponding mask are input into the SD model. The SD model is fine-tuned using the target masked image as the training target. SDXL inpainting [75] adopts this strategy and fine-tunes the model using large-scale datasets such as LAION-2B [75].
2. **Feature-Level Inpainting Without Fine-Tuning**: In this method, fine-tuning of the SD model is not required. Instead, at each step, the features at the latent layer corresponding to the unmasked regions are replaced with the noisy features from the original image at the current timestep. The features of the unmasked regions are determined by the input image, while the masked regions undergo reconstruction by calculating the noise with the UNet. This approach effectively restores the masked area by leveraging the features from the original image.

Normally there are several important parameters for inpainting models to consider during usage:

- **Guidance Scale**: This parameter acts as the weighting factor for classifier-free guidance. Higher values make the output more closely match the input conditions (e.g., text prompts).
- **Strength**: This parameter determines the level of noise added to the base image, influencing how closely the inpainted region resembles the original. For instance, if the strength is set to 0.8 and the total number of steps is 20, then noise will be added for 0.8×20 steps, while the remaining 4 steps will not add noise. This can be interpreted

as scrambling 80% of the features while retaining 20%. If the strength is 1, the original image is completely disregarded, while if it is 0, the output remains identical to the original.

- **Number of Inference Steps**: This parameter controls the number of denoising steps performed during image generation. More steps result in higher image quality, allowing for more refinement, but increase the inference time. For example, if the strength is set to 0.8, only 0.8×20 denoising steps are required to restore the image.

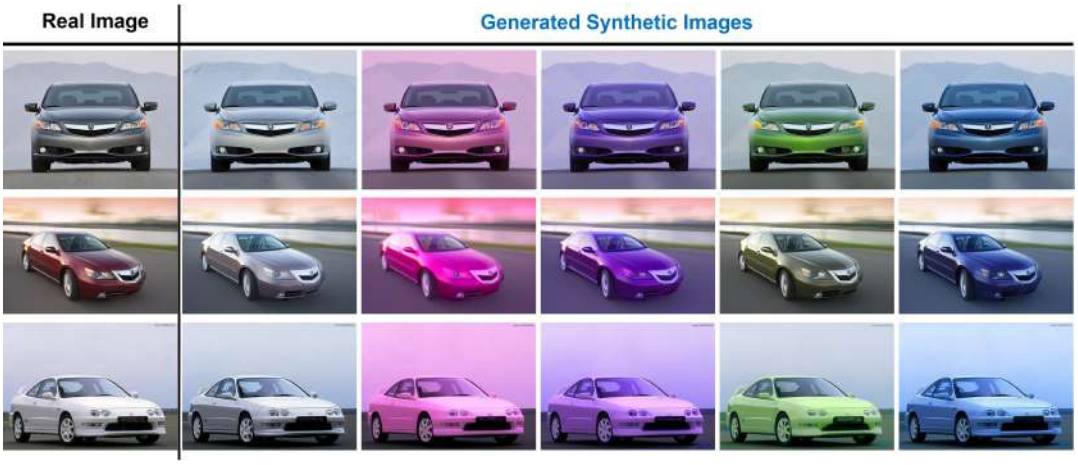

Figure 3.9.: Our image color editing experiments using FPE method on Oxford-Cars Dataset [76].

Specialized Methods

FPE [77] is an image editing method based on attentive maps, improving upon the p2p [32] approach without requiring task-specific retraining of the original SD model. In p2p, it was observed that the cross-attention layers exhibit representations of relevant tokens, with attention maps specific to certain tokens appearing even in the early cross-attention layers.

The FPE method builds on this by discovering that self-attention maps contain structural and spatial information, while cross-attention maps retain class-related attributes. Therefore, when editing the color or texture of an object while preserving its class-specific features, FPE replaces only the self-attention layers corresponding to the target color or texture. This selective replacement yields more meaningful edits to the desired attributes.

In Figure 3.9, we demonstrate color modification experiments on images from the Stanford Cars dataset [76]. The method achieves excellent results, particularly when editing light-colored images, where the modifications appear both accurate and visually consistent with the original object.

InstructPix2Pix [78] is an instruction-based image editing model that enables specific editing operations on images using natural language commands, such as changing colors,

adding objects, or adjusting shapes. This model is built on top of the SD 1.5 model and fine-tuned on a large-scale image editing dataset. It can handle various editing tasks without requiring separate training for each task. By combining multimodal features from text and images, InstructPix2Pix achieves automatic image editing based on instructions, significantly reducing user effort.

However, due to the lower version of the SD model and the limitations of the training dataset's domain coverage, real-world performance on user-provided data is often less impressive compared to the examples shown in the paper.

Several follow-up works have expanded on InstructPix2Pix, such as:

- **Hive [79]**: This approach uses the same prompt-to-prompt technique to generate more training data, extends the base model to SD v2.1, and employs reinforcement learning-based methods to train the image editing model, yielding better editing results compared to InstructPix2Pix.
- **MGIE [80]**: This work uses the same training dataset and SD 1.5 base model but emphasizes the limitations of using CLIP's text encoder alone for understanding complex modification instructions. Since CLIP employs static captions, MGIE incorporates language augmentation with Multimodal Large Language Model(MLLM).

Despite these improvements, these subsequent models are larger and may struggle with generalization on untrained datasets. Like our experiment results in Figure 3.10.

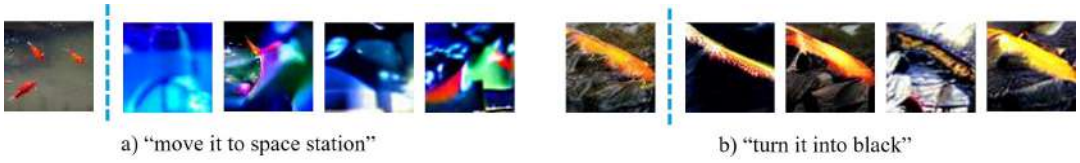

Figure 3.10.: Our experimental results using InstructPix2Pix on Tiny-ImageNet [53].

3.5. Large Language Models and Vision-Language Models

As discussed earlier, various models have been employed to improve synthetic image generation, including methods for prompt augmentation, enhanced language generation, and filtering generated samples. Given the rapid advancements in this field, providing a comprehensive overview is beyond the scope of this work. Instead, we will introduce a few selected representative models to facilitate a better understanding.

BERT

BERT [84] is based on a Transformer encoder architecture and was the first to introduce the Masked Language Modeling (MLM) loss for self-supervised pretraining. The architecture is the encoder part of Figure 3.11 a).The MLM task involves randomly masking approximately

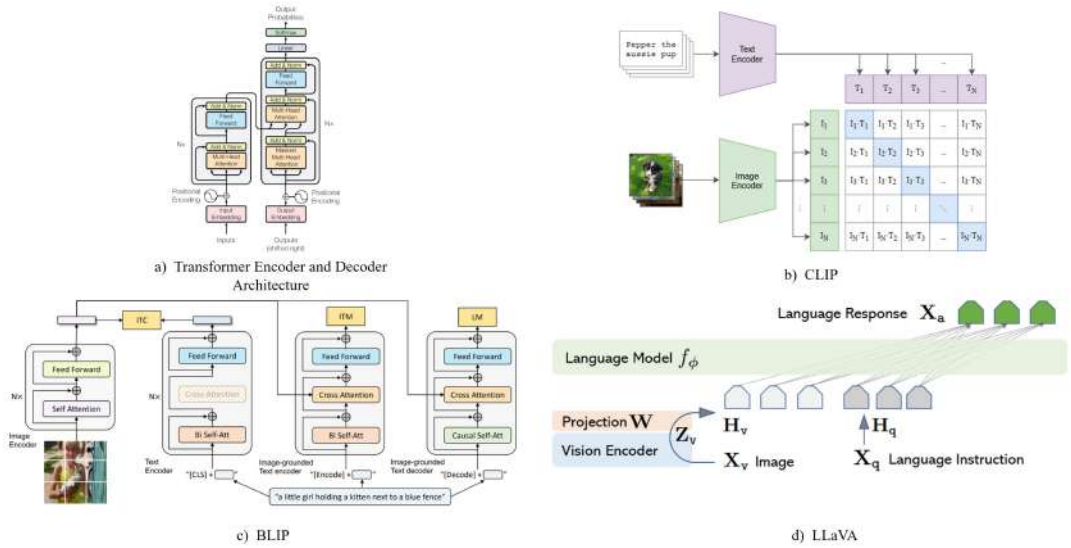

Figure 3.11.: The representative methods including a) Transformer [81] architecture, b) CLIP [21] , c) BLIP [82] , d) LLavA [83].

15% of the input tokens and predicting the original tokens from the masked ones. The corresponding loss is calculated as:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(x_i | \hat{x}), \tag{3.2}$$

where $\mathcal{M}$ is the set of masked positions, x_i is the original token, and $\hat{x}$ is the corrupted input sequence.

BERT also employs the Next Sentence Prediction (NSP) loss to determine if two sentences are consecutive. The NSP loss helps the model learn relationships between sentences. Due to its encoder-based architecture, BERT is more suitable for extracting textual features.

T5

T5 (Text-to-Text Transfer Transformer) [85] is an encoder-decoder Transformer model that generates output sequences in an autoregressive manner, using the standard cross-entropy loss. The architecture is the Figure 3.11 a).

The core idea of T5 is to unify all NLP tasks (such as translation, question answering, and text classification) into a single text-to-text format, enabling cross-task learning across various natural language tasks. It is pretrained on large-scale text data using a fill-in-the-blank objective and then fine-tuned for specific tasks, providing strong generalization capabilities. Compared to BERT, T5 is more widely used in dialog tasks.

CLIP

CLIP (Contrastive Language-Image Pre-training) [21] is a multimodal model with a dual-tower structure, where a Transformer-based encoder processes both the entire image and its corresponding textual description. Using contrastive learning, CLIP maps images and text into a shared embedding space, allowing them to be matched via cosine similarity. The objective function, known as the InfoCSE loss, maximizes the similarity of correct image-text pairs while minimizing the similarity of incorrect pairs:

$$\mathcal{L}_{\text{InfoCSE}} = -\log \frac{\exp(\text{sim}(v,t)/\tau)}{\sum_j \exp(\text{sim}(v,t_j)/\tau)}, \tag{3.3}$$

where $\text{sim}(v,t)$ is the cosine similarity between image embedding v and text embedding t, and τ is a temperature parameter.

CLIP is pretrained on a large number of image-text pairs, resulting in strong zero-shot classification and retrieval capabilities. The method is the Figure 3.11 b).

LLaMA

LLaMA (Large Language Model Meta AI) [86] is a well-known open-source lightweight LLM designed to reduce the training and inference costs of large language models. It employs a Transformer decoder-only architecture, with several optimizations to improve training stability, such as using pre-normalization (instead of post-normalization).

Additionally, LLaMA replaces absolute positional encoding with RoPE (Rotary Position Embedding). LLaMA-7B surpasses GPT-3-175B [3] on most benchmarks and exhibits competitive performance compared to other larger models, making it suitable for text generation, dialogue systems, and question answering. It strikes an excellent balance between computational cost and performance.

Vicuna

Vicuna [87] is an open-source dialogue model fine-tuned from LLaMA-13B, achieving performance close to ChatGPT. The training data consists of 70K user conversations. The success of Vicuna is attributed to fine-tuning on user feedback data, which enhances the model's ability to understand and generate natural language dialogues.

BLIP

BLIP (Bootstrapping Language-Image Pre-training) [82] is a multimodal pretraining framework that introduces cross-modal encoders and decoders to enable the flow of information across modalities. The term "bootstrapping" refers to its iterative training process, where the training data—composed of noisy web-sourced image-text pairs—undergoes filtering to improve quality. A filter model is used to remove inappropriate labels, while a captioning model generates captions for web images, and the refined data is then used to further train the original model. They use Image-Text Contrastive Loss (ITC), Image-Text Matching Loss (ITM)

and Language Modeling Loss (LM) to gain a strong understanding of both visual and textual information, making it suitable for tasks such as text generation, image-text alignment, and retrieval. The loss could be found in the top of the Figure 3.11 c). Due to these advantages, BLIP is widely used as a captioning model in various applications.

LLaVA

LLaVA [83] is a prominent model for multimodal learning and currently stands out among open-source multimodal models in terms of performance. Its core design utilizes an advanced pretrained image feature extractor, CLIP ViT-L/14, and a large language model (LLM) for information fusion and text output. A lightweight projection layer is incorporated to facilitate the reuse of pretrained image and text models. The architecture could be found in the Figure 3.11 d).

The training process consists of two stages:

1. **Alignment**: Similar to CLIP, LLaVA first aligns image features to the text feature space.
2. **Fine-tuning**: The model is then fine-tuned on a specialized multimodal instruction-based question-answering dataset, enabling it to handle multimodal dialogue tasks more effectively than models using text alone.

Through continuous improvements in data quality and scale, and iterative experimentation with LLM architectures (e.g., using Vicuna-1.5 with 7B and 13B parameters), LLaVA has evolved to versions like LLaVA 1.5 [88] and LLaVA Next [89].

InternVL

InternVL's [4] architecture is similar to LLaVA, using a ViT-based image feature extractor and LLaMA as the LLM. However, it differentiates itself through a broader range of training datasets, an enlarged visual encoder, and a more sophisticated projection layer called QLLaMA. Compared to traditional multimodal models, InternVL can handle more complex multimodal tasks with superior performance.

ChatGPT

ChatGPT [3], developed by OpenAI, is a large-scale conversational model based on GPT-3. It is fine-tuned for open-domain dialogue, enabling coherent and natural conversation generation. The GPT model is a Transformer-based language model trained in an autoregressive manner to predict the next token given the previous tokens, using the language modeling loss.

With models such as GPT-3.5 and beyond, additional supervised fine-tuning with instructions and Reinforcement Learning from Human Feedback (RLHF) [90] are used to improve model performance. The parameter size of GPT-3 is 175B, and it utilizes extensive training data.

While ChatGPT's API allows for large-scale data generation, it is not an open-source model, and usage involves associated fees.

3.6. Grounded Mask Generation

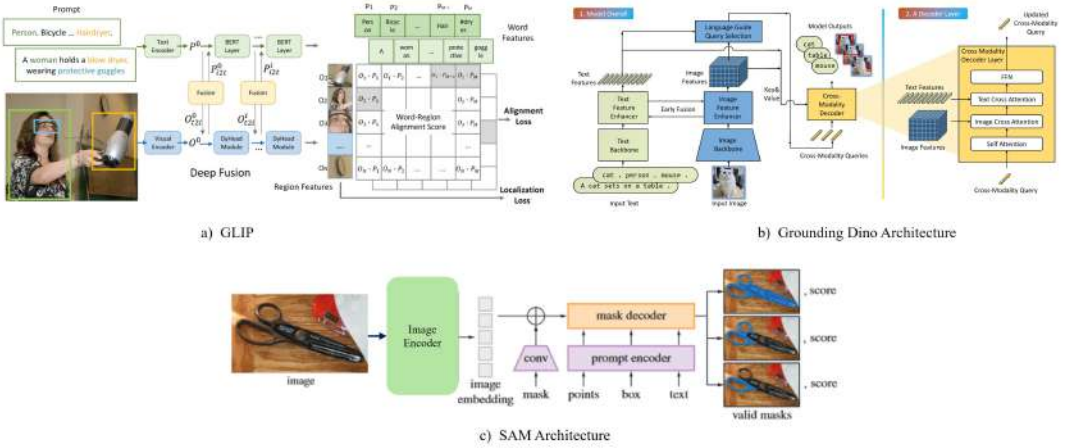

Figure 3.12.: The architecture for BLIP, GroundingDino and SAM [91, 92, 6].

The first stage of grounded mask generation usually is grounding detection. Grounding detection aims to perform detection in an open-world setting, where the goal is to complete zero-shot detection tasks. Then we could using the detection results for the downstream open-set segmentation task. We introduce several representative models below.

GLIP

GLIP(Grounded Language-Image Pre-training) [91] follows a similar approach to CLIP but extends it to the more fine-grained task of open-world detection. While CLIP matches an entire image with its corresponding text, GLIP calculates the similarity between each detected bounding box and all object category texts. Like CLIP, it constructs positive and negative samples for contrastive learning. The model is shown in Figure 3.12 a).

GLIP employs an interactive model structure that uses self-attention and cross-attention mechanisms to integrate and fuse features from both text and image inputs, thereby enhancing learning. In GLIP-v2, an additional masked language modeling loss is introduced, allowing the model to achieve better fine-grained understanding of images and corresponding text, enabling it to perform tasks such as multimodal question answering.

GroundingDINO

GroundingDINO [92] is a model designed for open-world object detection and annotation. It combines object detection with language understanding to perform natural language-based object detection for previously unseen objects or categories. Inspired by the success of the GLIP model in open-world detection and the pretraining of DINO in the detection domain, GroundingDINO leverages a Transformer-based detector architecture, which has proven

effective for handling large datasets and integrating features from text and vision Transformer models.

GroundingDINO utilizes a Swin Transformer-based [93] visual feature extractor, similar to DINO, and employs a BERT model [84] as the text feature extractor. To enhance the model's language understanding capabilities, the authors designed a text-image fusion layer based on deformable attention (similar to the self/cross-attention layers in GLIP), as well as a language-guided visual feature selection module and a decoder for detection tasks. The details are shown in Figure 3.12 b).

In addition to the aforementioned models, open-world detection remains an active research field. Approaches like YOLO-World [94] extend the YOLO model for detecting bounding boxes in open-world settings. Similar to contrastive learning, they compute the similarity between the image features within bounding boxes and corresponding text features, enabling zero-shot detection.

SAM Series

SAM(Segment Anything Model) [6] is a general-purpose model for image segmentation that allows segmentation through various prompts, including point prompts, box prompts, and mask prompts. It uses a ViT-based image encoder, with prompts having their encoder. In the decoder, different features are fused through cross-attention, and the output is produced by a prediction head. The details are shown in Figure 3.12 c).

Compared to traditional segmentation models, SAM don't rely on specific category labels or densely annotated data, significantly enhancing the generalization and usability of segmentation tasks. SAM is pretrained on a large-scale image dataset with masks, enabling it to handle images of various types and resolutions. It can generate fine-grained segmentation results, not limited to specific object categories. SAM2 [95] uses more training data and could achive a higher segmentation accuracy, also the model could expand to video segmentation.

By feeding the bounding boxes obtained from models such as GroundingDINO into SAM as prompts, zero-shot mask generation can be achieved using just the corresponding class names. This combination is commonly known as the GroundingSAM [96] model. In practice, this combination performs well, and we adopt this approach in our work.

RMBG Model

The RMBG model(Remove Background Model) [97] is specifically designed for removing backgrounds from images, preserving the foreground objects. It typically uses a U-Net or similar encoder-decoder architecture and is optimized with a cross-entropy loss function to improve foreground-background segmentation accuracy.

During training, the model learns to accurately distinguish between foreground and background from a large set of annotated image data. Compared to traditional background removal methods, the RMBG model handles complex backgrounds more precisely, especially when dealing with hair, semi-transparent objects, and other fine details. Its results are comparable to those achieved by models like GroundingDINO.

3.7. Parameter Efficient Fine-tuning Methods

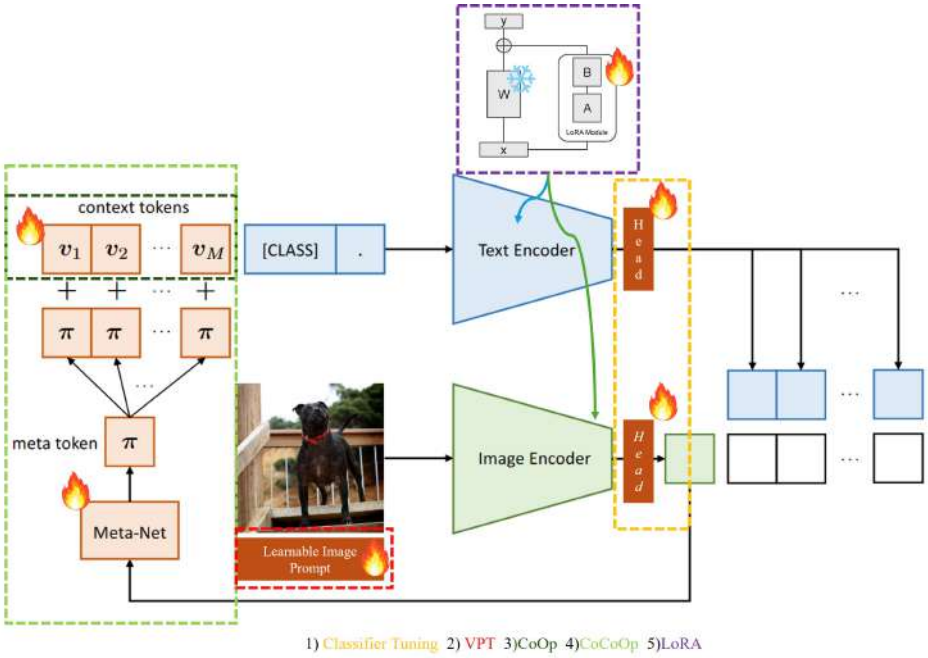

Figure 3.13.: The visualization summary for various fine-tuning methods, different method is highlighted by different color.

Classifier Tuning

Classifier Tuning [98] is a strategy focused on adjusting only the parameters of the classifier, typically used in transfer learning and model fine-tuning. When the feature extractor of a model (such as the initial layers of a CNN or Transformer) has been pre-trained on a large-scale dataset, Classifier Tuning keeps the parameters of the feature extraction layers fixed, while only fine-tuning the final few layers of the classifier to adapt to the target task. The method is highlighted in Figure 3.13 in yellow.

Since it adjusts only a small number of parameters, this approach is more efficient in terms of training time and computational cost. Compared to training a model from scratch, Classifier Tuning is more effective on small datasets, as it leverages pre-trained features to enhance the generalization ability of the classifier. By fine-tuning only the classifier layers, it allows for subtle adjustments specific to the task while maintaining the feature extraction capabilities, reducing the risk of overfitting.

TPT

Although Classifier Tuning can achieve good results, it may cause the model to lose its original generalization ability. TPT (Test-Time Prompt Tuning) [99] optimizes the text input

during test time based on a single test sample, without requiring any labels or additional data outside the zero-shot test samples. For image classification tasks, a unique prompt vector is introduced for each task, which can be dynamically adjusted during the model's forward pass.

Since no labels are needed, an unsupervised loss function is designed, where different views of the input image are generated by appropriate cropping, avoiding noise from random augmentation. The loss function is defined as the consistency between the predictions of different views of the test sample, minimizing the entropy of the average prediction probability distribution. This optimizes the learnable vectors added to the text input.

VPT

VPT(Visual Prompt Tuning) [100] is a method similar to Prompt Tuning in the text domain, but applied to the visual domain. It introduces a small number of trainable visual prompts (prompt tokens) that are added before the input image features, resembling text prompts in natural language processing. The prompts can be added to the first layer of the feature input, or to each layer.The method is highlighted in Figure 3.13 in red.

In experiments, some benchmark settings show that VPT can outperform full-parameter fine-tuning. The improvement of VPT over full fine-tuning is more significant in scenarios with limited data, and it still performs well even when the dataset size increases compared to other fine-tuning approaches.

CoOp

To effectively apply pre-trained vision-language models to downstream tasks without extensive fine-tuning, prompt engineering can be used, but it often requires expert knowledge and time-consuming design. CoOp (Context Optimization) [101] transforms fixed text input into learnable prompts combined with the class names, allowing the model to learn task-specific textual inputs. Experiments show that CoOp significantly improves performance over zero-shot learning with only a few downstream samples. The method is highlighted in Figure 3.13 in green.

CoCoOp

While CoOp performs well on downstream tasks, it tends to overfit and exhibits poor performance on unseen classes. To address this, CoCoOp [102] extends CoOp by adding a lightweight network (Meta-Net) that incorporates instance-level tokens obtained from the image encoder, enhancing CoOp's adaptability during training. However, in practice, CoCoOp sometimes underperforms compared to the original CoOp on seen classes, despite its improvements on unseen classes, and incurs higher computational costs.

LoRA

LoRA(Low-Rank Adaptation) [103] is a low-rank adaptation method for efficient fine-tuning of large models. It introduces low-rank decomposition into the pre-trained weight matrix to reduce the number of parameters that need to be fine-tuned, making it suitable for task adaptation in large models. The method is highlighted in Figure 3.13 in purple.

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{3.4}$$

where:

- $W_0 \in \mathbb{R}^{d \times d}$ represents the pre-trained weights.
- $\Delta W = BA$, with $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$.
- $r \ll \min(d, k)$, ensuring that the rank of the adaptation is low.
- A is randomly initialized, while B is initialized to zero, making AB zero at the beginning of training.

LoRA adjusts only a small number of parameters instead of the entire model, significantly reducing the computational resources required for fine-tuning. Compared to traditional fine-tuning methods, LoRA maintains performance similar to full fine-tuning while optimizing a much smaller number of parameters.

In addition to the common methods discussed above, there are other approaches, such as LogoPrompt [104]. This method involves overlaying the text describing a new concept onto the corresponding image of the concept and inputting this combined data into the SD model. The text prompts are similar to CoOp, includes learnable parameters, allowing for generalization to small sample data during training on new concepts.

4. Approach

In this section, we will first provide a comprehensive task formulation in Section 4.1, followed by an introduction to our MCSDG pipeline in Section 4.2. A detailed explanation of how we validate the effectiveness of feasible(ID) and infeasible(OOD) data will be presented in Section 4.3.

4.1. Task Formulation

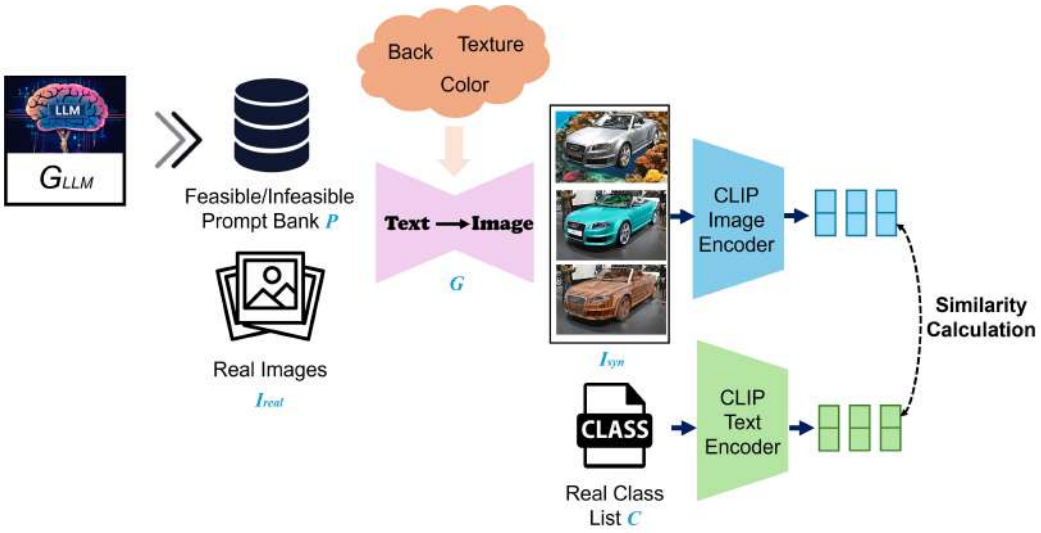

Figure 4.1.: Overview of our method, including MCSDG pipeline and CLIP Training.

For a dataset which consists total C (C) categories, our goal is to utilize an image classifier to analyze the impact of and the feasible (ID) or infeasible (OOD) data I_{Syn} corresponding to each individual class c_i, where $i = \{1, ..., C\}$. The feasible or infeasible images I in D_{SynB}, D_{SynC} and D_{SynT} are generated by a text-to-image generator G, based on specific prompts P generated by G_{LLM}.

4.1.1. Data Generation Strategy

Feasible ID data is generated using feasible prompts P_{ID}, where the prompt set $P_{ID} \in \{P_{fb}, P_{fc}, P_{ft}\}$. Conversely, OOD data is generated using infeasible prompts P_{OOD}, where the prompt set $P_{ood} \in \{P_{ifb}, P_{ifc}, P_{ift}\}$. The subscripts means:

- Background B_i: The background of a image for the category c_i.
- Color $C(Color_i)$: Represents the color of the primary object in a image of category c_i.
- Texture T_i: Defines the material properties associated with category c_i. Note that texture properties also encompass color characteristics.

4.1.2. Verification by A Classifier

The designed classifier consists of the following components given C categories:

- Text Encoder: $v_{text} = f_\theta(T)$ maps the category text input $t_i \in T$, $i = \{1, ..., C\}$ to a vector representation $v_{text} \in \mathbb{R}^{C \times d}$.
- Image Encoder: $v_{image} = f_\theta(I)$ maps the image I to a vector representation $v_{image} \in \mathbb{R}^d$.
- Classifier: $y = m(v)$, which computes the cosine similarity between the image features v_{image} and all text features v_{text} to generate a probability distribution over the N categories.

  The classification result is computed using the following cosine similarity formula:

$$\text{CosSim}(v_{image}, v_{text}) = \frac{v_{image} \cdot v_{text}}{\|v_{image}\| \|v_{text}\|}$$

The whole process is visualized in Figure 4.1.

4.1.3. Training and Evaluation Strategy

Specifically, the classification model is trained and evaluated under three different settings:

- Training with only the real dataset: Using real images for training.
- Training with only the synthetic dataset: Using the generated synthetic datasets for training.
- Mixed training: Combining real data with synthetic data for training.

By analyzing the classifier's performance under these different training conditions, we explore how feasibility influence the model's classification capabilities.

4.2. Generating Minimal-Change Synthetic ID and OOD Data

Due to the lack of an appropriate real-world minimal attribute modifications pairs, we use the T2I generator G mentioned in the section 4.1 to construct our MCSDG pipeline.

4.2.1. Generate Guidance Prompt

To generate feasible or infeasible images, we need to provide corresponding feasible prompts P_{ID} and infeasible prompts P_{OOD} for each image. The core of the prompt P lies in creating corresponding prompt words W that describe the background B, color C and texture T for each category c_i.

To generate as many accurate feasible and infeasible prompts as possible, we utilize an advanced language model, ChatGPT-4 [3]. To avoid errors or repetitive content, we employ In-Context Learning[3], providing the model with positive examples $Example+$ and negative examples $Example-$ to help the model better understand the task, and using a template to clearly define the output format.

Some generated prompt words W may contain overly abstract nouns. For infeasible backgrounds in the pets category, the model might generate phrases like "freezing tundra" or "deep sea." Such prompts can result in images lacking the desired level of background detail and the final output will have vague background structures. To improve the fine-grained detail and realism of the generated backgrounds or textures, we instruct the model to append a brief explanatory description when generating prompts, providing more detailed guidance for image generation.

An example of our generated prompts is as follows: the $[Attribute]$ represent the feasible/infeasible background/color/texture, $[CLASS]$ represent a specific class c_i:

Prompt Example. *"Task: As an AI language model, generate [Attribute] where the given class of objects typically exists ('feasible') and where they absolutely cannot exist ('unfeasible'). For each [Attribute], provide a one-sentence description detailing its visual appearance. You should adhere to the specified criteria.*
Criteria:

- ...

Positive Example:

- **Object Class:** [CLASS]
- **Question:** Provide five different [Attribute] for the object class, each accompanied by a concise visual description.
- **Answer:**
  - ...

Negative Examples:

- The answers are not acceptable as follows:
  - ...
- **Reasons:** ...

Question: Please give me [NUMBER] different [Attribute] for the class [CLASS]; in the meantime, also give me corresponding detailed descriptions for the given [Attribute].

Figure 4.2.: The prompt example for ChatGPT [3].

Using the above prompts, we can obtain many preliminary feasible P_{ID} and infeasible P_{OOD} prompts. Although large language models possess broad knowledge across various domains, they may sometimes struggle to accurately distinguish between feasible and infeasible attributes for a specific category c_i. For instance, ChatGPT might generate "yellow" or "blue" as feasible colors for the airplane category "737-500". Due to these colors not appearing in its real training set, they should be treated as "infeasible colors." If such prompts were used in generation, our feasible synthetic dataset would instead contain infeasible samples.

To address this issue, we design additional prompts to instruct the model to perform preliminary checks and filtering on its outputs. The related process is illustrated in Figure 4.3. Specifically, for the attributes of a given category, the model filters out infeasible features that do not align with the real attributes, ensuring that the generated prompts meet our requirements. With the following prompt:

```
"Can you modify or filter your answers to ensure each
[background/color/texture] is definitely [feasible/infeasible] for
class [CLASS]? Please delete and ignore some of the answers if you
can't guarantee them."
```

	Background						**Color(Per CLS)**						**Texture**					
	Pets		AirC		Cars		Pets		AirC		Cars		Pets(Per CLS)		AirC		Cars	
	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF
Raw output	50	70	50	70	50	70	10	10	10	10	10	10	8	50	30	50	15	70
Auto-filtering	47	64	36	68	44	67	6∼7	8∼9	7∼8	8∼9	7∼8	8∼10	7	42	25	46	12	64
Manual-filtering	43	50	22	50	31	50	5	5	5∼8	5∼6	5	5	5	27	24	44	7	57
Final Accept Rate	0.86	0.714286	0.44	0.71429	0.62	0.71429	0.5	0.5	0.5∼0.8	0.5∼0.8	0.5	0.5	0.625	0.54	0.8	0.88	0.467	0.814

Table 4.1.: The number of prompts which are generated initially by LLM, after self-filtering and manual-filtering for each specific settings and some datasets. The Pets, AirC, Cars refer to our experimental dataset introduced in 5.1.

Before finalizing the prompts, manual verification and confirmation are conducted to ensure that the prompts input into the SD model strictly adhere to our requirements. Table 4.1 lists the number of the initial and final confirmed prompts for each dataset. More details could be found in Section A.1 and Section A.2.

4.2.2. Vanilla Generation Approach

To generate minimally altered synthetic images $I = g(P|I_{real})$ based on text prompts P while constrained by real images I_{real}, we design a simple baseline workflow. First,

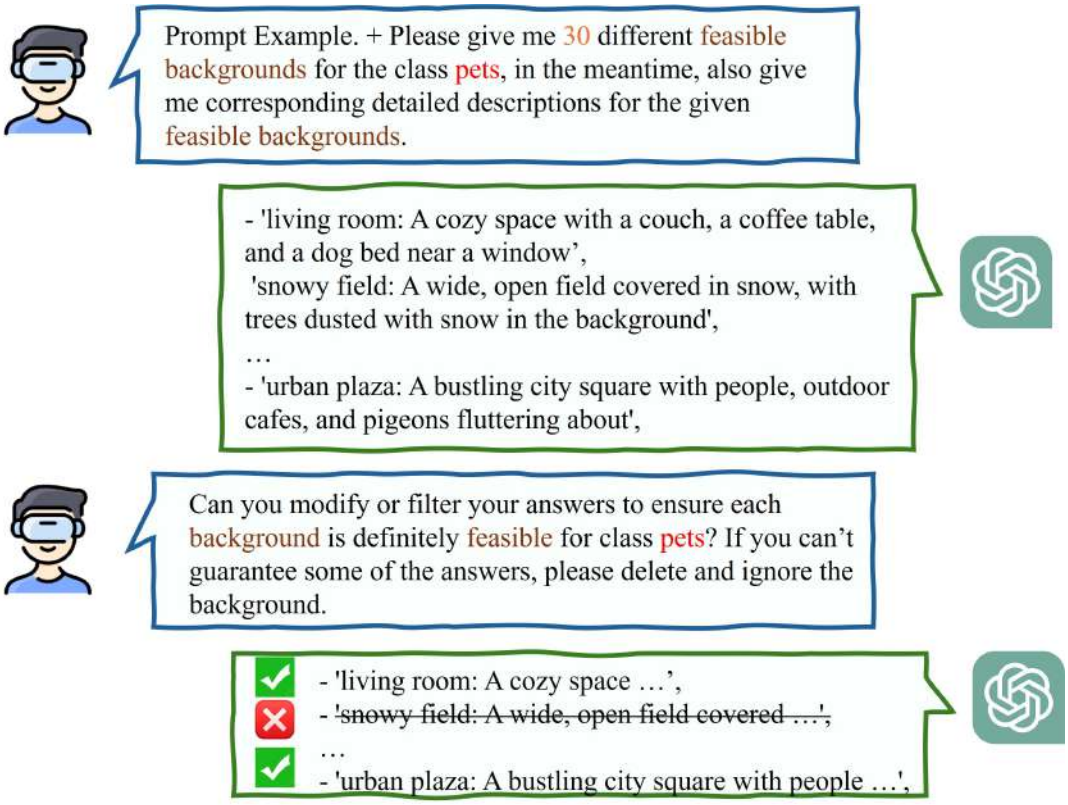

Figure 4.3.: The prompt words generation and self-filtering process using ChatGPT-4 [3].

we use a fundamental prompt P_{base} = "a photo of a [CLS]" and then make minor adjustments based on the target attributes (background, color, texture). The final prompt is $P = P_{base}.replace([WORD], w)$, as illustrated in Figure 4.4.

Generation Strategy

Since the feasible or infeasible prompt words W generated by ChatGPT might not belong to the Generator model G's training distribution, our goal is not to propose a new image attributes editing method but rather the stable generation of a minimally altered dataset. So we would like to develop an automatic generation pipeline as simple as possible. As discussed in Section 3, several existing text-based image editing methods can intuitively achieve our goal. We conduct preliminary experiments using the following text-based image editing models: InstructPix2Pix model [78], FPE model [77], SDXL inpainting model [74], and ControlNet model [40].

Mask Generation Workflow

The above methods like SDXL inpainting requires mask to edit the image, we design a mask generation workflow based on grounding. This workflow is illustrated in Figure 4.5. It

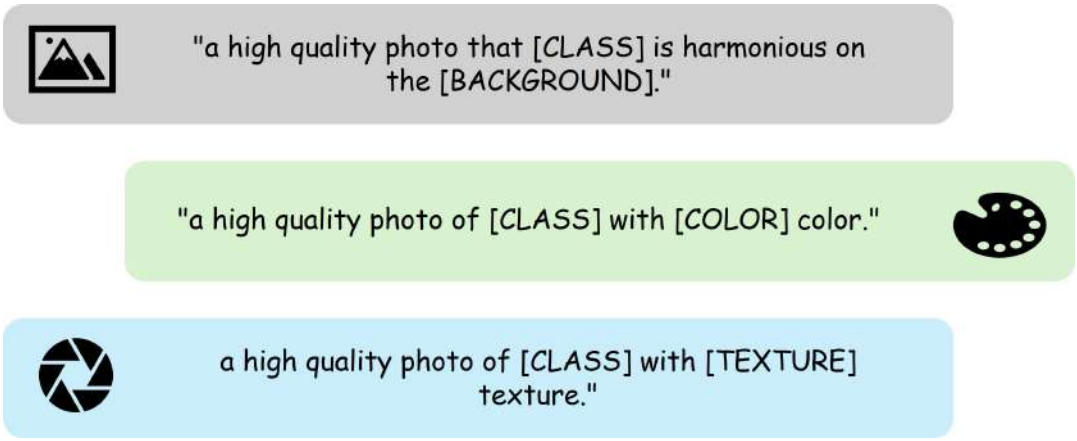

Figure 4.4.: Final prompt templates for SD model for background, color and texture settings.

uses Grounding Dino [92] to generate bounding boxes $bbox_i$ based on the category label c_i, and these bounding boxes are then fed into the SAM2 [95] model to produce masks m_i corresponding to category c_i. For some samples where bounding boxes could not be generated, we use the RMBG1.4 foreground-background segmentation model as a fallback to ensure each sample have a corresponding mask. For ControlNet, we provide the Canny edge maps as conditional inputs. When modifying the background, to ensure the preservation of foreground structures, we first extract the foreground $Foreground_i$ from $mask_i$ and then generated the corresponding Canny map based on $Foreground_i$. We directly obtain a complete Canny map from the real image I_{real} as the conditional input for color and texture modifications. The corresponding used masks are visualized in Figure 4.6.

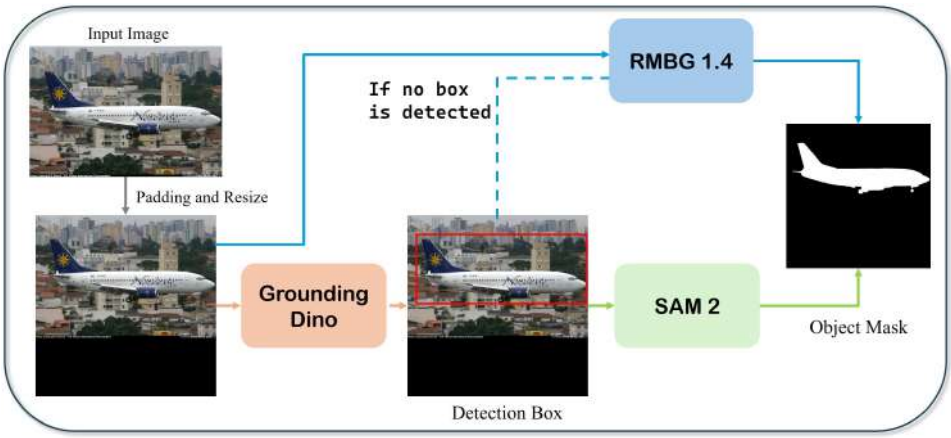

Figure 4.5.: The workflow of Grounding Mask to get arbitrary mask image.

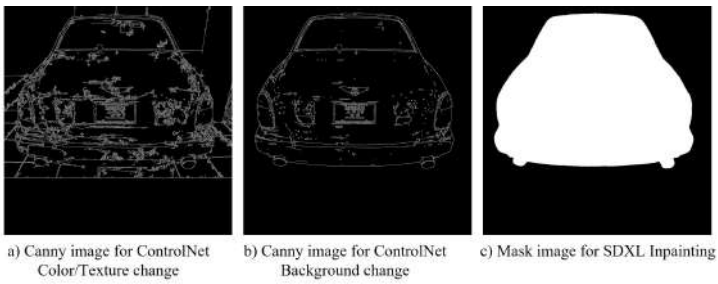

Figure 4.6.: The mask and canny images used in vanilla generation approach.

Analysis

We select three samples from different datasets and compare the results using these models. Since the results visualization takes up too much space, we choose the background setting to represent the editing results shown in Figure 4.7. The color and texture editing results comparison are in Figure B.1 and Figure B.2 in Section B.1.

All experiments use the exact feasible or infeasible prompt words W. For InstructPix2Pix and FPE, we use prompts similar to those in the original paper. Our guidance prompt template shown in Figure 4.4 is used for SDXL inpainting and ControlNet. Our Minimal-Change dataset has the following specific requirements: 1) When modifying the background, the generated image should reflect the feasible or infeasible background described by the prompt, maintain rich background details, and avoid giving the impression that the main object is "floating" on the new background; 2) When modifying the color or background, the shape of the main object should remain unchanged and consistent with the original background.

We observe the experimental results and find that InstructPix2Pix performs well on samples within its training data range. However, in many cases, it fails to make modifications, especially in background changes. For example, when modifying backgrounds for pets, InstructPix2Pix sometimes only outputs the background without preserving the subject, and overall, its performance across the three modification settings is poor.

The FPE method, on the other hand, is specifically designed to maintain the structure of the main object, making it unsuitable for background modifications. In our color modification results, shown in Figure B.1, FPE excels in preserving structure but struggles to modify original solid colors. For instance, FPE cannot change the color of intensely colored objects, such as black cars and blue airplanes.

SDXL Inpainting performs well for background modifications, especially with pets, where it meets our requirements in most cases. However, SDXL Inpainting struggles with airplane backgrounds, and while it generally produces natural-looking changes in color and texture, the modifications are often subtle compared to ControlNet. An explicit limitation with SDXL Inpainting is that the original image heavily influences the modifications; for example, when changing the color of a black car, the output often retains the original black color.

ControlNet, by contrast, is less affected by this constraint and demonstrates more effective

Figure 4.7.: The background attribute change setting results for four different base models.

color and texture changes. It meets our requirements, especially with infeasible airplane color changes and feasible texture transformations. However, one drawback of ControlNet is that objects sometimes appear to float in the background, affecting realism. While feasible color and texture changes are generally accurate, ControlNet struggles with specific infeasible attributes. For instance, when applying "mosaic tiles" as a texture to a car, the model does not accurately represent this texture, likely due to unfamiliarity with such attributes, resulting in incorrect outputs. As a result, we choose SDXL inpainting and ControlNet model for our next generation candidate models.

4.2.3. Prior-Guided Generation for Improved Control

To further enhance the model's performance, we propose a method based on prior information (Prior) to guide the model in generating content that meets the requirements. Specifically, we first use the Stable Diffusion model to generate raw background priors or raw texture priors based on the prompt W, guiding the modification of backgrounds and textures. For color modifications, we define a Color Bank from which the RGB values ValueRGB $\in$ ColorBank

corresponding to the prompt W are selected to generate the raw color prior.

Combining Priors with Real Images

When integrating generated images with real images, we first use the Groundingmask method introduced in Section 4.2.2 to generate masks for the subject or background. For background modification, we replace the background region of the original image with the generated Background Prior. To maintain a natural relationship between the subject and background in the generated image (e.g., ensuring that a pet remains grounded), we use a mask dilation method to expand the original ground truth mask, preserving the spatial relationship between the subject and the original background.

Figure 4.8.: Our Prior combination method with real image.

We apply the generated color or texture prior as an alpha channel (transparency channel) overlay on the subject in the original image for color or texture modifications. This approach provides the modified color or texture information while preserving the shape and structural details of the subject. The process is shown in Figure 4.8.

In the SDXL inpainting model, we use the above method to get a prior image called Prior. In addition to using the Canny edge map as a constraint for the ControlNet model, we also used an image IP-adapter [46] to input prior information. Two prior images were provided: one directly generated by Stable Diffusion called raw prior, and the other is the same Prior.

Analysis

This time we select color setting in Figure 4.9 here, the left generation results please refer Figure B.3 and Figure B.4 in Section B.1. Figure 4.9 compares the generated color setting results after introducing prior images. By incorporating prior information, generation quality has significantly improved compared to the first section. The model could receive the precise background, color, or texture we want to change to minimize the generation gap. For the SDXL inpainting model, the background editions now meet the expected requirements. The color and texture settings also perform very well; the main issue is that when the parameter "strength" is small, we cannot change the original color or texture. However, suppose we increase the value, although the target attributes can be generated. In that case, the details of the subject may inevitably be altered, such as the plane's engines or the car's rear being incorrectly modified to resemble the front. ControlNet ensures that the structure of the main object remains unchanged when modifying color and texture. However, the results

generated by ControlNet often appear less natural than those produced by inpainting, like for the car color change setting. In some cases, such as the color change for pets, even the feasible outputs appear less realistic than those generated without applying ControlNet's conditioning.

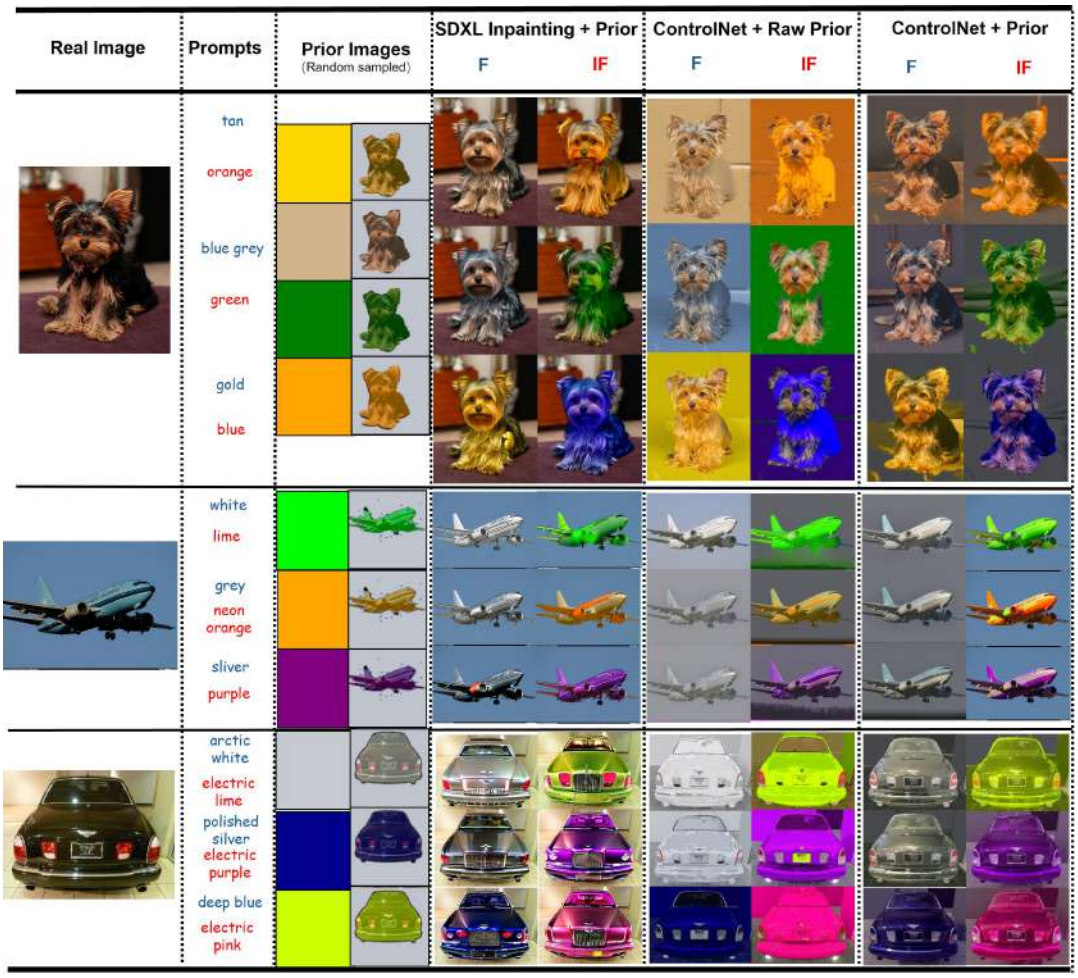

Figure 4.9.: The comparison results for SDXL inapinting and Controlnet with raw prior and prior images. We randomly select the raw prior and prior images on the left for visualization.

4.2.4. Ensuring Minimal Attribute Changes Data Generation Pipeline

Based on Section 4.2.3, we have demonstrated that introducing prior information significantly improves the success rate of attribute modification compared in Section 4.2.2. And for the background setting, the SDXL Inpainting has already fulfilled our requirements. From the experiments in Section 4.2.3, we observed that SDXL inpainting and ControlNet have their respective advantages for color and texture change. To ensure the generated image is

as natural as possible while the subject structure should not be changed, we combine the strengths of these two models by integrating both methods. Specifically, we first generate a refined image using SDXL inpainting based on the prior image for the color and texture settings. Then, this refined image is used as conditional input through the IP-adaptor [46] into ControlNet to produce the final output.

Minimizing Background Modifications

For minimal background modification, we first use a Stable Diffusion model to generate the corresponding background prior image based on the prompt P. Next, this initial image is combined with the real image as introduced in 4.2.3. The combined prior, background region mask, and corresponding prompt P are input into the SDXL inpainting model to obtain an initially generated result. The detailed process is illustrated in Figure 4.10.

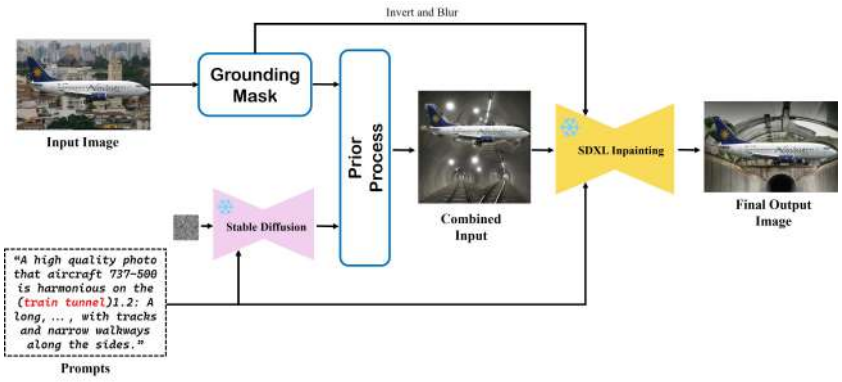

Figure 4.10.: Our minimal background change pipeline.

Minimizing Color Modifications

The color editing process is similar to background modification. First, based on the prompt P, the required RGB image is retrieved from the Color Bank and combined with the real image as introduced in 4.2.3. This is then input into the SDXL inpainting model to generate an initial refined image. Subsequently, the refined image is used as conditional input, along with the Canny edge map of the subject structure, the prompt P into the ControlNet model to obtain the final output. Throughout this process, we ensure that the shape of the main object remains unchanged while the final color is our desired and natural.

For the color and texture setting, due to the output of ControlNet might change the original background, we design a "Final Process" step at the end. The final process process is illustrated in Figure 4.11. The detailed total process is illustrated in Figure 4.12.

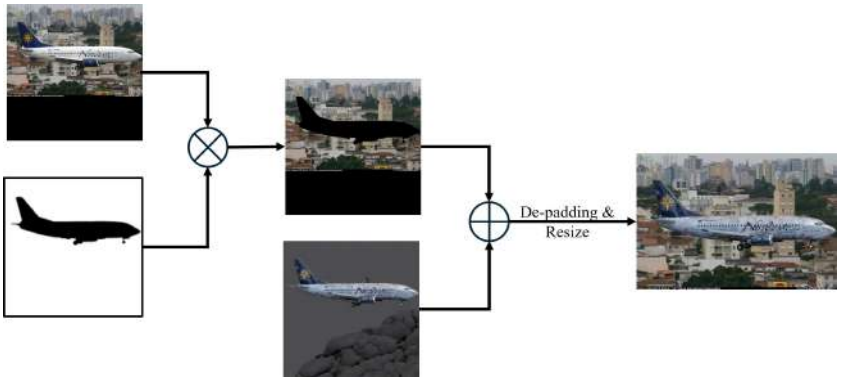

Figure 4.11.: According to the mask image, we crop the original background image based on real input and generated subject without background, then we just add the pixel values from these two images.

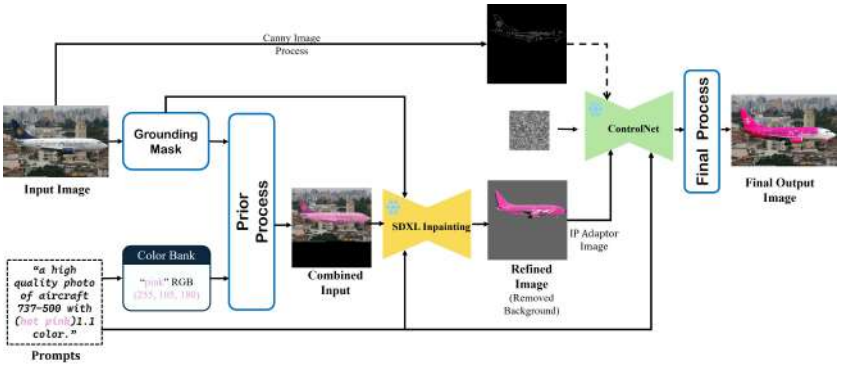

Figure 4.12.: Our minimal color change pipeline.

Minimizing Texture Modifications

The texture modification process is similar to the color modification process, with the difference being that our texture prior is generated by the Stable Diffusion model. First, a Stable Diffusion model generates the texture prior image based on the prompt P, which is then combined with the real image and input into the SDXL inpainting model to produce a refined image. Subsequently, the ControlNet model, along with the Canny edge map, ensures that the image structure remains intact while using the refined image as a reference to generate the final output image. The final output also undergoes a same "final process" step. The detailed process is described in Figure 4.13.

Final Results Analysis

Figure 4.14 shows some final results in D_{SynB}, D_{SynC}, and D_{SynT}. As seen, our method produces better image quality compared with generated images in Section 4.2.2 and Section 4.2.3, ensuring minimal attribute modifications while maintaining the integrity and realism of the

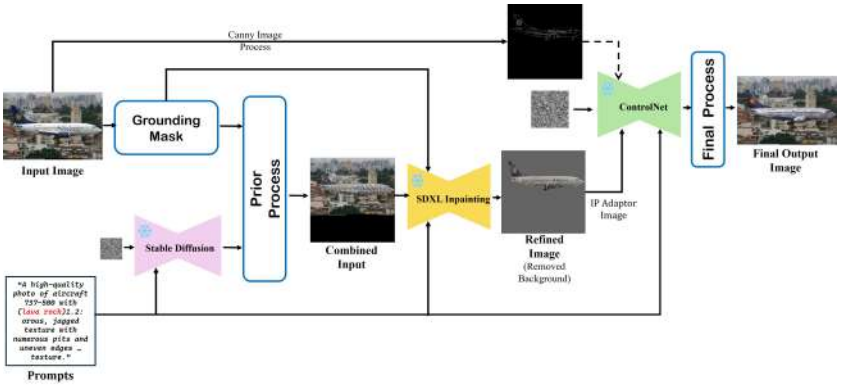

Figure 4.13.: Our minimal texture change pipeline.

main object.

4.3. Validate Effectiveness Using Minimal-change Synthetic Dataset

After establishing our MCSDG pipeline for generating a minimally altered dataset across the three defined attributes, for each dataset, we generated D_{SynB}, D_{SynC}, and D_{SynT} under feasible and infeasible settings. These synthetic datasets will be used to train classifiers to verify the impact of data feasibility by model performance.

4.3.1. Classifier Training

We selected the CLIP [21] model as the classifier. Building on prior research, we fine-tuned both the image encoder (ViT-B/16) and the text encoder of CLIP by incorporating LoRA [103] modules. The CLIP image encoder ViT-B/16 is based on the Vision Transformer (ViT) architecture, which effectively captures spatial relationships in images using self-attention mechanisms. The text encoder maps category labels into the embedding space to match with image features. The ViT-Base model has an embedding size of 768, with 12 transformer layers and a patch size of 16. For the text encoder, the input format is "a photo of [CLS]," where "[CLS]" represents the class name c_i in the real dataset. We employed a supervised learning strategy to train the learnable parameters θ added by the LoRA modules in both the text and image encoders, using a classification loss function while keeping the pretrained weights frozen.

4.3.2. Loss Function

For the case where the classifier is trained using only the synthetic dataset, the loss function is the cross-entropy loss between the synthetic images and their corresponding category labels. In the mixed training scenario, the loss function is an average of the real data and the synthetic data, with the weights controlled by a parameter λ. The loss function is formulated as follows:

Figure 4.14.: Generated images from our MCSDG pipeline, we choose the same samples and prompts as Section 4.2.2 and Section 4.2.3 for show.

$$\mathcal{L}_{\text{mix}} = \lambda \mathcal{L}_{\text{real}} + (1 - \lambda)\mathcal{L}_{\text{synthetic}}$$

where $\mathcal{L}_{\text{real}}$ is the loss for the real dataset, $\mathcal{L}_{\text{synthetic}}$ is the loss for the synthetic dataset, and λ is the parameter that controls the balance between the real and synthetic data during mixed training.

5. Experiments

5.1. Experiments Setup

Datasets

Since our modifications for background, color, and texture require both a well-defined foreground object and a visible background, datasets with images dominated solely by foreground objects, such as ImageNet [105], are unsuitable for our experiments. Furthermore, fine-grained variations within an extensive category provide a better basis for comparing feasible and infeasible attribute changes.

Thus, we selected three base datasets to generate our minimal-change synthetic datasets: Oxford Pets [106] Dataset containing fine-grained pet classes, FGVC Aircraft [107] Dataset containing fine-grained aircraft classes, and Stanford Cars [76] Dataset containing fine-grained car classes.

Implementation Details

In our MCSDG pipeline, we use Stable Diffusion [2] v2.1 to generate prior images for background and texture modifications. We employ the SDXL ControlNet [40] model based on Canny edge images for methods involving ControlNet. The real images for modification are sourced from each dataset's training set. Detailed generation parameters for each dataset and class can be found in Section A.3.

We randomly select seeds for the diffusion model during generation to ensure variation. For each class in each real dataset, we generated synthetic data up to five times the size of the original data. For instance, if a class contained 100 real images, we generated 500 synthetic images to create the synthetic training set. Our main results are based on training with the complete set of real images and five times as many synthetic images.

We use the Adam [108] optimizer to train the CLIP model, with specific training parameters detailed in Section A.4. Notably, since the dataset sizes vary for the three training categories introduced in Section 4,namely only real, only synthetic and mixed training, we ensure a fair comparison by setting the same training iterations and adjusting the number of training iterations to the equal iterations for 70 epochs of the mixed dataset setting. All experiments are conducted using NVIDIA A100 GPUs.

Baseline Methods

Our primary approach focuses on self-comparisons based on our defined feasible and infeasible settings. Specifically, we analyze CLIP classification performance by training on

either only synthetic data or a mixed training. To assess the impact of feasible or infeasible data on classification performance, we use a zero-shot CLIP model and a fine-tuned CLIP model trained on real data under similar conditions. These serve as our baseline comparison models.

Criterion

We use top-1 and top-5 accuracy on the test set for each dataset to assess our model's classification performance. For a given test sample i, let $\hat{y}_i^{(k)}$ denote the k-th highest-ranked predicted label, and let y_i represent the true label. The top-1 accuracy A_1 and top-5 accuracy A_5 are formally defined as follows:

$$A_1 = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\hat{y}_i^{(1)} = y_i) \tag{5.1}$$

$$A_5 = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(y_i \in \{\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots, \hat{y}_i^{(5)}\}) \tag{5.2}$$

where N is the total number of samples in the test set, and $\mathbb{1}(\cdot)$ is an indicator function that equals one if the condition inside is true and zero otherwise.

We assess the model learned knowledge by calculating the Jaccard index for the sets of correctly predicted samples across the test set, defined as follows:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.3}$$

where A and B represent the correctly classified samples in two different training configurations. This metric is to evaluate correct samples overlap between two correct prediction sets.

For dataset distribution analysis, we employ a a common metric for evaluating the distance between generated and real data distributions: Fréchet Inception Distance score. The FID score is defined as:

$$\text{FID}(X, Y) = \|\mu_X - \mu_Y\|^2 + \text{Tr}(\Sigma_X + \Sigma_Y - 2(\Sigma_X\Sigma_Y)^{\frac{1}{2}}) \tag{5.4}$$

where μ_X and Σ_X are the mean and covariance of the features for the real data X, and μ_Y and Σ_Y are those for the generated data Y.

Additionally, we calculate the CLIP Score, Dino Score and LPIPS Score to measure the data similarity: 1) **CLIP Score**: We used the ViT-L/14 model [21], as it is the same CLIP model used in stable diffusion training. 2) **Dino Score**: We employed the DINOv2-Base [5] model for feature extraction. 3) **LPIPS Score**: [109]: The Learned Perceptual Image Patch Similarity score is used to capture fine-grained visual differences between images, particularly in texture and color. LPIPS is calculated as:

$$\text{LPIPS}(x, y) = \sum_{l} w_l \|f_l(x) - f_l(y)\|^2 \tag{5.5}$$

where f_l represents features extracted from layer l of a pretrained network, and w_l are learned weights.

5.2. Classification Performance with Minimal-change Data

5.2.1. Main Quantitative Results

	Settings	*Real*	*Synth*	Pets		AirC		Cars		Avg	
				Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Baseline	CLIP(Zero-shot)	-	-	91.000	99.673	23.8	61.319	63.181	92.744	59.327	84.579
	Real-finetune	✓		95.234	99.891	84.544	97.254	93.004	99.540	90.927	98.895
Feasible	Back			95.398	99.946	86.822	97.704	93.675	99.491	91.965	99.047
	Color		✓	94.510	99.891	80.048	96.805	90.693	99.279	88.417	98.658
	Texture			93.818	99.836	81.157	96.475	90.880	99.379	88.618	98.563
	Back + R			95.261	99.946	88.026	97.554	93.787	99.565	92.358	99.022
	Color+ R	✓	✓	95.261	99.918	84.664	97.194	92.694	99.552	90.873	98.888
	Texture + R			95.245	99.923	83.885	97.224	92.184	99.565	90.438	98.904
Infeasible	Back			95.261	99.891	84.094	97.164	93.812	99.528	91.056	98.861
	Color		✓	94.363	99.864	81.606	96.715	91.464	99.379	89.144	98.653
	Texture			93.273	99.782	81.876	96.984	86.816	98.733	87.322	98.100
	Back + R			95.343	99.918	88.441	97.884	93.725	99.540	92.503	99.114
	Color+ R	✓	✓	95.153	99.891	83.974	97.374	92.520	99.515	90.549	98.927
	Texture + R			95.207	99.946	83.735	97.224	92.197	99.553	90.380	98.908

Table 5.1.: The main quantitative results. We refer "Pets" as Oxford Pets Dataset, "AirC" as Fgvc Aircraft Dataset, and "Cars" as Standford Cars Dataset. The N_{Synth} is 5 times double the N_{Real}. The green value is the baseline performance, while the read values are the settings better than the baseline setting.

Table 5.1 compares the performance of models trained using baseline method, only using synthetic data and mixed training. For mixed training, we fix the synthetic data number as five times double the real images, while we use all of the real images from training set. We make the following observations:(a) The zero-shot CLIP model performs well on the `pets` dataset, achieving similar high performance across all proposed settings and almost reaching an upper limit. (b) We observe that the real images have more effective compared with synthetic images, as the training combined with real images will increase the performance. (c) Compared to feasible and infeasible backgrounds, except for a slight decrease in the infeasible background `aircraft` setting, the overall trend remains consistent: even when using only synthetic data, the fine-tuned CLIP model can surpass baseline results for all datasets. And the difference between feasible and infeasible setting is very small. For only using feasible background synthetic data for training, the Top-1 accuracy can reach around 2.3% Top-1 Accuracy higher than the baseline. (d) However, examining color and texture settings reveals a different pattern. Model performance does not reach baseline levels even for mixed training, nonetheless the only synthetic training. This suggests that changing the objects' color and

texture do not effectively contribute to the model learning invariant features. For mixed training, it even appears to hinder the classifier learning. (e) Feasible and infeasible data yield consistent results across different settings.

We analyze this due to the classifier not only learn the objects' shape, but also learn the objects' color and texture. The background change could be treated as an augmentation to force the model learn the object's own invariant knowledge. Although we ensure minimal change for the subject, the feasible and infeasible color and texture still introduce misleading information compared with test set that degrades classification performance.

We conducted additional experiments to investigate further: we disregarded feasible and infeasible distinctions for each of the three modification settings. We drew synthetic samples from both sets balanced, totaling five times the real data volume for training.

Takeaway:

Modifying color and texture poses significant challenges for model learning and, in some cases, leads to decreased performance, whereas background adjustments can offer positive effects.

	Settings	*Real*	*Synth*	Pets		AirC		Cars		Avg	
				Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Baseline	CLIP(Zero-shot)	-	-	91.000	99.673	23.8	61.319	63.181	92.744	59.327	84.579
	Real-finetune	✓		95.234	99.891	84.544	97.254	93.004	99.540	90.927	98.895
Ours	Back			*95.179*	99.891	86.582	97.524	93.837	99.890	91.866	99.102
	Color		✓	94.144	99.891	81.846	96.655	91.541	99.354	89.177	98.633
	Texture			92.783	99.837	82.036	96.954	91.836	99.428	88.885	98.740
	Back + R			95.288	99.918	87.991	97.644	93.601	99.540	92.293	99.034
	Color+ R	✓	✓	*95.016*	99.864	83.315	97.314	92.768	99.590	90.366	98.923
	Texture + R			95.152	99.946	83.825	97.044	92.557	99.790	90.511	98.927

Table 5.2.: The experiment results for CLIP fine-tuning using mixed feasible and infeasible data. The N_{Synth} is 5 times double the N_{Real}. The green value is the baseline performance, while the read values are the settings better than the baseline setting.

In Table 5.2, we find similar overall classification results as in Table 5.2. By comparing with separated experimental results, we conclude that classification performance remains nearly unchanged after combining data. Combining feasible and infeasible data even marginally improves final classification metrics compared to separate training for synthetic-only training on the `aircraft` and `cars` datasets. However, this improvement diminishes when real data is integrated. Thus, we infer that the feasibility of object attributes does not influence classification outcomes. Instead, the modified attribute (e.g., background, color, texture) significantly impacts classification results.

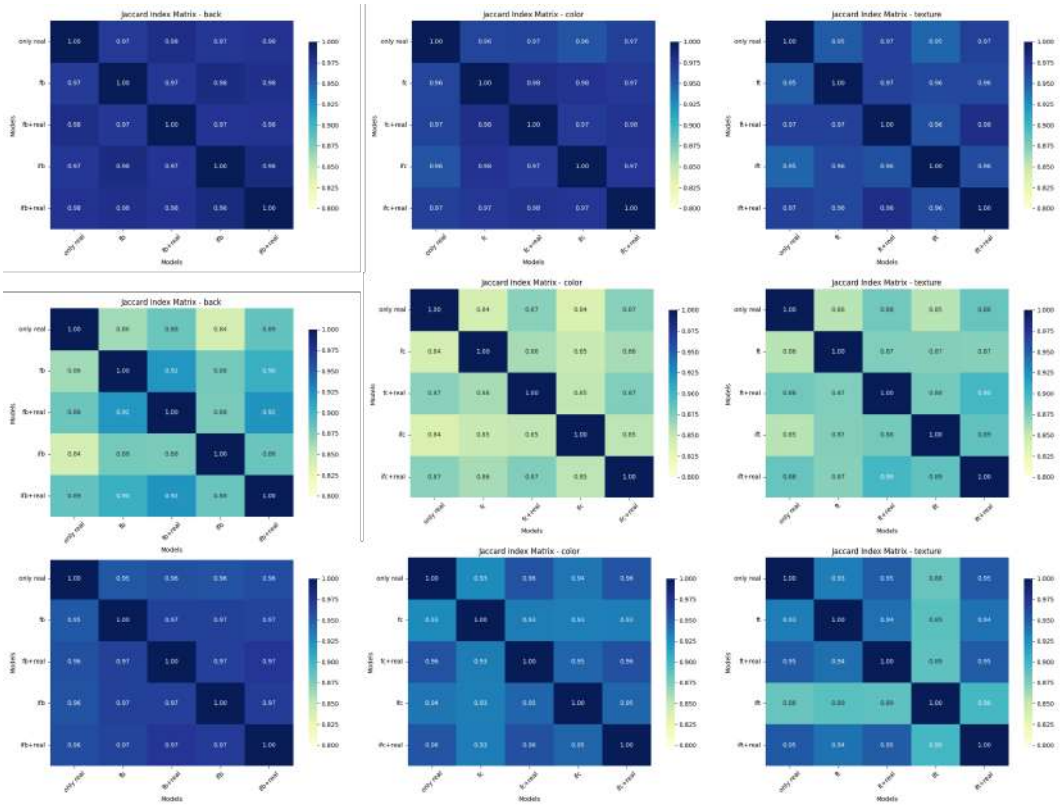

Figure 5.1.: The Jaccard index matrix for three edition settings across three dataset. The first row is the Oxford Pets [106] dataset, the second row is FGVC Aircraft [107] dataset, and the last row is the Stanford Cars [76] dataset.

5.2.2. Classification Results Analysis

We evaluate the prediction correctness for each test sample to analyze whether each model learns similar knowledge under different training settings. As shown in Figure 5.1, we see a consistent trend across the three datasets. When trained using synthetic data, it will change the learned knowledge. As the background setting example, while synthetic data can improve classification accuracy, we observe that the Jaccard index is almost lowest compared only real data and only real data settings(The first row). If mixed training is used, the model's predictions become more aligned with the baseline. But the Jaccard index still kind of different with the only real data, meaning the model is effected by the synthetic data.

However, the knowledge learned under feasible and infeasible overlaps not high, indicating that while performance metrics might be close, the underlying learned representations are not identical.

Takeaway:

The feasible and infeasible data lead the model to learn in different directions, while they achieve very similar performance. The feasibility has no significant impact on classification learning outcomes.

5.3. Analysis of OOD Data

5.3.1. Qualitative Results

Beyond the data samples in our methodology section, we have sampled two additional examples from each of the three datasets. Due to space limitations, we present the samples from Standford Cars [76] Dataset here; more additional samples can be found in the Figure B.5 and Figure B.6 in Section B.2. As shown in the Figures 5.2, the generated data meets our minimal-change requirements. Specifically, feasible modifications align with natural visual perceptions, allowing these samples to be considered as ID data. Conversely, we see clear adaptations to the required transformations for infeasible modifications, especially in cases where the target and original images differ significantly. These infeasible changes are highly unlikely to be present in the test set, thus qualifying them as OOD data.

5.3.2. Distribution Analysis

We calculated the FID score [110] between our generated data and the corresponding real data, using per class's FID score as an indicator of distribution similarity. Additionally, for each synthetic-real data pair, we calculated CLIP Score, Dino Score, and LPIPS score as measures of similarity.

Figure 5.3 illustrates that our generated feasible data is closer to real images compared to infeasible data, aligning with our definitions of in-distribution (ID) and out-of-distribution (OOD) data. For the feasible settings, the generated samples can be considered as ID data, as they have similar counterparts in the real training set. However, as discussed in Section 5.2.1, color and texture modifications do not enhance model performance. One possible reason for the lack of improvement is that color and texture modifications alter the training set distribution. For example, in our aircraft dataset, white is the most common color, with some instances of "red" or "yellow" aircraft. However, our generation method disrupts this balance by generating five synthetic images for each real image. As a result, for each real sample, we generate "red feasible" image (if that color exists within the class), which manually shifts the color proportions and consequently the overall training distribution away from the original balance. Although our intent was to emphasize class-specific characteristics, our experiments show that this approach does not assist the model in identifying useful invariant features. Instead, it appears to introduce challenges in the learning process.

Another possible reason is related to our assumption that the training and test set distributions are aligned. During dataset collection, some samples, like "red airplane," may

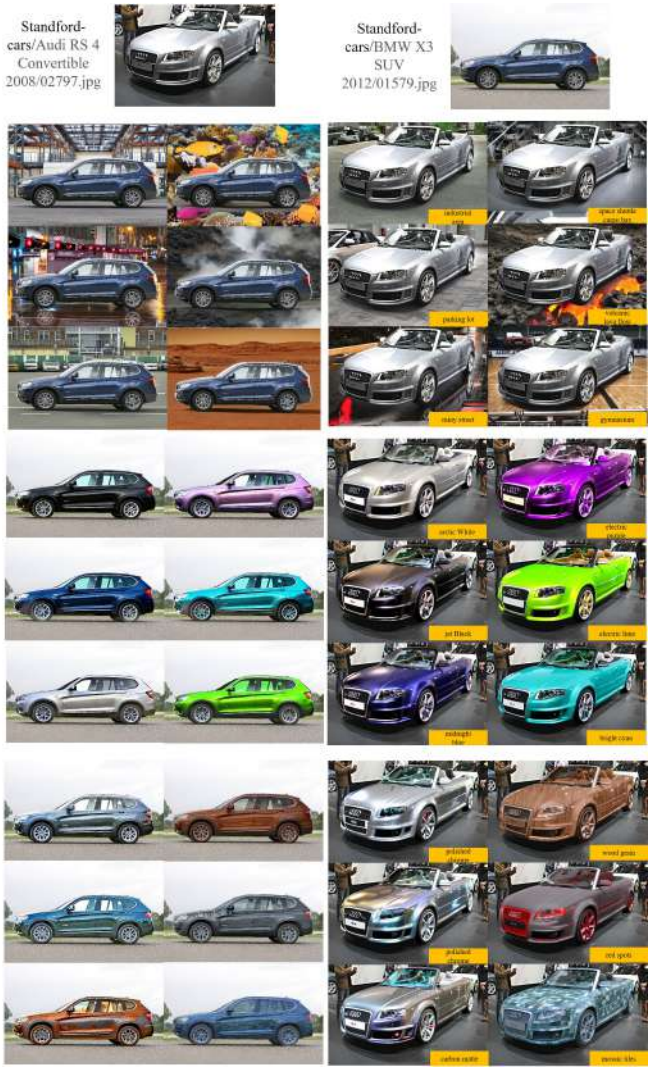

Figure 5.2.: The selected generation visualizations for the Stanford Cars Dataset [76] are shown below. Two real samples are listed at the top, with the corresponding prompt word annotated in the bottom-right corner to indicate the prompts used. For space considerations, we do not display the full prompts, omitting detailed descriptions for background and texture.

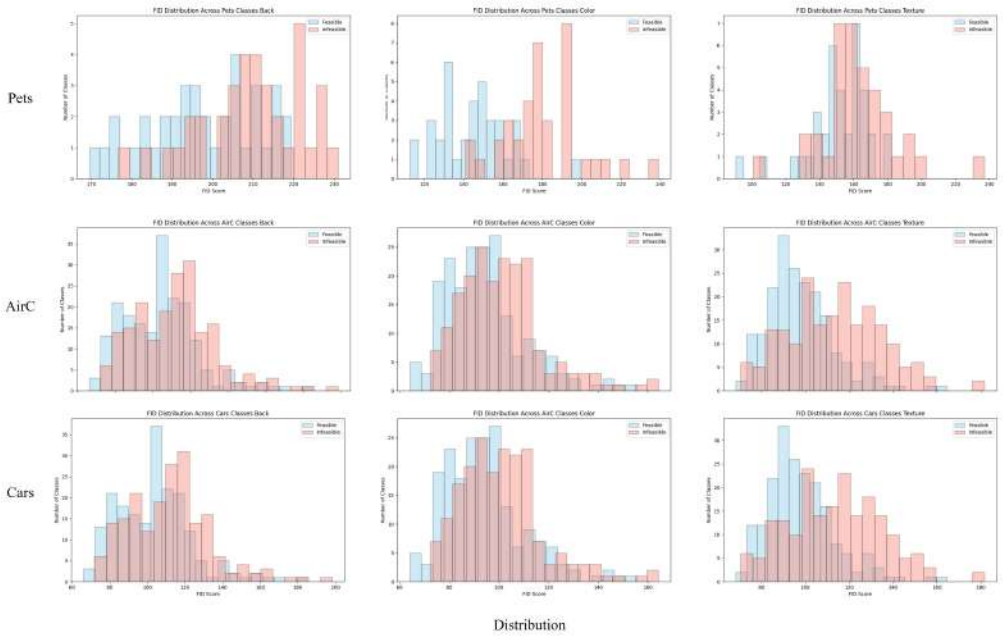

Figure 5.3.: The FID score settings compared using feasible and infeasible settings acorss different dataset.

appear only in the training set and not in the test set, so augmenting such "red" samples in the training set may not only cause domain shift with the training set, but also inadvertently enlarge the domain difference with respect to the test set. The FID distribution further supports this conclusion: while the feasible data distribution is closer to real images than the infeasible data, both distributions are still somewhat distinct from the real images, with feasible data only marginally closer.

A similar pattern is observed in the numeric table analysis, where feasible data is generally more similar to real data than infeasible data. However, metrics like CLIP Score and Dino Score, which do not capture fine-grained details such as texture or color, show close values across different settings. LPIPS, by contrast, reflects some of these nuanced differences. Nonetheless, since feasible data is also generated by modifying a real image, neither feasible nor infeasible data can be similar as real images exactly. Moreover, because both feasible and infeasible settings modify the same area of the real images, their similarity scores appear close.

Takeaway:

The feasible data generated is closer to real images but does not necessarily improve model performance, as altering color and texture disrupts the original training distribution, adding complexity to the learning process.

Setting		Pets			AirC			Cars			Avg		
		CLIP Score↑	DINO Score↑	LPIPS↓	CLIP Score↑	DINO Score↑	LPIPS↓	CLIP Score↑	DINO Score↑	LPIPS↓	CLIP Score↑	DINO Score↑	LPIPS↓
Feasible	B	0.875	0.794	0.543	0.882	0.927	0.447	0.941	0.907	0.352	**0.899**	**0.876**	**0.447**
	C	0.939	0.888	0.344	0.951	0.993	0.089	0.953	0.986	0.149	**0.948**	**0.956**	**0.194**
	T	0.920	0.874	0.379	0.991	0.949	0.089	0.938	0.981	0.152	**0.950**	**0.935**	**0.207**
Ineasible	B	0.875	0.774	0.543	0.882	0.824	0.563	0.920	0.891	0.361	**0.892**	**0.830**	**0.489**
	C	0.872	0.872	0.418	0.914	0.991	0.150	0.926	0.985	0.194	**0.904**	**0.949**	**0.254**
	T	0.882	0.835	0.399	0.941	0.991	0.092	0.912	0.975	0.162	**0.912**	**0.934**	**0.218**

Table 5.3.: The DINO score, CLIP score and LPIPS scrores calculated for each settings in each dataset.

5.3.3. Scaling the Number of Training Images

All previous experiments were conducted using a ratio of five synthetic images per real image. However, similar to prior studies [54], the ratio of synthetic to real data can impact the results. We conducted a scaling experiment on the `aircraft` dataset (and also tested on the `pets` dataset, where results were too similar to show significant variation). For the `aircraft` dataset, we used the entire set of real images and increased the synthetic data from a 1:1 ratio up to a 5:1 ratio.

The results show a nonlinear trend consistent with increasing the scale factor, though the turning points differ across settings. Specifically, for the color and texture settings, peak performance slightly exceeds the baseline, indicating that when the synthetic data distribution diverges significantly from the real data distribution, the benefit of OOD data to real data may be limited to a small range.

Takeaway:

Although modifications to color and texture increase the shift between training data and real-world data, combining real and synthetic data at certain ratios can still enhance classifier performance.

5.4. Ablation Study

In our ablation study, we investigate two specific aspects of our method: (1) the impact of expanding the object mask in the background edition setting, and (2) the effect of removing the SDXL inpainting refinement step for color and texture generation, where we directly use ControlNet with the prior image as input without further refinement.

In Figure 5.5, we can clearly observe that the expanded mask retains the spatial relationship between the object and its background. When the prior image loses the direct relationship between the main subject and its background, the final generated image often exhibits a noticeable "floating" effect, as if the object is not naturally integrated within a specific environment.

As discussed in Section 4, combining the real image and prior image improves the ControlNet generation results compared to using only the raw prior image. In Figure 5.6, we observe that even with "final processing" applied to ControlNet's output, the level of realism and natural

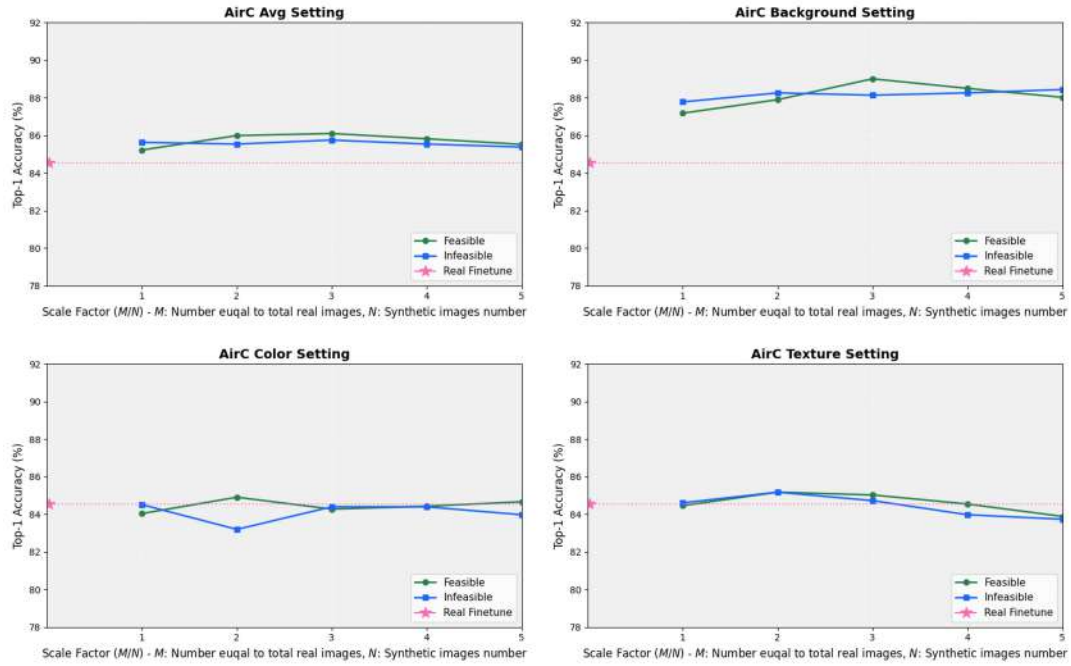

Figure 5.4.: The scaling experiment results for the FGVC-Aircraft [107] dataset are shown for background, color, and texture settings. The horizontal axis represents the scale factor for synthetic images relative to real images. Here, total real image training set is used, with scale factors ranging from 1 to 5.

Figure 5.5.: The ablation study for the usage to expand object mask for background edition setting. We show the real generated prior background on the left, and then present the different combined image with real and prior image.

integration is still inferior to results where SDXL inpainting refinement is first applied before further processing. This refinement step thus enhances both the realism and the seamless appearance of generated images.

Figure 5.6.: The ablation study for the generated images for SDXL Controlnet with final process(left), and our final generation images using both SDXL inpainting for refinement and SDXL Controlnet for final generation(right).

6. Future Work

Although we have made efforts to ensure that the generated images meet the requirements as closely as possible, and most results align with expectations as shown in the Section 5, certain exceptions may still arise during the generation process. Our method includes several tunable hyperparameters, such as: 1) The dilation ratio of the mask during background modification; 2) The transparency when combining prior images with real images during color and texture modifications; 3) The modification intensity for the mask regions in the SDXL inpainting [74] model; 4) The strength of the IP-adaptor [46] in ControlNet [40].

Figure 6.1.: Failure case examples.

These hyperparameters act as potential variables that can influence the output results. Additionally, since the feasible and infeasible prompts P are generated by ChatGPT [3] and manually reviewed, occasional errors in specific scenarios are inevitable. For example in Figure 6.1, certain abstract or similar textures may result in generated images that fail to accurately retain the texture characteristics. Additionally, while certain backgrounds are feasible for specific categories of objects, they may create infeasible combinations with the object itself. Like a plane in a hangar is realistic; however, if the real image depicts a flying plane, altering the background to place it in a hangar mid-flight is clearly not a feasible scenario in Figure 6.1. Fortunately, these cases constitute only a small proportion of the data, but developing a filtering algorithm for such scenarios would be essential in future work if time permits.

For these sub-optimal generation results, we can employ a filtering method based on Visual Question Answering (VQA) to screen the outputs. Some current open-source large-scale vision-language models, such as Llava-Next [89] or InternVL [4], have shown promising results in filtering generated images. By inputting the generated images along with predefined questions, these models can filter out images that do not meet the expected requirements.

The process is shown in Figure 6.2.

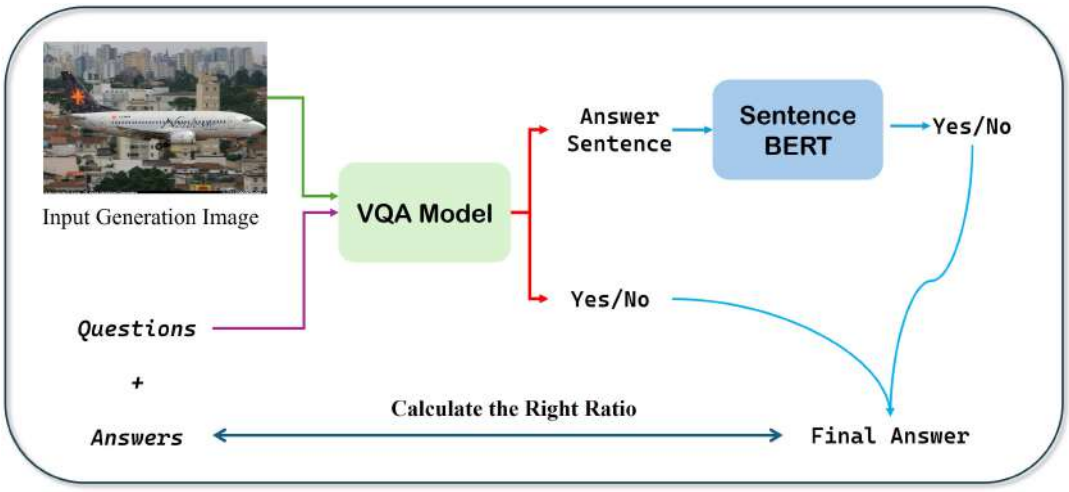

Figure 6.2.: The VQA process using a VLM model to filter the generated images using predefined qustions to check certain aspect for the generated image and ground truth answers.

We have implemented this filtering method; however, due to the time-consuming nature of the process, we have not yet performed a thorough screening of all generated images. Although VQA filtering can significantly enhance the quality of the generated results, ensuring perfect generation still requires manual screening.

Furthermore, although our method achieves minimal modifications, the overall model complexity remains relatively high, requiring significant inference resources and time. Developing an approach that effectively modifies these three attributes without the need for extensive training would be valuable. This would allow for extending synthetic data generation to additional datasets and scaling beyond a scale factor of 5, potentially up to a scale factor of 30, to yield more substantial and insightful experimental results. Also we have implemented some other methods to change the object relative size, but due to the pipeline is not very stable, we did not conduct detailed generation and experimentation. More attributes edition might be helpful for the project.

7. Conclusion

In this work, we explored whether the feasibility of the data is necessary for downstream tasks using our Minimal-Change Synthetic Data Generation (MCSDG) pipeline. From the controlled background, color, and texture of alterations in real-world images, we create corresponding feasible and infeasible scenarios for three fine-grained datasets. We validate our question using the generated synthetic dataset with real images by fine-tuning a CLIP model with LoRA. Our findings demonstrate that modifying background attributes can effectively enhance a classifier's ability to focus on the invariant features of the main object. However, when modifications directly alter the main object in ways that deviate from the test set, classifier performance can be adversely affected. The classifier learns not only the shape of the object itself but also the color or texture of the object. However, even modifying the feasible color or texture will inevitably produce some differences from the real test data. It might due to the resolution of the modified image is reduced or there is some deviation between the training set and the test set, resulting in a decrease in the classification ability of the classifier.

Moreover, our experiments reveal that combining synthetic data with a suitable proportion of real-world data can improve classifier performance. We observed that feasible and infeasible data settings lead the classifier to learn differently, although the overall learning effectiveness remains consistent.

In summary, our work provides a comprehensive approach for modifying attributes in real-world images through the MCSDG pipeline. Our extensive experiments validate our findings and offer detailed insights into the behavior of feasible or infeasible synthetic data in machine learning. These results shed light on how to generate synthetic data generation, emphasizing its role in effectively augmenting real-world datasets for robust downstream tasks.

A. Appendix

A.1. Generation Prompts for G_{LLM}

Due to we have 9 settings for three dataset we used and the most prompts are the same, only few details are changed for a specific setting. Here we use background setting for Oxford Pets [106] dataset as example.

Example for Pets Background Word Generation

```
Task: As an AI language model, generate backgrounds where the
given class of objects typically exists ('feasible') and where they
absolutely cannot exist ('unfeasible'). For each background, provide
a one-sentence description detailing its visual appearance. Imagine
the background and describe it accordingly, adhering to the specified
criteria.
Criteria:

  1. Feasible Backgrounds: Identify environments where the object
     class naturally occurs.

  2. Unfeasible Backgrounds: Identify environments where the object
     class cannot naturally logically be present. Avoid including
     fantastical or impossible scenarios, such as "inside a sun".

  3. Unique Backgrounds: Ensure each listed background is distinct
     and not synonymous with others provided.

  4. Empty List Handling: If no unfeasible backgrounds can be
     identified, use 'EMPTY' to denote this.

  5. Format Requirement: Answers must be formatted as a Python list,
     following the structure shown in the 'Answer' section of the
     'Example'.

Positive Example:

     Object Class: Dog
     Question: Provide five different unfeasible backgrounds for
     a dog, each accompanied by a concise visual description.
     Answer:
```

```
    [
        'underwater coral reef: A vibrant underwater scene
           filled with colorful corals, schools of fish, and
           shimmering light filtering through the water surface
           .',
        'volcano crater: A rugged, rocky landscape with molten
           lava, steam vents, and an eerie red glow from the
           molten rock below.',
        'desert dunes: A vast, arid landscape with rolling sand
           dunes, scorching heat, and sparse vegetation under a
            blazing sun.',
        'unfeasible 4',
        'unfeasible 5'
    ]

Negative Examples:

    The Answers are not acceptable as follows:

    [
        'industrial furnace room: A high-temperature environment
            with large furnaces used for metal smelting, filled
            with intense heat and noise.',
        'operating theater: A sterile room in a hospital where
           surgeries are performed, requiring a clean and
           controlled environment.'
    ]

    Because:

      • Responses are not in a single 'Python List' format: ['',
        '', ..., ''].
      • No need to describe reasons why this background is not
        suitable, like 'unsuitable for pets'. Descriptions
        should focus on specific visual elements (like objects,
        colors, lighting) within the scene rather than abstract
        concepts.
      • Example like 'Industrial furnace room' should describe
        'a large furnace with workers and red-hot objects'.

Question: Please give me 20 different feasible and unfeasible
background respectively for the class 'pets', in the meantime, also
give me the detailed description for the background.
```

A.2. Feasible and Infeasible Prompts Example

In this section, we select some prompt words W to T2I SD model for show. Due to the space, we give part of the prompts from Oxford Pets [106] dataset. Here we present 5 prompt words for each classes. The number like "1.2" here is due to we use the prompt weighting techniques, the value here is just the prompt weights.

Feasible Prompt Word Examples from Pets.
Background: Feasible Background:

- **(suburban backyard)1.2:** A grassy area with a wooden fence, a few trees, and a doghouse in one corner.
- **(city park)1.2:** A green space with open fields, walking paths, and other people walking their dogs.
- **(beach)1.2:** A sandy shore with gentle waves, seashells scattered about, and a few beachgoers in the distance.
- **(forest trail)1.2:** A dirt path winding through tall trees, with patches of sunlight filtering through the leaves.

  ...
- **(dog park)1.2:** A fenced area with agility equipment, water bowls, and several dogs playing together.
- **(rural countryside)1.2:** Rolling hills with grazing cows, wooden fences, and a distant farmhouse.
- **(patio)1.2:** A stone patio with outdoor furniture, potted plants, and a view of the garden.
- **(orchard)1.2:** A fruit orchard with rows of trees bearing apples, oranges, or other fruits.
- **(cafe)1.2:** A cafe with outdoor seating, water bowls for dogs, and a few other patrons with their pets.

Feasible Color:

- **Abyssinian:** (ruddy)1.3, (blue gray)1.3, (silver)1.3, (fawn)1.3, (fawn)1.3.
- **American Bulldog:** (white)1.3, (brindle)1.3, (brown)1.3, (fawn)1.3, (brown)1.3.
- **American Pit Bull Terrier:** (blue gray)1.3, (fawn)1.3, (black)1.3, (white)1.3, (brown)1.3.

- **Basset Hound:** (brown)1.3, (white)1.3, (black)1.3, (tan)1.3, (black)1.3.
- **Beagle:** (black)1.3, (brown)1.3, (white)1.3, (tan)1.3, (brown)1.3.
- ...
- **Siamese:** (seal)1.3, (blue gray)1.3, (chocolate)1.3, (lilac)1.3, (cream)1.3.
- **Sphynx:** (white)1.3, (black)1.3, (cream)1.3, (cream)1.3, (chocolate)1.3.
- **Staffordshire Bull Terrier:** (blue gray)1.3, (black)1.3, (black)1.3, (brindle)1.3, (white)1.3.
- **Wheaten Terrier:** (wheaten)1.3, (golden)1.3, (wheaten)1.3, (wheaten)1.3, (golden)1.3.
- **Yorkshire Terrier:** (blue gray)1.3, (tan)1.3, (black)1.3, (gold)1.3, (tan)1.3.

Feasible Texture:

- **Abyssinian:**
  - (ruddy ticked coat)1.3: warm ruddy brown fur with black ticking throughout.
  - (sorrel coat)1.3: light reddish-brown fur with coppery tones.
  - (blue coat)1.3: soft blue-gray fur with warm undertones.
  - (fawn coat)1.3: light cream-colored fur with a gentle rose tint.
  - (chocolate ticked coat)1.3: rich chocolate fur with lighter ticking.
- **American Bulldog:**
  - (white and brindle coat)1.3: short fur with a mix of white and brindle patches.
  - (solid white coat)1.3: smooth, short white fur.
  - (fawn and white coat)1.3: short fur with fawn patches on a white base.
  - (brindle coat)1.3: short fur with a mix of dark stripes on a lighter background.
  - (red and white coat)1.3: short fur with red patches mixed with white.
- **American Pit Bull Terrier:**
  - (blue coat)1.3: short, sleek blue-grey fur.
  - (red nose coat)1.3: smooth red-brown fur with a coppery hue.
  - (black and white coat)1.3: short fur with black and white patches.
  - (brindle coat)1.3: short fur with dark stripes over a lighter background.

  - (solid black coat)1.3: glossy black short fur.

- **Basset Hound:**
  - (tri-color coat)1.3: short fur in black, white, and tan patches.
  - (lemon and white coat)1.3: smooth, short fur in light yellow and white.
  - (mahogany coat)1.3: rich red-brown short fur.
  - (black and white coat)1.3: short fur with bold black and white patches.
  - (red and white coat)1.3: short fur with red and white patches.
- ...
- **Staffordshire Bull Terrier:**
  - (solid blue coat)1.3: short, sleek blue-grey fur throughout.
  - (red coat)1.3: short fur of a rich red color.
  - (black brindle coat)1.3: black fur with brindle (tiger-stripe) pattern.
  - (black and white coat)1.3: short fur with black and white patches.
  - (solid black coat)1.3: short, glossy black fur.
- **Wheaten Terrier:**
  - (soft wheaten coat)1.3: wavy fur of a warm beige color.
  - (golden wheaten coat)1.3: wavy fur with golden hues.
  - (red wheaten coat)1.3: wheaten fur with reddish tint.
  - (pale wheaten coat)1.3: light cream-colored wavy fur.
  - (wheaten coat with black tipping)1.3: wheaten fur with black tips.
- **Yorkshire Terrier:**
  - (steel blue and tan coat)1.3: long, silky fur in steel blue with tan points.
  - (black and tan coat)1.3: shiny black fur with tan points.
  - (golden tan coat)1.3: long fur in a rich golden tan color.
  - (blue and gold coat)1.3: dark blue fur with golden tan accents.
  - (silver and tan coat)1.3: light silver fur with warm tan points.

Infeasible Prompt Word Examples from Pets.
Infeasible Texture:

- **(space station)1.2:** A high-tech interior with floating objects, control panels, and a view of Earth through a window.

- **(deep sea)1.2:** A dark, underwater environment with bioluminescent creatures and no sunlight.
- **(volcano interior)1.2:** A fiery landscape with flowing lava, molten rocks, and intense heat.
- **(airplane cockpit)1.2:** A confined space with numerous controls, screens, and a view of the sky through the windshield.
- **(submarine interior)1.2:** A cramped, metallic space with control panels, periscopes, and no natural light.
- ...
- **(mars surface)1.2:** A barren, reddish landscape with rocks, dust, and no signs of life.
- **(server farm)1.2:** A large room filled with rows of servers, cooling systems, and blinking lights.
- **(intense storm at sea)1.2:** A chaotic scene with high waves, strong winds, and dark storm clouds.
- **(deep space)1.2:** The vast, empty expanse of space with distant stars, no gravity, and no solid ground.
- **(tornado path)1.2:** A chaotic environment with high winds, flying debris, and damaged structures.
- **(sewer system)1.2:** A dark, damp network of tunnels with flowing water, pipes, and an unpleasant smell.

Infeasible Color:

- **Abyssinian:** (purple)1.3, (blue)1.3, (pink)1.3, (orange)1.3, (neon green)1.3.
- **American Bulldog:** (purple)1.3, (pink)1.3, (blue)1.3, (green)1.3, (yellow)1.3.
- **American Pit Bull Terrier:** (purple)1.3, (green)1.3, (blue)1.3, (orange)1.3, (pink)1.3.
- **Basset Hound:** (purple)1.3, (blue)1.3, (green)1.3, (yellow)1.3, (pink)1.3.
- ...
- **Scottish Terrier:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (orange)1.3.
- **Shiba Inu:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (pink)1.3.

- **Siamese:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (pink)1.3.
- **Sphynx:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (orange)1.3.
- **Staffordshire Bull Terrier:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (pink)1.3.
- **Wheaten Terrier:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (pink)1.3.
- **Yorkshire Terrier:** (green)1.3, (purple)1.3, (blue)1.3, (yellow)1.3, (orange)1.3.

Infeasible Texture:

- **(elephant skin texture)1.3:** characterized by thick, rough, and wrinkled surfaces, with deep creases and fine cracks forming a complex, uneven pattern that provides both strength and flexibility.
- **(cracked marble)1.3:** a smooth, solid surface interspersed with intricate, jagged cracks running across the skin, resembling broken marble.
- **(wood grain)1.3:** parallel grooves and rings resembling tree bark, with a natural flow pattern typically seen in wooden planks.
- **(stone mosaic)1.3:** composed of small, irregularly shaped stone pieces arranged in a decorative, tiled pattern. ...
- **(lava rock)1.3:** porous, jagged surface with numerous holes and sharp edges, resembling solidified lava.
- **(metallic scales)1.3:** small, shiny scales arranged in an overlapping pattern, giving a reflective and armor-like quality.

A.3. Minimal-Change Dataset Genration Config

The Table A.1 below presents the specific generation parameters used in our MCSDG pipeline throughout the generation process.

A.4. CLIP Fine-tuning Detailed Config

The Table A.2 below presents the specific hyper-parameters used in our CLIP fine-tuning process.

Parameters	Back						Color						Texture					
	Pets		AirC		Cars		Pets		AirC		Cars		Pets		AirC		Cars	
	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF	F	IF
Guidance Scale for SDXL	40		7.5		7.5		12		12		30		12		8		30	
Guidance Scale for Contronet	×						7.5						7.5					
Strength for SDXL	0.99		0.95		0.9		0.3		0.8		0.85		0.3	0.3	0.65	0.3	0.65	0.3
IP-Adptor Strength	×						0.7		0.4		0.4		0.2	0.5	0.65	0.4	0.65	0.4
Inference Step for SD	20						×						15					
Inference Step for SDXL	30						20						20					
Inference Step for ControlNet	×						30						30					
Mask dilated factor/alpha factor	120		50		25		0.3		0.6		0.6		0.5	0.4	0.5	0.65	0.65	0.65

Table A.1.: The detailed generation parameters for MCSDG pipeline for each settings and each dataset.

HyperParameters	**lamda**	**lr**	**Min_lr**	**Weight decay**	**Warm up steps**	**CLIP LoRA rank**	**CLIP LoRA alpha**
Values	0.5	{1e-3,1e-4,1e-5}	1e-08	1e-03	5% total iterations	16	32
HyperParameters	**Training bs**	**Test bs**	**Train iterations**	**Val iterations**	**Data augmentation**		
Values	64	8	Pets:20700/AirC:72000/Cars:91840	1/70 Train iterations	random resized crop, random horizontal flip, random color jitter, and random gray scale		

Table A.2.: The hyper-parameter details for fine-tuning CLIP model.

B. Figures

B.1. Approach Part Method Generations Comparison

In this appendix section, we present comparison figures omitted from Section 4 due to space limitations. Figures B.1 and B.2 illustrate the four baseline methods in color and texture adjustments, respectively. Figures B.3 and B.4 provide comparisons of generation results using prior information with SDXL inpainting and ControlNet.

Figure B.1.: The color attribute change setting results for four different base models.

Figure B.2.: The texture attribute change setting results for four different base models.

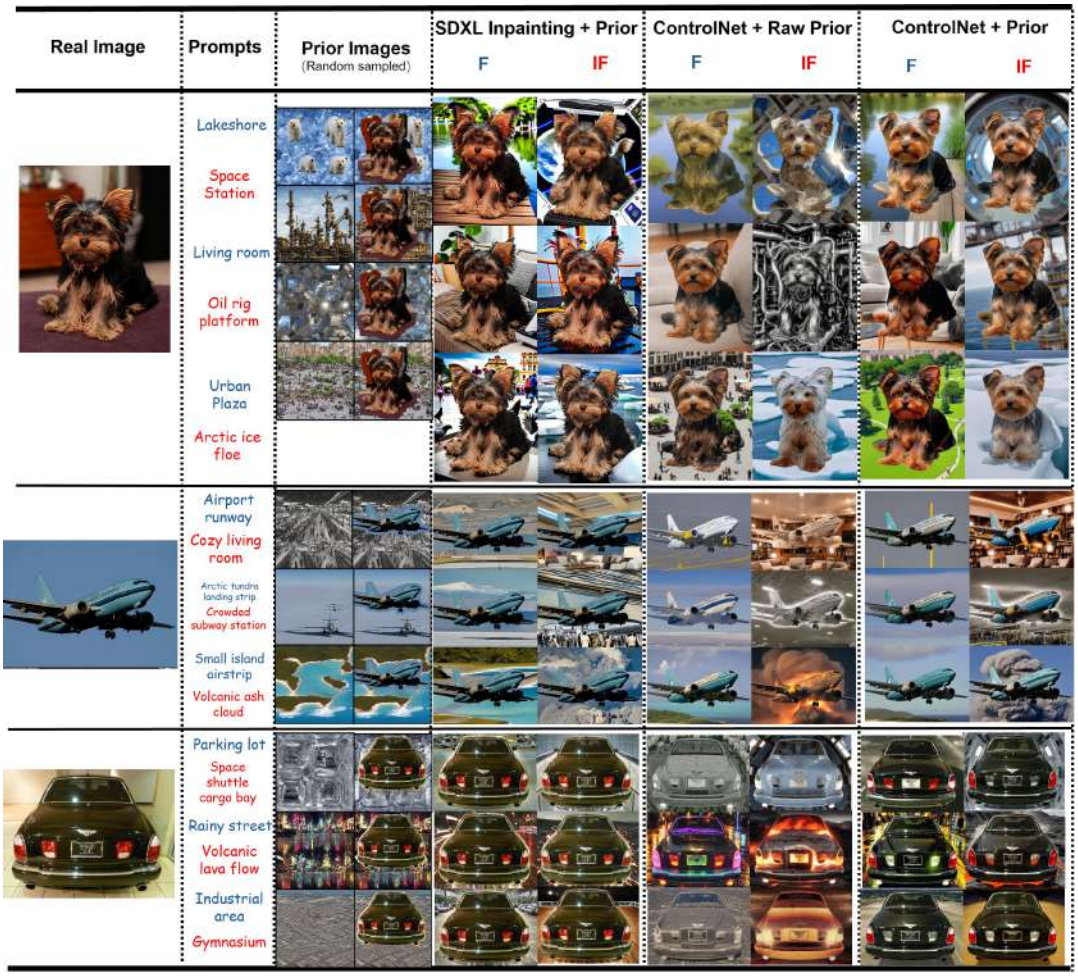

Figure B.3.: The background attribute change setting results with prior input listed on the left.

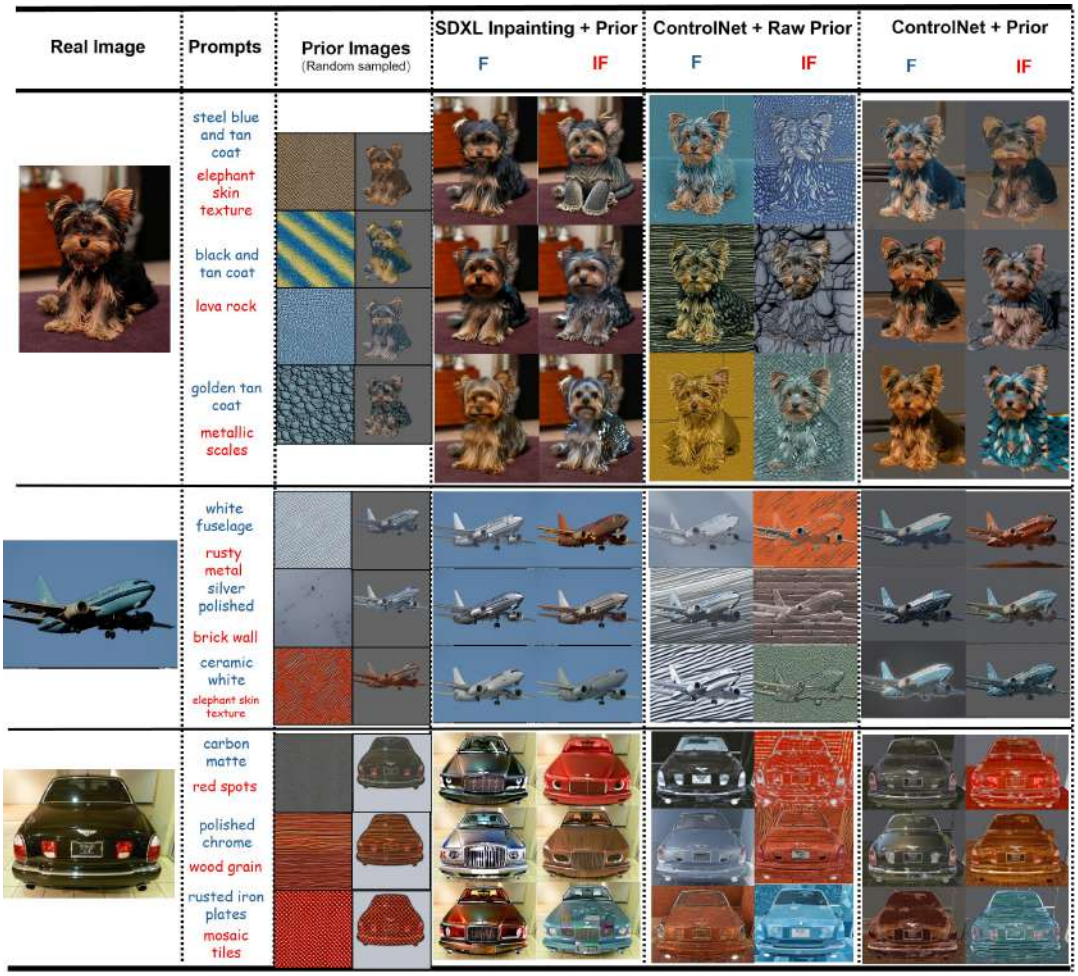

Figure B.4.: The texture attribute change setting results with prior input listed on the left.

B.2. Final Generation Examples Visualization

We present additional generated examples in Figure B.5 and B.6 as an addition describe in Section 5. Also more diverse samples in Figure B.7. As shown, the modifications are successfully applied in each randomly selected example.

Figure B.5.: The selected generation visualizations for the Oxford Pets Dataset [106] are shown below. Two real samples are listed at the top, with the corresponding prompt word annotated in the bottom-right corner to indicate the prompts used. For space considerations, we do not display the full prompts, omitting detailed descriptions for background and texture.

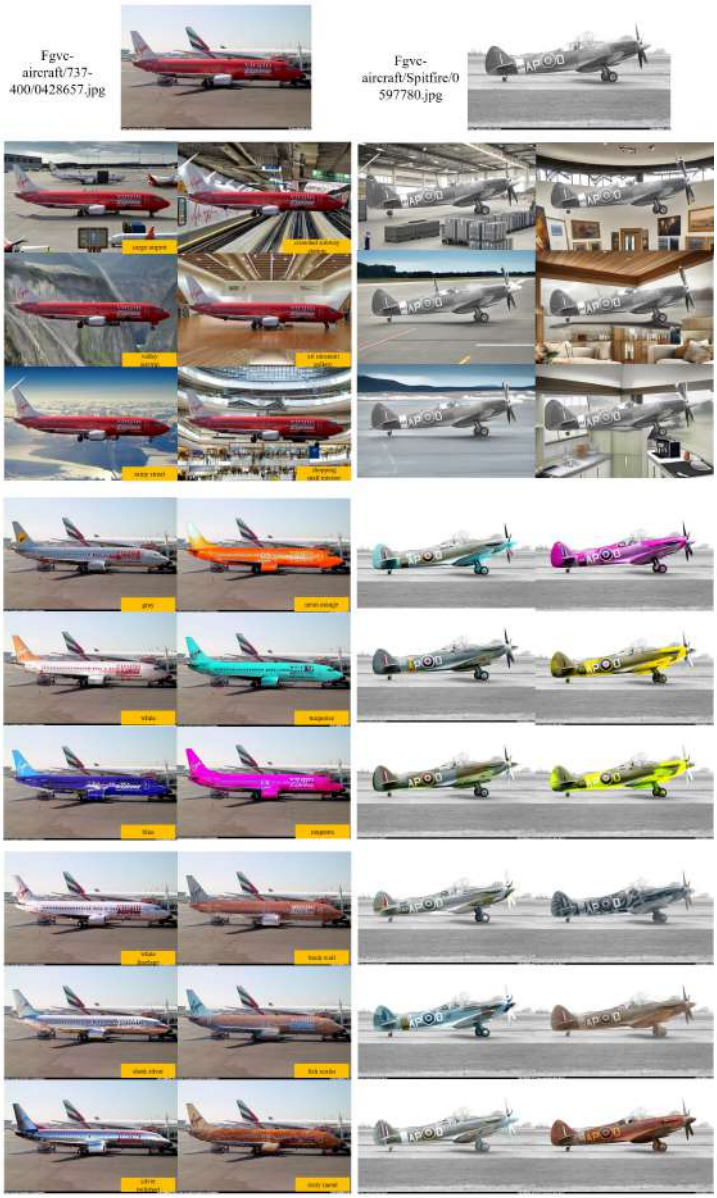

Figure B.6.: The selected generation visualizations for the Fgvc-Aircraft Dataset [107] are shown below. Two real samples are listed at the top, with the corresponding prompt word annotated in the bottom-right corner to indicate the prompts used. For space considerations, we do not display the full prompts, omitting detailed descriptions for background and texture.

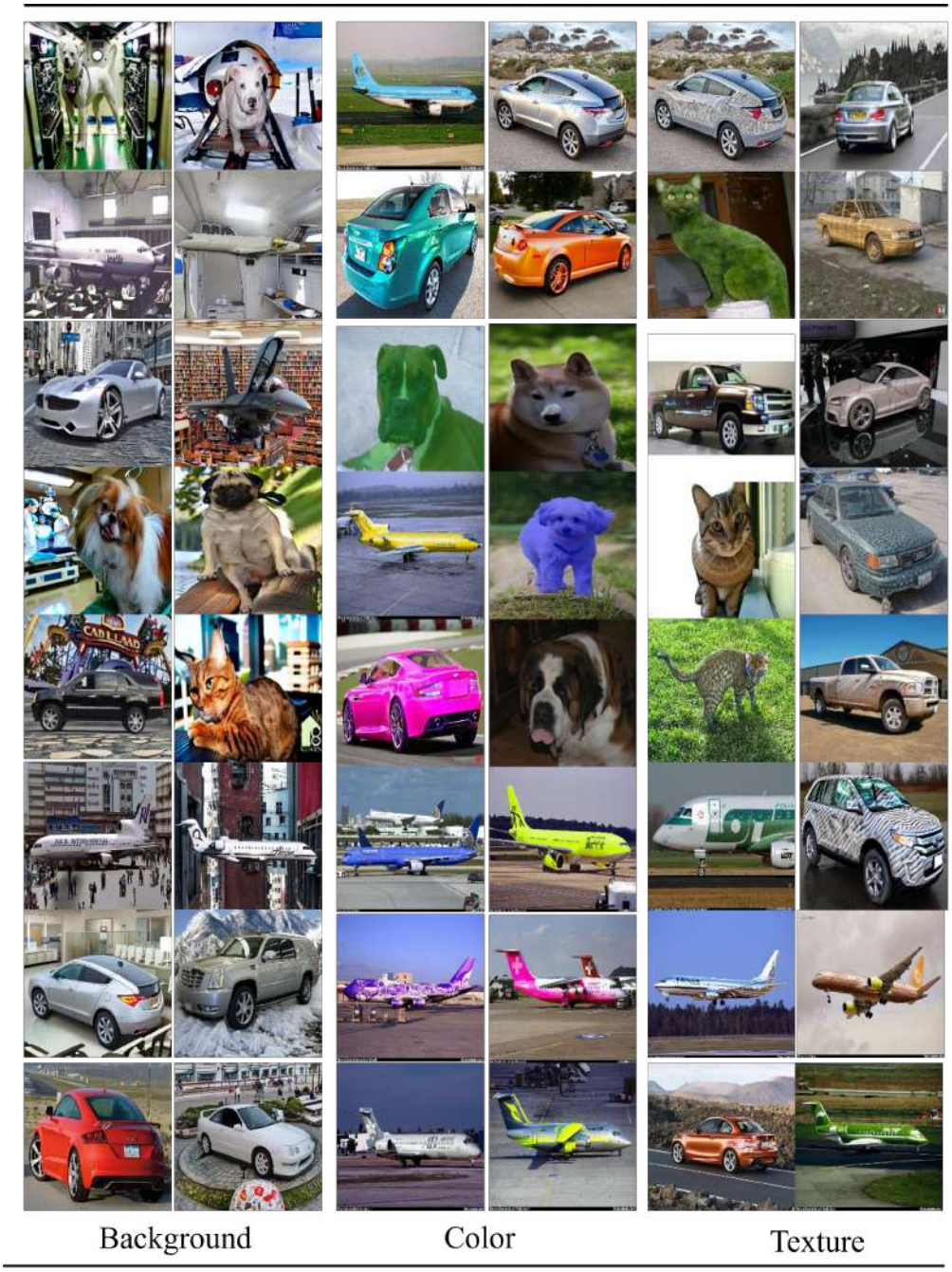

Figure B.7.: More generation examples randomly selected from our D_{SynB}, D_{SynC}, and D_{SynT}. For visualization, we resize all of the images to be the same size.

List of Figures

List of Tables

Glossary

[*Attribute*] The edition attributes, like background, color and texture. 36

c_i The individual class names in a dataset. 34–37, 46

[*CLASS*] The class name in a dataset. 36

$D_{\textsf{SynB}}$ Synthetic Minimal Change Background Dataset. 34, 46, 76, 79

$D_{\textsf{SynC}}$ Synthetic Minimal Change Color Dataset. 34, 46, 76, 79

$D_{\textsf{SynT}}$ Synthetic Minimal Change Texture Dataset. 34, 46, 76, 79

G_{LLM} LLM prompt text generator. 34

G T2I Stable Diffusion image generator. 34, 35, 38

P_{ID} In distribution prompts. 34, 36, 37

P_{OOD} Out-of-distribution prompts. 34, 36, 37

Acronyms

C C. 34–36

ID In Distribution. iv, vi, 19–21, 34, 35, 53

MCSDG Minimal-Change Synthetic Data Generation. iv, 34, 46, 48, 61, 78

OOD Out-of-Distribution. iv–vi, 18–22, 34, 35, 53, 56

SD Stable Diffusion. vi, 23–25, 33

T2I Text-to-Image. vi, 23

Bibliography

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

[2] 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. 10684–10695.

[3] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al. "Summary of chatgpt-related research and perspective towards the future of large language models". In: *Meta-Radiology* (2023), p. 100017.

[4] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 24185–24198.

[5] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum. "Dino: Detr with improved denoising anchor boxes for end-to-end object detection". In: *arXiv preprint arXiv:2203.03605* (2022).

[6] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. "Segment anything". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026.

[7] F. Zhou, Z. Wang, Q. Liu, J. Li, and P. Liu. "Programming Every Example: Lifting Pre-training Data Quality like Experts at Scale". In: *arXiv preprint arXiv:2409.17115* (2024).

[8] D. P. Kingma. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[10] Z. Yu, C. Zhu, S. Culatana, R. Krishnamoorthi, F. Xiao, and Y. J. Lee. "Diversify, Don't Fine-Tune: Scaling Up Visual Recognition Training with Synthetic Images". In: *arXiv preprint arXiv:2312.02253* (2023).

[11] J. M. Kim, J. Bader, S. Alaniz, C. Schmid, and Z. Akata. "DataDream: Few-shot Guided Dataset Generation". In: *arXiv preprint arXiv:2407.10910* (2024).

[12] M. B. Sarıyıldız, K. Alahari, D. Larlus, and Y. Kalantidis. "Fake it till you make it: Learning transferable representations from synthetic imagenet clones". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8011–8021.

[13] R. He, S. Sun, X. Yu, C. Xue, W. Zhang, P. Torr, S. Bai, and X. Qi. "Is synthetic data from generative models ready for image recognition?" In: *arXiv preprint arXiv:2210.07574* (2022).

[14] L. Dunlap, A. Umino, H. Zhang, J. Yang, J. E. Gonzalez, and T. Darrell. "Diversify your vision datasets with automatic diffusion-based augmentation". In: *Advances in neural information processing systems* 36 (2023), pp. 79024–79034.

[15] V. G. T. da Costa, N. Dall'Asen, Y. Wang, N. Sebe, and E. Ricci. "Diversified in-domain synthesis with efficient fine-tuning for few-shot classification". In: *arXiv preprint arXiv:2312.03046* (2023).

[16] H. A. A. K. Hammoud, H. Itani, F. Pizzati, P. Torr, A. Bibi, and B. Ghanem. "SynthCLIP: Are We Ready for a Fully Synthetic CLIP Training?" In: *arXiv preprint arXiv:2402.01832* (2024).

[17] W. Wu, Y. Zhao, M. Z. Shou, H. Zhou, and C. Shen. "Diffumask: Synthesizing images with pixel-level annotations for semantic segmentation using diffusion models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1206–1217.

[18] A. De Silva, R. Ramesh, C. Priebe, P. Chaudhari, and J. T. Vogelstein. "The value of out-of-distribution data". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 7366–7389.

[19] J. Geiping, M. Goldblum, G. Somepalli, R. Shwartz-Ziv, T. Goldstein, and A. G. Wilson. "How much data are augmentations worth? An investigation into scaling laws, invariance, and implicit regularization". In: *arXiv preprint arXiv:2210.06441* (2022).

[20] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger–Lewandowski, T. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, et al. "Deep learners benefit more from out-of-distribution examples". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 164–172.

[21] Y. Li, F. Liang, L. Zhao, Y. Cui, W. Ouyang, J. Shao, F. Yu, and J. Yan. "Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm". In: *arXiv preprint arXiv:2110.05208* (2021).

[22] T. Karras. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[23] A. Vahdat and J. Kautz. "NVAE: A deep hierarchical variational autoencoder". In: *Advances in neural information processing systems* 33 (2020), pp. 19667–19679.

[24] J. D. Curtó, I. C. Zarza, F. De La Torre, I. King, and M. R. Lyu. "High-resolution deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1711.06491* (2017).

[25] J. Song, C. Meng, and S. Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).

[26] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-shot text-to-image generation". In: *International conference on machine learning*. Pmlr. 2021, pp. 8821–8831.

[27] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *arXiv preprint arXiv:2112.10741* (2021).

[28] 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". In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.

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

[30] J. Ho and T. Salimans. "Classifier-free diffusion guidance". In: *arXiv preprint arXiv:2207.12598* (2022).

[31] H. Lin, X. Cheng, and X. Wu. "Cat: Cross attention in vision transformer". In: *IEEE international conference on multimedia and expo (ICME)* (2022).

[32] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. "Prompt-to-prompt image editing with cross attention control". In: *arXiv preprint arXiv:2208.01626* (2022).

[33] *Prompt weighting library*. `https://github.com/damian0815/compel?tab=readme-ov-file`. 2024.

[34] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. "Sdxl: Improving latent diffusion models for high-resolution image synthesis". In: *arXiv preprint arXiv:2307.01952* (2023).

[35] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev. "Reproducible scaling laws for contrastive language-image learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2818–2829.

[36] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or. "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: *arXiv preprint arXiv:2208.01618* (2022).

[37] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 22500–22510.

[38] H. Li, C. Shen, P. Torr, V. Tresp, and J. Gu. "Self-discovering interpretable diffusion latent directions for responsible text-to-image generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 12006–12016.

[39] X. Jia, Y. Zhao, K. C. Chan, Y. Li, H. Zhang, B. Gong, T. Hou, H. Wang, and Y.-C. Su. "Taming encoder for zero fine-tuning image customization with text-to-image diffusion models". In: *arXiv preprint arXiv:2304.02642* (2023).

[40] L. Zhang, A. Rao, and M. Agrawala. "Adding conditional control to text-to-image diffusion models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3836–3847.

[41] S. Zhao, D. Chen, Y.-C. Chen, J. Bao, S. Hao, L. Yuan, and K.-Y. K. Wong. "Unicontrolnet: All-in-one control to text-to-image diffusion models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[42] J. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon. "Ilvr: Conditioning method for denoising diffusion probabilistic models. In 2021 IEEE". In: *CVF international conference on computer vision (ICCV)*. Vol. 1. 2021, p. 2.

[43] I. Najdenkoska, A. Sinha, A. Dubey, D. Mahajan, V. Ramanathan, and F. Radenovic. "Context diffusion: In-context aware image generation". In: *arXiv preprint arXiv:2312.03584* 5 (2023).

[44] J. Huang, M. Yan, Y. Liu, and S. Chen. "Color-SD: Stable Diffusion Model Already has a Color Style Noisy Latent Space". In: *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2024, pp. 1–6.

[45] B. Yang, S. Gu, B. Zhang, T. Zhang, X. Chen, X. Sun, D. Chen, and F. Wen. "Paint by example: Exemplar-based image editing with diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 18381–18391.

[46] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. "Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models". In: *arXiv preprint arXiv:2308.06721* (2023).

[47] K. Bai, H. Zeng, L. Zhang, Y. Liu, H. Xu, Z. Chen, and J. Zhang. "ClearDepth: Enhanced Stereo Perception of Transparent Objects for Robotic Manipulation". In: *arXiv preprint arXiv:2409.08926* (2024).

[48] A. Raistrick, L. Mei, K. Kayan, D. Yan, Y. Zuo, B. Han, H. Wen, M. Parakh, S. Alexandropoulos, L. Lipson, et al. "Infinigen Indoors: Photorealistic Indoor Scenes using Procedural Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21783–21794.

[49] C. Feng, Y. Zhong, Z. Jie, W. Xie, and L. Ma. "Instagen: Enhancing object detection by training on synthetic dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 14121–14130.

[50] 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". In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[51] R. Beaumont. "Clip retrieval: Easily compute clip embeddings and build a clip retrieval system with them". In: *GitHub* (2022).

[52] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. "Describing textures in the wild". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3606–3613.

[53] Y. Le and X. Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7.7 (2015), p. 3.

[54] C. Huyen. *Data Distribution Shifts and Monitoring*. `https://huyenchip.com/2022/02/07/data-distribution-shifts-and-monitoring.html#data-shift-types`. 2022.

[55] J. Yang, K. Zhou, Y. Li, and Z. Liu. "Generalized out-of-distribution detection: A survey". In: *International Journal of Computer Vision* (2024), pp. 1–28.

[56] S. Mariani, S. R. Klomp, R. Romijnders, and P. H. de With. "The Effect of Covariate Shift and Network Training on Out-of-Distribution Detection". In: *Conference Proceedings*. 2023, pp. 723–730.

[57] D. Hendrycks and K. Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136* (2016).

[58] S. Liang, Y. Li, and R. Srikant. "Principled detection of out-of-distribution examples in neural networks". In: *CoRR, abs/1706.02690* 1 (2017).

[59] R. Roady, T. L. Hayes, R. Kemker, A. Gonzales, and C. Kanan. "Are Out-of-Distribution Detection Methods Effective on Large-Scale Datasets? CoRR abs/1910.14034 (2019)". In: *arXiv preprint arXiv:1910.14034* (2019).

[60] W. Liu, X. Wang, J. Owens, and Y. Li. "Energy-based out-of-distribution detection". In: *Advances in neural information processing systems* 33 (2020), pp. 21464–21475.

[61] Y. Zhou. "Rethinking reconstruction autoencoder-based out-of-distribution detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7379–7387.

[62] R. Huang, A. Geng, and Y. Li. "On the importance of gradients for detecting distributional shifts in the wild". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 677–689.

[63] Y. Sun, C. Guo, and Y. Li. "React: Out-of-distribution detection with rectified activations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 144–157.

[64] H. Wang, Z. Li, L. Feng, and W. Zhang. "Vim: Out-of-distribution with virtual-logit matching". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 4921–4930.

[65] S. Fort, J. Ren, and B. Lakshminarayanan. "Exploring the limits of out-of-distribution detection". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7068–7081.

[66] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li. "Open set learning with counterfactual images". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 613–628.

[67] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6023–6032.

[68] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky. "Resolution-robust large mask inpainting with fourier convolutions". In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2022, pp. 2149–2159.

[69] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia. "Mat: Mask-aware transformer for large hole image inpainting". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10758–10768.

[70] B. Xia, Y. Zhang, S. Wang, Y. Wang, X. Wu, Y. Tian, W. Yang, and L. Van Gool. "Diffir: Efficient diffusion model for image restoration". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 13095–13105.

[71] L. Chi, B. Jiang, and Y. Mu. "Fast fourier convolution". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4479–4488.

[72] B. Xia, S. Wang, Y. Tao, Y. Wang, and J. Jia. "Llmga: Multimodal large language model based generation assistant". In: *arXiv preprint arXiv:2311.16500* (2023).

[73] Q. Xiao, G. Li, and Q. Chen. "Image outpainting: Hallucinating beyond the image". In: *Ieee Access* 8 (2020), pp. 173576–173583.

[74] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. R. Van Gool. "Inpainting using denoising diffusion probabilistic models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471.

[75] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. "Repaint: Inpainting using denoising diffusion probabilistic models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11461–11471.

[76] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. "3d object representations for fine-grained categorization". In: *Proceedings of the IEEE international conference on computer vision workshops*. 2013, pp. 554–561.

[77] B. Liu, C. Wang, T. Cao, K. Jia, and J. Huang. "Towards Understanding Cross and Self-Attention in Stable Diffusion for Text-Guided Image Editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 7817–7826.

[78] T. Brooks, A. Holynski, and A. A. Efros. "Instructpix2pix: Learning to follow image editing instructions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 18392–18402.

[79] S. Zhang, X. Yang, Y. Feng, C. Qin, C.-C. Chen, N. Yu, Z. Chen, H. Wang, S. Savarese, S. Ermon, et al. "Hive: Harnessing human feedback for instructional visual editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9026–9036.

[80] T.-J. Fu, W. Hu, X. Du, W. Y. Wang, Y. Yang, and Z. Gan. "Guiding instruction-based image editing via multimodal large language models". In: *arXiv preprint arXiv:2309.17102* (2023).

[81] A. Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[82] J. Li, D. Li, C. Xiong, and S. Hoi. "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation". In: *International conference on machine learning*. PMLR. 2022, pp. 12888–12900.

[83] H. Liu, C. Li, Q. Wu, and Y. J. Lee. "Visual instruction tuning". In: *Advances in neural information processing systems* 36 (2024).

[84] J. D. M.-W. C. Kenton and L. K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of naacL-HLT*. Vol. 1. Minneapolis, Minnesota. 2019, p. 2.

[85] 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". In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[86] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[87] B. Peng, C. Li, P. He, M. Galley, and J. Gao. "Instruction tuning with gpt-4". In: *arXiv preprint arXiv:2304.03277* (2023).

[88] H. Liu, C. Li, Y. Li, and Y. J. Lee. "Improved baselines with visual instruction tuning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 26296–26306.

[89] F. Li, R. Zhang, H. Zhang, Y. Zhang, B. Li, W. Li, Z. Ma, and C. Li. "Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models". In: *arXiv preprint arXiv:2407.07895* (2024).

[90] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[91] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al. "Grounded language-image pre-training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10965–10975.

[92] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499* (2023).

[93] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[94] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. "Yolo-world: Real-time open-vocabulary object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 16901–16911.

[95] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. "Sam 2: Segment anything in images and videos". In: *arXiv preprint arXiv:2408.00714* (2024).

[96] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. "Grounded sam: Assembling open-world models for diverse visual tasks". In: *arXiv preprint arXiv:2401.14159* (2024).

[97] briaai. *BRIA Background Removal v1.4 Model*. `https://huggingface.co/briaai/RMBG-1.4`. 2024.

[98] Z. Yang, M. Ding, Y. Guo, Q. Lv, and J. Tang. "Parameter-efficient tuning makes a good classification head". In: *arXiv preprint arXiv:2210.16771* (2022).

[99] M. Shu, W. Nie, D.-A. Huang, Z. Yu, T. Goldstein, A. Anandkumar, and C. Xiao. "Test-time prompt tuning for zero-shot generalization in vision-language models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14274–14289.

[100] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. "Visual prompt tuning". In: *European Conference on Computer Vision*. Springer. 2022, pp. 709–727.

[101] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. "Learning to prompt for vision-language models". In: *International Journal of Computer Vision* 130.9 (2022), pp. 2337–2348.

[102] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. "Conditional prompt learning for vision-language models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16816–16825.

[103] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[104] C. Shi and S. Yang. "Logoprompt: Synthetic text images can be good visual prompts for vision-language models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 2932–2941.

[105] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In.

[106] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. "Cats and dogs". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3498–3505.

[107] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. "Fine-grained visual classification of aircraft". In: *arXiv preprint arXiv:1306.5151* (2013).

[108] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015.

[109] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[110] A. Mathiasen and F. Hvilshøj. "Backpropagating through Fr\'echet Inception Distance". In: *arXiv preprint arXiv:2009.14075* (2020).