Graph-guided Cross-composition Feature Disentanglement for Compositional Zero-shot Learning

Anonymous ACL submission

Abstract

Disentanglement of visual features of primitives (i.e., attributes and objects) has shown exceptional results in Compositional Zero-shot Learning (CZSL). However, due to the feature divergence of an attribute (resp. object) when combined with different objects (resp. attributes), it is challenging to learn disentangled primitive features that are general across different compositions. To this end, we propose the solution of *cross-composition feature disentanglement*, which takes multiple primitive-sharing compositions as inputs and constrains the disentangled primitive features to be general across these compositions. More specifically, we leverage a compositional graph to define the overall primitive-sharing relationships between compositions, and build a task-specific architecture upon the recently successful large pre-trained vision-language model (VLM) CLIP, with dual cross-composition disentangling adapters (called L-Adapter and V-Adapter) inserted into CLIP's frozen text and image encoders, respectively. Evaluation on three popular CZSL benchmarks shows that our proposed solution significantly improves the performance of CZSL, and its components have been verified by solid ablation studies. Our code and data are available at: `https://anonymous.4open.science/r/DCDA-0BF7`.

1 Introduction

Compositional Zero-shot Learning (CZSL) aims to recognize novel attribute-object compositions by disentangling visual primitives from seen combinations, a capability crucial for scaling visual recognition systems (Misra et al., 2017). For instance, a model trained on *red tomato* and *green apple* should infer *green tomato* through primitive recombination, despite never encountering this specific composition. This paradigm not only enables zero-shot generalization to exponentially many combinations (Chen et al., 2023), but also advances vision-language understanding by requiring precise feature decomposition aligned with textual semantics (Chen et al., 2024).

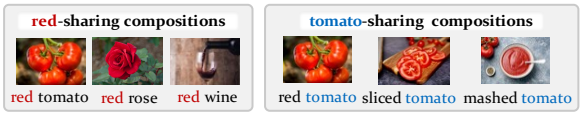

Figure 1: Examples of divergent visual features of a primitive (e.g., an attribute *red* or an object *tomato*) across different compositions.

Early CZSL approaches establish shared embedding spaces to compare image features with composition embeddings (Wei et al., 2019; Naeem et al., 2021; Mancini et al., 2022). Recent advances leverage CLIP's visual-semantic alignment from large-scale pretraining (Radford et al., 2021; Nayak et al., 2023), yet face inherent challenges: Attribute-object primitives exhibit strong visual entanglement—consider how *red* permeates all pixels of a *red tomato*. This entanglement hinders both primitive alignment and novel composition generalization. Current CLIP-based solutions address this through either disentangled text prompts (Lu et al., 2023; Wang et al., 2023a) or vision adapters (Zheng et al., 2024; Huang et al., 2024; Li et al., 2024), but crucially overlook the **diversity** of primitive manifestations across compositions. As Figure 1 illustrates, the visual realization of *red* varies significantly when combined with different objects (e.g., *tomato* vs. *wine*), exhibiting divergent color tones and spatial distributions.

To quantify this challenge, we conduct feature-space analysis on MIT-States (Isola et al., 2015) using CAILA (Zheng et al., 2024), the current SOTA method. As visualized in Figure 2 (left), disentangled attributes like *broken* (purple circles) exhibit two critical limitations: (1) Features from the same attribute scatter widely with overlapping clusters (e.g., *broken* intermixed with *cooked* and

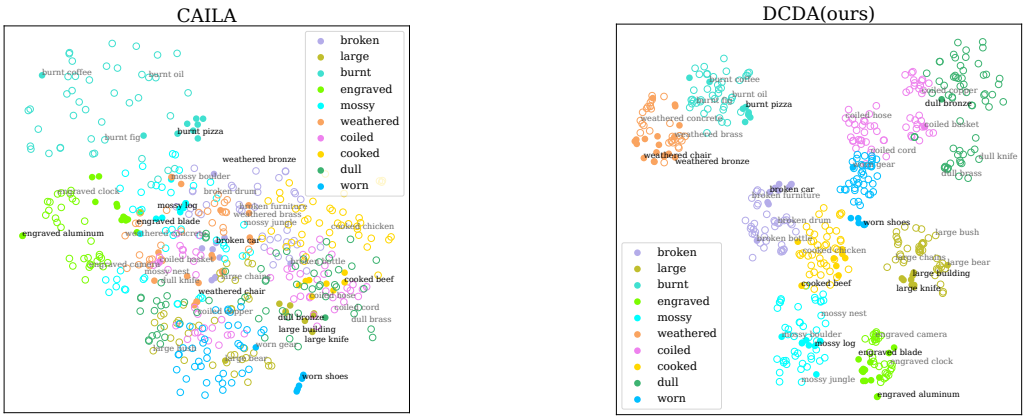

Figure 2: t-SNE visualizations of disentangled attribute representations of images in the test set of MIT-States, learned by CAILA (Zheng et al., 2024) and our DCDA. Solid and hollow circles represent images of seen and unseen compositions, respectively. Best viewed in color.

mossy), indicating poor intra-class consistency; (2) This dispersion directly impacts generalization—compositions containing unseen *broken* objects become indistinguishable due to the attribute's non-discriminative embeddings.

To overcome these limitations, we propose cross-composition feature aggregation through graph-guided learning. Our key insight is that effective primitive disentanglement requires cross-composition feature aggregation. Drawing inspiration from compositional graphs (Naeem et al., 2021), we construct a tripartite graph (Figure 3a) connecting attributes, objects, and their compositions. This graph enables our **D**ual **C**ross-composition Feature **D**ecomposing **A**dapters (**DCDA**), which enhance CLIP's text and image encoders through complementary strategies: **(1) L-Adapter** propagates textual features across compositionally related nodes via GNNs, consolidating attribute/object semantics from multiple contexts. **(2) V-Adapter** employs cross-attention between primitive-sharing images (e.g., *red tomato* and *red wine*) to extract invariant visual patterns, augmented by a novel sampling strategy that weightedly selects compositions according to the attribute/object co-occurrence degrees (derived from our graph). When integrating L-Adapters and V-Adapters into multiple layers of CLIP's text and image encoders, we retain the original parameters of CLIP to avoid overfitting, but inject the task-specific knowledge. Our contributions can be summarized below:

- DCDA is the first systematic approach for cross-composition feature disentanglement in CLIP-based CZSL, explicitly addressing primitive diversity.
- Dual adapter architecture leveraging compositional graphs for text-side aggregation and vision-side contrastive attention, enabling discriminative yet generalizable primitive representations.
- DCDA achieves great performance on MIT-States and UT-Zappos (closed/open world), with 5.1%/7.3% gains over CAILA method[1].

2 Related Work

Conventional CZSL methods are roughly divided into two groups. One is classifier-based which first trains two separate classifiers to predict an input image's attribute and object labels, respectively, and then combines them to predict the compositional labels (Misra et al., 2017; Nagarajan and Grauman, 2018). The subsequent works further enhance the dependence of the attribute and object in a composition (Li et al., 2020, 2022; Wang et al., 2023b). The other group is embedding-based which jointly represents attributes and objects to capture the dependence, and then aligns them with the images in a shared embedding space (Wei et al., 2019; Karthik et al., 2022). In particular, (Naeem et al., 2021) learn the joint representation through graph convolutional networks. There are also some works concerning the disentanglement of attribute and object features in the visual space (Saini et al.,

[1]Visualizations confirm tighter clustering of disentangled attributes (Figure 2, right), with extended object analyses in Appendix C.

2022; Hao et al., 2023; Kim et al., 2023). However, all these methods have to learn the alignment between image features and text embeddings from scratch and are prone to overfit to the seen compositions. It is expected to derive pre-trained alignment knowledge from VLMs.

CLIP-based CZSL. After pre-training using 400 million image-text pairs, CLIP can be applied to any visual classification task without fine-tuning by setting prompts like "a photo of [class]", where "[class]" is filled with the name of the class to be recognized. (Nayak et al., 2023) then had the first attempt to design prompt "a photo of [attribute] [object]" for CZSL, where "[attribute] [object]" are tunable tokens to teach CLIP how to compose attributes and objects.

To stress the roles of individual primitives, (Wang et al., 2023a) additionally set an attribute and an object prompt with only "[attribute]" or "[object]" tunable; (Lu et al., 2023) make the whole prompt trainable and fuses the decomposed text features with the encoded (entangled) image features through a cross-modal fusion module. Different from these works focusing on optimizing the prompts, (Zheng et al., 2024) propose to insert trainable adapters inside the frozen transformer layers to decompose and recompose the attribute and object features. With disentangled primitive features, (Huang et al., 2024) establish three prediction branches and pulls a static class prompt to its dynamic images via a cross-modal traction module. (Li et al., 2024) investigate the relative specificity of attributes when paired with different objects. In contrast to these methods, our method DCDA is more generalizable, with cross-composition knowledge injected and dual adapters inserted in CLIP's image and text encoders.

3 Methodology

CZSL Task Formulation. Let $\mathcal{D}_{tr} = \{(x, c)|x \in \mathcal{X}_s, c \in \mathcal{C}_s\}$ be the training set, where $\mathcal{X}_s$ contains the training images and $\mathcal{C}_s$ is a set of seen compositional labels that are available during training. Each label is a tuple $c = (a, o)$ of an attribute class $a \in \mathcal{A}$ and an object class $o \in \mathcal{O}$. After training, the CZSL model can predict images of a set of new compositions $\mathcal{C}_u$ that are unseen during training, with $\mathcal{C}_u \cap \mathcal{C}_s = \emptyset$. Following previous works, we study generalized CZSL (Purushwalkam et al., 2019), where images of seen and unseen compositions are both tested and the candidate label space includes both seen and unseen labels. The test set is thus denoted as $\mathcal{D}_{te} = \{(x, c)|x \in \mathcal{X}_{te}, c \in \mathcal{C}_{te}\}$, where $\mathcal{X}_{te} = \mathcal{X}_u \cup \mathcal{X}'_s$ with $\mathcal{X}'_s \cap \mathcal{X}_s = \emptyset$, and $\mathcal{C}_{te} = \mathcal{C}_u \cup \mathcal{C}_s$. Notably, $\mathcal{C}_s$ and $\mathcal{C}_u$ share the same attribute set $\mathcal{A}$ and object set $\mathcal{O}$, CZSL assumes that each a and o has been trained before testing and only the composition $(a, o) \in \mathcal{C}_u$ is novel.

Overview. In the following, we will first introduce the details of **L-Adapters** for the language side and **V-Adapters** for the vision side, and then introduce how to integrate them into the frozen CLIP encoders for CZSL. As shown in Figure 3(c), the adapters are inserted into CLIP's intermediate computational units such as self-attention layers or feed-forward layers. This means the input of each adapter is the output of CLIP's one computational unit, and its output is the input of CLIP's next computational unit. We use $\boldsymbol{H}^t \in \mathbb{R}^{l \times d}$ and $\boldsymbol{H}^v \in \mathbb{R}^{l' \times d'}$ to denote the output of CLIP's one specific computational unit in the text and image encoders, respectively, where l (resp. l') is the length of a tokenized input text (resp. image), d and d' is the hidden state size of each token.

3.1 The design of L-Adapter

Each L-Adapter is built upon a compositional graph for representing the global compositional relationships among attributes, objects and compositions, and a GNN module for propagating and aggregating features among them to realize the cross-composition learning of textual primitive features.

We first define the compositional graph. It consists of $N = |\mathcal{A}| + |\mathcal{O}| + |\mathcal{C}'|$ nodes, including all the attributes, all the objects, and the compositions in the current computation (i.e., $\mathcal{C}' = \mathcal{C}_s$ for training and $\mathcal{C}' = \mathcal{C}_{te}$ for testing). Given these nodes, for each $c = (a, o)$, we connect (a, o), (a, c) and (o, c) to form a triangle in the graph, as shown in Figure 3(a). For simplicity and efficiency, we keep all graph edges unweighted and undirected as in (Naeem et al., 2021) and obtain a symmetric adjacency matrix $A \in \mathbb{R}^{N \times N}$ to store the graph structure, $A_{ij} = 1$ if there is a connection between node i and j otherwise $A_{ij} = 0$.

To obtain the initial representations of graph nodes, we first define individual prompts for attributes and objects as "a photo of [attribute] object" and "a photo of [object]" besides the composition prompts "a photo of [attribute] [object]", then for each $c = (a, o)$, we feed three prompts into CLIP's text encoder to output three hidden states $\boldsymbol{H}^t_a$, $\boldsymbol{H}^t_o$ and $\boldsymbol{H}^t_c$ for a, o, c, respectively, and finally extract

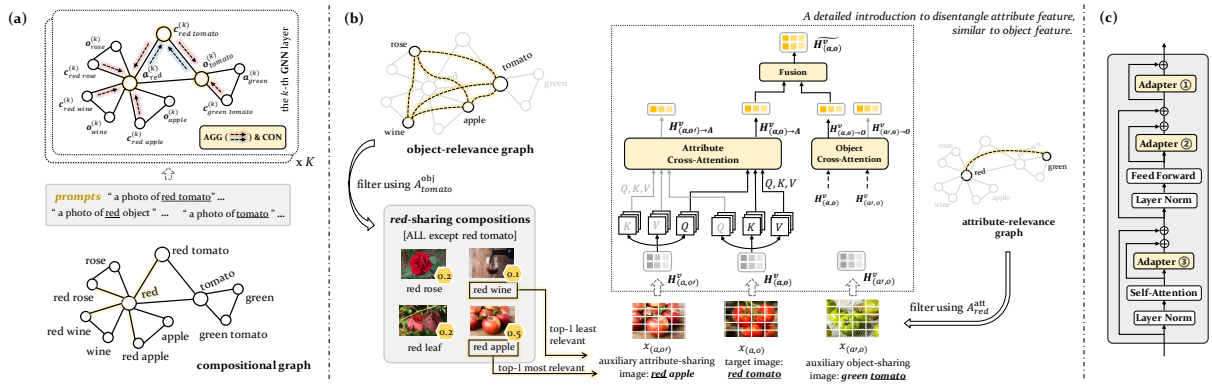

Figure 3: Overview of DCDA during training: (a) The L-Adapter built upon the composition graph and GNN module; (b) The V-adapter built upon the cross-attention and attribute/object relevance-guided sampling strategy; (c) An illustration of the candidate position for inserting adapters in a transformer block. We take the learning of *red tomato* as an example.

the embeddings of the special token [EOT] in the prompts as the initial features $\{\boldsymbol{h}_i^t\}_{i=1}^N$ of graph nodes, with $\boldsymbol{h}_i^t = \boldsymbol{H}_{i,\texttt{[EOT]}}^t$ and $\boldsymbol{h}_i^t \in \mathbb{R}^d$. In this way, the text feature of each primitive is naturally disentangled from the composition one.

Next, we exploit multiple GNN layers to propagate features among graph nodes following the graph structure defined in A. Formally, as Figure 3(a)'s top shows, for each $c = (a, o)$, three identical **AGG** functions are parallelly applied to aggregate neighborhood features for nodes a, o, c at the k-th GNN layer, $k \in \{0, ..., K-1\}$, as:

$$\boldsymbol{a}_{\mathcal{N}_a}^{(k)} = \text{AGG}^{(k)}(\{\boldsymbol{c}_i^{(k)} | c_i \in \mathcal{N}_a^c\}, \{\boldsymbol{o}_i^{(k)} | o_i \in \mathcal{N}_a^o\}) \quad (1)$$

$$\boldsymbol{o}_{\mathcal{N}_o}^{(k)} = \text{AGG}^{(k)}(\{\boldsymbol{c}_j^{(k)} | c_j \in \mathcal{N}_o^c\}, \{\boldsymbol{a}_j^{(k)} | a_j \in \mathcal{N}_o^a\}) \quad (2)$$

$$\boldsymbol{c}_{c=(a,o)}^{(k)} = \text{AGG}^{(k)}(\boldsymbol{a}^{(k)}, \boldsymbol{o}^{(k)}) \quad (3)$$

where $\mathcal{N}_a^c$ (resp. $\mathcal{N}_o^c$) denotes the composition neighbor set of a (resp. o) on the graph, and $\mathcal{N}_a^o$ (resp. $\mathcal{N}_o^a$) includes the objects (resp. attributes) that compose the compositions in $\mathcal{N}_a^c$ (resp. $\mathcal{N}_o^c$) together with a (resp. o). Considering the example in Figure 3(a) where $c = (a, o)$ is *red tomato*, $\mathcal{N}_a^c$ includes compositions like *red apple* and $\mathcal{N}_a^o$ includes objects like *apple*. While the neighbor set of each c only contains its primitives a and o. $\boldsymbol{a}^{(k)}$ is the input feature of a at the k-th layer, and is updated using **CON** function to obtain the k-th-layer output as $\boldsymbol{a}^{(k+1)} = \text{CON}(\boldsymbol{a}_{\mathcal{N}_a}^{(k)}, \boldsymbol{a}^{(k)})$, similar for o and c with outputs $\boldsymbol{o}^{(k+1)}$ and $\boldsymbol{c}^{(k+1)}$. The input feature of a, o, c at the first layer of GNN is the initialized node feature, e.g., $\boldsymbol{c}^{(0)} = \boldsymbol{h}_c^t$. The output features of a, o after K GNN layers $\boldsymbol{a}^{(K)}, \boldsymbol{o}^{(K)}$, which have already fused their neighboring compositions' features, and c's output feature $\boldsymbol{c}^{(K)}$, which has aggregated the updated features of a and o, are the final output of one L-Adapter, and will be inputted into the next computation unit of CLIP for the latter computation.

3.2 The design of V-Adapter

Since attributes and objects are highly entangled within the input image, we cannot build the same computational graph in V-Adapters as that in L-Adapters. Targeting this, we leverage the cross-attention over primitive-sharing image pairs to extract cross-composition-sharing primitive features, and design a primitive relevance-guided sampling strategy to introduce more valid primitive-sharing compositions. We take disentangling attribute features from an input image as an example, object features are processed similarly.

Specifically, as shown in Figure 3(b), given a target image $x_{(a,o)}$ to predict, which is labeled by $c = (a, o)$, we first randomly sample an auxiliary composition that shares the same attribute as $x_{(a,o)}$ but has different object o', and select one of its images $x_{(a,o')}$ as an auxiliary image. Then, we feed these two images into CLIP to output two hidden states $\boldsymbol{H}_{(a,o)}^v$ and $\boldsymbol{H}_{(a,o')}^v$ and compute the cross-attention as, with $\boldsymbol{H}_{(a,o')}^v$ as the query and $\boldsymbol{H}_{(a,o)}^v$ as the key and value:

$$\text{CrossAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d'}})\boldsymbol{V} \quad (4)$$

$$\boldsymbol{Q} = \boldsymbol{H}_{(a,o')}^v \boldsymbol{W}_Q, \boldsymbol{K} = \boldsymbol{H}_{(a,o)}^v \boldsymbol{W}_K, \boldsymbol{V} = \boldsymbol{H}_{(a,o)}^v \boldsymbol{W}_V \quad (5)$$

where $\boldsymbol{W}_{\{Q,K,V\}} \in \mathbb{R}^{d' \times d'}$ are three linear transformation matrices for flexible computation. We can see that every output embedding is a weighted sum of the value embeddings, and the weights are calculated by the similarity of the query and the key. By setting query as $\boldsymbol{H}_{(a,o')}^v$, we can refine $\boldsymbol{H}_{(a,o)}^v$ to keep features that are more specific to a, as well as attribute features that are general across these two compositions. We also swap the query and the value (also key) to refine $\boldsymbol{H}_{(a,o')}^v$. The output of cross-attention is thus denoted as $\boldsymbol{H}_{(a,o)\to A}^v$

and $\boldsymbol{H}^v_{(a,o')\to A}$. A feed-forward layer is also added after the cross-attention.

Notably, the above cross-attention can only process two attribute-sharing compositions at one time. To introduce more compositions to learn more general attribute features, the model relies on the batched data and random sampling to switch the auxiliary composition. However, when an attribute is diverse with extensive composition neighbors, e.g., *red* or *broken*, which also means extensive candidate auxiliary compositions, the model requires more switches to traverse them, leading to inferior overall performance as shown in Table 1. To balance the switch times of attributes with different numbers of neighbors, we propose to select some representative compositions instead of all the compositions as the candidates, and for a target image $x_{(a,o)}$, the top-n objects that are most and least relevant to the target object o are paired with the target attribute a to serve as the representative auxiliary compositions, while the relevance between two objects can be determined by the number of common attributes that co-occur in their associated compositions in the training set; for example, if there are compositions *red tomato* and *red apple* in the training set, *red* is a common attribute of the objects *tomato* and *apple*.

To this end, we refer to the attribute-object edges in the training compositional graph to first create an **att-obj** graph with structure matrix $A^{\text{att-obj}} \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{O}|}$, where $A^{\text{att-obj}}_{i,j} = 1$ means attribute i and object j form a valid seen composition while 0 not. Then, an object relevance graph can be created and its structure matrix is found as $A^{\text{obj}} = (A^{\text{att-obj}})^T A^{\text{att-obj}}$ with size $|\mathcal{O}| \times |\mathcal{O}|$, where $A^{\text{obj}}_{i,j}$ represents the number of common attributes between i-th and j-th objects, large numbers mean higher relevance. With this relevance graph, for target image $x_{(a,o)}$, and all of its a-sharing compositions besides (a, o), denoted as $\{(a, o')\}$, we next refer to A^{obj}_o to obtain the relevance scores of objects in $\{o'\}$ w.r.t o, and select the compositions whose objects have top-n maximum and top-n minimum non-zero scores as the representative compositions. Figure 3(b) presents a running example of this procedure. Finally, we perform a weighted random sampling over these representative compositions, i.e., the probability for selecting (a, o') is determined by the normalized relevance score between o' and o, more details are attached in Appendix A.1.

The same applies to the output of the object cross-attention $\boldsymbol{H}^v_{(a,o)\to O}$, which is learned from a set of representative o-sharing compositions selected by referring to the attribute relevance matrix $A^{\text{att}} = A^{\text{att-obj}}(A^{\text{att-obj}})^T$ and its row value A^{att}_a. In this way, we learn the cross-composition-sharing primitive features of an input image, and have the updated image features: $\widetilde{\boldsymbol{H}}^v_{(a,o)} = \boldsymbol{H}^v_{(a,o)\to A} + \boldsymbol{H}^v_{(a,o)\to O}$, which will be the final output of one V-Adapter together with $\boldsymbol{H}^v_{(a,o)\to A}$ and $\boldsymbol{H}^v_{(a,o)\to O}$.

3.3 Integrating Adapters into CLIP

Given a single L-Adapter and V-Adapter, we next present how to insert them into two connected computation units in CLIP. Inspired by ViT (Dosovitskiy et al., 2020) which tends to learn general features at lower layers and learn specific features at higher layers, we add our adapters starting from the top transformer blocks. After preliminary validations on MIT-States (see our Ablation Study results for more), we decided to *i)* insert adapters at the last three transformer blocks of both text and image encoders, and *ii)* add L-Adapters behind the self-attention layer and feed-forward layer in each language transformer block (i.e., the positions ③ and ② in Figure 3(c)), and attach V-Adapters after the whole vision transformer block (i.e., the position ① in Figure 3(c)). Moreover, we build a skip connection between the input and the output of each adapter before inputting it into the next computation unit of CLIP. We use CLIP's pre-trained word embeddings to initialize each word in our prompts, including the prefix words "a photo of", and keep these word embeddings trainable for capturing more task-specific knowledge.

3.4 Training and Inference

At the last transformer blocks, we obtain the output of L-Adapter and V-Adapter after skip connection, denoted as $\widehat{\boldsymbol{H}}^v$ and $\widehat{\boldsymbol{H}}^t$, based on this, we extract the embeddings of `[EOT]` token, i.e., $\widehat{\boldsymbol{h}}^v(=\widehat{\boldsymbol{H}}^v_{\texttt{[EOT]}})$ and $\widehat{\boldsymbol{h}}^t(=\widehat{\boldsymbol{H}}^t_{\texttt{[EOT]}})$, to measure the compatibility of visual and textual features. Formally, for an input image x_i and a training composition $c_i = (a_i, o_i)$, we compute the compatibility score as:

$$s(x_i, c_i = (a_i, o_i)) = \alpha\,[\widehat{\boldsymbol{h}}^v_i \cdot \widehat{\boldsymbol{h}}^t_{c_i}] + \beta\,[\widehat{\boldsymbol{h}}^v_{i\to A} \cdot \widehat{\boldsymbol{h}}^t_{a_i}] + \gamma\,[\widehat{\boldsymbol{h}}^v_{i\to O} \cdot \widehat{\boldsymbol{h}}^t_{o_i}] \quad (6)$$

where we also measure the individual primitive compatibility via the second and the third items as if one image belongs to an attribute-object pair, its

disentangled attribute and object features also belong to the corresponding primitive labels. We use three learnable parameters α, β, γ to balance the overall score. $\cdot$ denotes the dot-product similarity.

We optimize these adapters and trainable token embeddings by minimizing the cross-entropy loss on the training set $\mathcal{D}_{tr}$ with seen compositions from $\mathcal{C}_s$, with τ as the temperature widely used in CLIP, and $c_i = (a_i, o_i)$ as the ground-truth label:

$$\mathcal{L} = -\frac{1}{|\mathcal{D}_{tr}|} \sum_{x_i \in \mathcal{D}_{tr}} log \frac{e^{[s(x_i, c_i=(a_i,o_i))/\tau]}}{\sum_{c_j \in \mathcal{C}_s} e^{[s(x_i, c_j=(a_j,o_j)/\tau)]}} \quad (7)$$

During inference, for each testing image x_t, we estimate the compatibility score between x_t and each testing composition $c = (a, o)$ from $\mathcal{C}_{te}$ as Equation 6. The composition that has the highest compatibility score is the predicted label. Besides, since we have no idea about the attribute and object labeled for x_t, we take itself as the primitive-sharing images to compute $\boldsymbol{H}^v_{t\to A}$ and $\boldsymbol{H}^v_{t\to O}$.

4 Experiments

4.1 Experimental Settings

Datasets and Metrics. We experiment with three popular benchmarks for CZSL: MIT-States (Isola et al., 2015), UT-Zappos (Yu and Grauman, 2014) and C-GQA (Naeem et al., 2021), and follow (Purushwalkam et al., 2019; Naeem et al., 2021) to split the data for training, validation and testing. For each dataset, we compute the prediction accuracy of seen and unseen compositions, and report four metrics: the best Seen (**S**) and Unseen (**U**) accuracy, the best harmonic mean (**H**), and the Area Under the accuracy Curve (**AUC**). Among them, AUC is the most comprehensive one and is widely adopted as the core metric by previous works (Purushwalkam et al., 2019; Zheng et al., 2024). Please see Appendix B for more datasets and metrics details.

Closed World and Open World Settings. Given the attribute set $\mathcal{A}$ and object set $\mathcal{O}$, the complete compositional label set $\mathcal{C}$ should be the Cartesian product of $\mathcal{A}$ and $\mathcal{O}$, i.e., $\mathcal{C} = \mathcal{A} \times \mathcal{O}$ with size $|\mathcal{A}| \times |\mathcal{O}|$. However, current benchmarks often operate in a *closed world* setting where unseen compositions $\mathcal{C}_u$ in $\mathcal{C}_{te}$ are a small subset of $\mathcal{C} \setminus \mathcal{C}_s$ and are assumed to be known. For example, MIT-States contains 28,175 possible compositions with 115 attributes and 245 objects, but the label space for testing is limited to 1,962 compositions (1,262 seen and 700 unseen), covering less than 7% of the complete set. Thus, we follow (Mancini et al., 2021) to evaluate our model in the *open world* setting, where the testing images remain unchanged but the testing label space is all possible combinations, i.e., $\mathcal{C}_{te} = \mathcal{C}$ and $\mathcal{C}_u = \mathcal{C} \setminus \mathcal{C}_s$, which is more challenging as the models have to generalize from a small set of seen to a very large set of unseen compositions. Notably, not all the combinations are feasible, such as *eroded cat*, for this, we apply post-training calibration (Nayak et al., 2023; Xu et al., 2024) to filter out unreasonable compositions.

Baselines and Model Variants. We mainly compare our DCDA with the existing CLIP-based CZSL methods, including the vanilla CLIP without fine-tuning, CSP (Nayak et al., 2023), HPL (Wang et al., 2023a), DFSP (Lu et al., 2023), CAILA (Zheng et al., 2024), and Troika (Huang et al., 2024). We also include two non-CLIP-based baselines that are most similar to us, namely CGE (Naeem et al., 2021) and ADE (Hao et al., 2023).

In V-Adapters, we propose a novel primitive-relevance guided (PRG) sampling method to select representative auxiliary compositions to sample rather than performing purely random (RD) sampling over all neighboring compositions. For detailed comparisons, we develop a variant DCDA[RD] and denote the vanilla model as DCDA[PRG]. With the imbalanced sample distribution, we also focus on the tail compositions in the neighbor set, and sample them according to the reciprocal of their image numbers (N) to supply DCDA[PRG], leading to a new variant DCDA[PRG+N], these two sampling strategies are switched batch by batch. Please see Appendix B.3 for more implementation details.

4.2 Main Results

Closed-world Performance. As shown in Table 1 (top), our DCDA variants achieve state-of-the-art performance across benchmarks. On MIT-States, both DCDA[PRG] and DCDA[PRG+N] surpass previous methods with **over 3% improvements** in AUC and harmonic mean (H), and **over 6% gains** in seen accuracy compared to CAILA. For UT-Zappos, DCDA[PRG+N] is the best on three metrics and achieves **substantial AUC improvements** over CGE despite the inferior result on H. The initial experiments showed suboptimal performance on C-GQA due to its low-quality images, unfreezing CLIP's image encoder eventually enabled DCDA to achieve competitive second-place results, see Appendix B.4 for more.

Setting	Methods	MIT-States				UT-Zappos				C-GQA			
		S	U	H	AUC	S	U	H	AUC	S	U	H	AUC
Closed World	CGE	32.8	28.0	21.4	6.5	64.5	71.5	**60.5**	33.5	33.5	15.5	16.0	4.2
	ADE	–	–	–	–	–	–	–	–	35.0	17.7	18.0	5.2
	CLIP	30.2	46.0	26.1	11.0	15.8	49.1	15.6	5.0	7.5	25.0	8.6	1.4
	CSP	46.6	49.9	36.3	19.4	64.2	66.2	46.6	33.0	28.8	26.8	20.5	6.2
	HPL	47.5	50.6	37.3	20.2	63.0	68.8	48.2	35.0	30.8	28.4	22.4	7.2
	DFSP	46.9	52.0	37.3	20.6	66.7	71.7	47.2	36.0	*37.3*	*26.1*	*23.5*	*8.2*
	CAILA	51.0	53.9	39.9	23.4	67.8	74.0	57.0	44.1	***40.4***	28.6	***26.1***	***9.9***
	Troika	49.0	53.0	39.3	22.1	66.8	73.8	54.6	41.7	*38.0*	*28.4*	25.3	*9.2*
	DCDA[RD]	42.2	46.7	32.8	16.2	64.7	71.5	54.5	40.1	39.8	*25.3*	*23.9*	*8.5*
	DCDA[PRG]	**57.3**	55.1	**43.2**	26.9	68.7	72.4	56.5	43.0	*39.1*	*26.7*	*24.5*	*8.9*
	DCDA[PRG+N]	57.1	**55.5**	43.1	**27.0**	**69.1**	**74.1**	57.2	**44.2**	*38.5*	***28.8***	25.3	9.4
Open World	CGE	32.4	5.1	6.0	1.0	61.7	47.7	39.0	23.1	32.1	1.8	2.9	0.47
	ADE	–	–	–	–	–	–	–	–	35.1	4.8	7.6	1.42
	CLIP	30.1	14.3	12.8	3.0	15.7	20.6	11.2	2.2	7.5	4.6	4.0	0.27
	CSP	46.3	15.7	17.4	5.7	64.1	44.1	38.9	22.7	28.7	5.2	6.9	1.20
	HPL	46.4	18.9	19.8	6.9	63.4	48.1	40.2	24.6	30.1	5.8	7.5	1.37
	DFSP	47.5	18.5	19.3	6.8	66.8	60.0	44.0	30.3	*35.0*	*4.9*	*7.3*	*1.42*
	CAILA	51.0	20.2	21.6	8.2	67.8	59.7	49.4	32.8	***40.4***	***6.6***	***9.6***	***2.26***
	Troika	48.8	18.7	20.1	7.2	66.4	61.2	47.8	33.0	37.4	*4.5*	*6.0*	*1.11*
	DCDA[PRG]	54.6	27.3	25.8	11.5	**68.6**	56.4	51.2	33.8	*35.5*	*4.4*	*6.7*	*1.30*
	DCDA[PRG+N]	**55.0**	**27.7**	**26.7**	**12.0**	67.8	**62.5**	**51.4**	**35.8**	*35.3*	6.4	8.5	1.76

Table 1: Overall Results (%) on three benchmarks. In each setting, the best results are in bold and the second best are underlined. We report DFSP in its $t2i$ setting. Numbers in italics mean the results implemented with CLIP-base, others are with CLIP-large.

Model Variants Analyses. The significant margin between DCDA[PRG] and DCDA[RD] on all datasets validates our primitive-relevance guided sampling strategy. Especially, the improvement on MIT-States is promising, since the number of compositions surrounding its each primitive is more imbalanced than that of UT-Zappos. Moreover, in contrast to the slight performance gap between DCDA[PRG+N] and DCDA[PRG] on MIT-States and C-GQA, DCDA[PRG+N] shows great superiority on UT-Zappos, demonstrating superior handling of class imbalance (sample size std: UT-Zappos's 465 vs. MIT-States' 12 vs. C-GQA's 22).

Cross-method Insights. Among CLIP-based methods, CAILA, Troika, and DCDA form the top tier by injecting adapters for visual disentanglement. Our approach outperforms both competitors on both the general dataset MIT-States and the domain-specific dataset UT-Zappos with fewer trainable parameters through partial-layer adapter insertion (our last-3 layers vs. full layers in CAILA and Troika), demonstrating superior cross-domain adaptability, and preserving CLIP's generalization as evidenced by open-world results (Table 1, bottom). The comparable results on C-GQA also motivate us to develop more advanced learning paradigms in the future.

4.3 Effectiveness of Adapters

The whole Adapters. We evaluate the contribution of L-Adapter and V-Adapter by analyzing the performance drop when one of them is removed.

Models	S	U	H	AUC
Full Model (e.g., DCDA[PRG])	57.3	**55.1**	**43.2**	**26.9**
w/o L-Adapters	55.9	54.7	42.2	26.1
w/o V-Adapters	44.9	46.9	33.8	17.1
L-Adapter w/o other compositions	**57.5**	54.6	43.0	26.7
V-Adapter w/o other compositions	44.5	46.2	33.7	17.0
L&V-Adapter w/o other compositions	44.2	46.1	33.6	16.7

Table 2: Adapter Analysis on MIT-States in *closed world*.

Notably, when all L-Adapters (reps. V-Adapters) are removed, the text (resp. image) encoder will act like CLIP's default frozen encoders to output an entangled representation for each input prompt (resp. image). We conduct experiments on MIT-States dataset under the *closed world* setting, the results are shown in the second and third lines of Table 2. We can see that the performance both declines when L-Adapters or V-Adapters are removed, indicating that they two both have a positive contribution to the DCDA model and are complementary to each other. We also observe that the performance decrease of removing V-Adapters is greater than that of removing L-Adapters, which is consistent with our statement: textual primitive features are less entangled than visual ones, and the independent textual primitive features can still be captured by setting individual primitive prompts.

Cross-composition Information in Adapters. We further validate the effectiveness of introducing primitive-sharing compositions for each target composition by deleting the compositional graph in L-Adapters and/or auxiliary compositions in V-

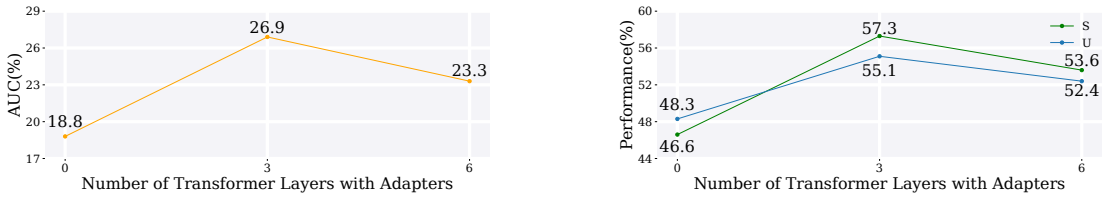

Figure 4: Performance (AUC, S and U) of increasing the number of transformer layers with adapters on MIT-States.

Adapters. Concretely, we merely keep the prompts' token embeddings trainable in L-Adapters; and/or replace the two auxiliary images with the target image itself in V-Adapters, which thus turns into the self-attention on the input image, but still projects the primitive features into different subspaces with different attention networks. The results on MIT-States are shown in the last three lines of Table 2. The performance drop indicates the superiority of introducing primitive-sharing compositions in both text and image encoders. In particular, we find that V-Adapters without primitive-sharing compositions even perform worse than removing the whole V-Adapters, illustrating that these compositions play a considerable role in constraining the learning of primitive features in different subspaces. In contrast, L-Adapters without neighboring (primitive-sharing) compositions perform better than removing the whole L-Adapters. However, the performance gap is slight, which may be attributed to that we also set the embeddings of tokens in the prompts tunable, the cross-composition primitive features can be implicitly captured by optimizing the token embeddings with multiple samples.

4.4 Ablation Studies

Insertion Location of Adapters. There are two choices for inserting adapters into a transformer block, i.e., inside or outside. Therefore, we experiment with four configurations where every L-Adapter and V-Adapter are added inside or outside a transformer block, as shown in Table 3. Notably, the inside insertion includes adding after the self-attention and feed-forward layers in a block, i.e., the positions ③ and ② in Figure 3(c). From Table 3, it can be seen that adding V-Adapters outside transformer layers often achieves better performance no matter where L-Adapters are located, while there is no significant difference when shifting L-Adapters at different positions. This may be because when adding a V-Adapter inside the transform layer, there is no nonlinear transformation like Feed-Forward layer between two attention operations for extracting more informative features. Regarding the better performance with inside L-Adapters and outside V-Adapters, we finally apply this configuration to three benchmarks.

L-Adapters		V-Adapters		MIT-States			
I	O	I	O	S	U	H	AUC
✓		✓		53.5	53.4	40.6	24.0
✓			✓	**57.3**	55.1	**43.2**	**26.9**
	✓	✓		53.1	53.0	41.0	24.0
	✓		✓	56.0	**55.5**	**43.2**	26.7

Table 3: Ablation Study on Adapters' Insertion Locations, i.e., inside (I) or outside (O) one transformer block.

Insertion Depth of Adapters. We further show the performance change when we increase the number of transformer blocks with trainable adapters in Figure 4. As mentioned earlier, we start from the top transformer layers. It is clear that the best performance is achieved with the last 3 layers, while the last 6 layers may overfit the training data, with less generalization knowledge from our adapters.

5 Conclusions and Outlook

We present DCDA, a graph-guided framework that enhances CLIP for compositional zero-shot learning through dual adapters. Our key innovations include: 1) L-Adapters that aggregate textual primitives via compositional graph propagation, addressing label-side entanglement; 2) V-Adapters that extract invariant visual patterns through cross-attention and primitive-relevance guided sampling, effectively mitigating image feature entanglement. Experiments across three benchmarks validate that DCDA learns more discriminative and generalizable primitive representations. This underscores the critical role of visual disentanglement in CZSL, as visual primitives exhibit stronger composition-dependent entanglement than textual counterparts. In the future, it is expected to apply our dual adapters in other vision-language tasks and develop dynamic graph construction mechanisms to handle open-vocabulary primitive discovery.

Limitations

While our proposed DCDA demonstrates strong performance in compositional zero-shot learning (CZSL), several limitations warrant discussion as follows: (1) the computational overhead of constructing and updating the compositional graph grows with the scale of attributes and objects, which may pose challenges for applications requiring real-time inference on large combinatorial spaces. (2) while our primitive-relevance guided sampling mitigates data imbalance, extreme long-tailed distributions of attributes or objects (e.g., rare primitives with few compositions) may still lead to suboptimal disentanglement due to insufficient cross-composition supervision. Addressing these limitations could further enhance the robustness and scalability of our framework.

References

Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Jeff Z Pan, Yuan He, Wen Zhang, Ian Horrocks, and Huajun Chen. 2023. Zero-shot and few-shot learning with knowledge graphs: A comprehensive survey. *Proceedings of the IEEE*, pages 653–685.

Zhuo Chen, Yichi Zhang, Yin Fang, Yuxia Geng, Lingbing Guo, Xiang Chen, Qian Li, Wen Zhang, Jiaoyan Chen, Yushan Zhu, et al. 2024. Knowledge graphs meet multi-modal learning: A comprehensive survey. *arXiv preprint arXiv:2402.05391*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Shaozhe Hao, Kai Han, and Kwan-Yee K Wong. 2023. Learning attention as disentangler for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15315–15324.

Siteng Huang, Biao Gong, Yutong Feng, Min Zhang, Yiliang Lv, and Donglin Wang. 2024. Troika: Multi-path cross-modal traction for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24005–24014.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Phillip Isola, Joseph J Lim, and Edward H Adelson. 2015. Discovering states and transformations in image collections. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1383–1391.

Shyamgopal Karthik, Massimiliano Mancini, and Zeynep Akata. 2022. Kg-sp: Knowledge guided simple primitives for open world compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9336–9345.

Hanjae Kim, Jiyoung Lee, Seongheon Park, and Kwanghoon Sohn. 2023. Hierarchical visual primitive experts for compositional zero-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5675–5685.

Xiangyu Li, Xu Yang, Kun Wei, Cheng Deng, and Muli Yang. 2022. Siamese contrastive embedding network for compositional zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9326–9335.

Yong-Lu Li, Yue Xu, Xiaohan Mao, and Cewu Lu. 2020. Symmetry and group in attribute-object compositions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11316–11325.

Yun Li, Zhe Liu, Hang Chen, and Lina Yao. 2024. Context-based and diversity-driven specificity in compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17037–17046.

Xiaocheng Lu, Song Guo, Ziming Liu, and Jingcai Guo. 2023. Decomposed soft prompt guided fusion enhancing for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23560–23569.

Massimiliano Mancini, Muhammad Ferjad Naeem, Yongqin Xian, and Zeynep Akata. 2021. Open world compositional zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5222–5230.

Massimiliano Mancini, Muhammad Ferjad Naeem, Yongqin Xian, and Zeynep Akata. 2022. Learning graph embeddings for open world compositional zero-shot learning. *IEEE Transactions on pattern analysis and machine intelligence*.

Ishan Misra, Abhinav Gupta, and Martial Hebert. 2017. From red wine to red tomato: Composition with context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1160–1169.

Muhammad Ferjad Naeem, Yongqin Xian, Federico Tombari, and Zeynep Akata. 2021. Learning graph embeddings for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 953–962.

Tushar Nagarajan and Kristen Grauman. 2018. Attributes as operators: factorizing unseen attribute-object compositions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–190.

Nihal V. Nayak, Peilin Yu, and Stephen H. Bach. 2023. Learning to compose soft prompts for compositional zero-shot learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.

Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc'Aurelio Ranzato. 2019. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3592–3601.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Nirat Saini, Khoi Pham, and Abhinav Shrivastava. 2022. Disentangling visual embeddings for attributes and objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13658–13667.

Henan Wang, Muli Yang, Kun Wei, and Cheng Deng. 2023a. Hierarchical prompt learning for compositional zero-shot recognition. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 1470–1478.

Qingsheng Wang, Lingqiao Liu, Chenchen Jing, Hao Chen, Guoqiang Liang, Peng Wang, and Chunhua Shen. 2023b. Learning conditional attributes for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206.

Kun Wei, Muli Yang, Hao Wang, Cheng Deng, and Xianglong Liu. 2019. Adversarial fine-grained composition learning for unseen attribute-object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3740–3748.

Guangyue Xu, Joyce Chai, and Parisa Kordjamshidi. 2024. Gipcol: Graph-injected soft prompting for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5774–5783.

Aron Yu and Kristen Grauman. 2014. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 192–199.

Zhaoheng Zheng, Haidong Zhu, and Ram Nevatia. 2024. Caila: Concept-aware intra-layer adapters for compositional zero-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1721–1731.

A Supplementary Methodology Details

A.1 Computing the Sampling Probability

To sample the top-n most and least relevant compositions according to the top-n maximum and minimum primitive relevance scores, we take two ways to compute the sampling probabilities. More specifically, the probability for selecting the top-n most relevant compositions, e.g., (a, o'_i), is computed as:

$$P(o' = o'_i) = \frac{A^{\text{obj}}_{o,o'_i}}{\sum_{o'_k \in \{o'\}} A^{\text{obj}}_{o,o'_k}},$$

where A^{obj}_{o,o'_i} means the relevance score between objects o'_i and o, while the probability for selecting the top-n least relevant compositions, e.g., (a, o'_j), is computed as:

$$P(o' = o'_j) = -\frac{A^{\text{obj}}_{o,o'_j}}{\sum_{o'_k \in \{o'\}} A^{\text{obj}}_{o,o'_k}}.$$

The difference is that the n normalized minimum scores are multiplied by -1 to ensure that the top-1 least relevant composition has the highest probability of being sampled among the n least relevant compositions. Consider the example in Figure 3(b), *red wine, red rose* and *red leaf* are top-3 least relevant compositions for *red tomato*, with top-3 normalized minimum object-relevance scores: $0.1, 0.2$ and 0.2, respectively, while *red wine* is the top-1 least relevant composition with the highest sampling probability among these three compositions.

To avoid being confused by the positive and negative sampling probabilities, we divide the representative compositions into two groups and sample over the top-n most and least relevant compositions independently. To be more specific, we switch them batch by batch.

B Supplementary Experiment Details

B.1 Dataset

MIT-States contains 53,753 real-world images, annotated by a variety of classes with 245 objects and their 115 attributes in the general domain. In *closed world*, it provides 1,962 compositions in total, 1,262 of which are seen used for training, and 700 are unseen with 300 for validation and 400 for testing. UT-Zappos is a more domain-specific dataset, containing 50,025 images of shoes paired

with their material attributes. In total, it has 16 attributes and 12 objects, yielding 83 seen and 33 unseen compositions under the *closed world* setting. C-GQA, derived from of Stanford GQA dataset (Hudson and Manning, 2019), is the most extensive dataset for CZSL, containing 7,767 compositions (5,592/2,175 as seen/unseen), 413 attribute classes, 674 object classes, and 39,298 images in total. Table 4 summarizes the statistics of these three datasets in the *closed world* setting, the *open world* has the same set of testing images, i.e., $\mathcal{X}_{te}$, but larger candidate label set, i.e., all possible compositions $\mathcal{C}$ obtained by the Cartesian product of $\mathcal{A}$ and $\mathcal{O}$.

Datasets	Composition	Train		Validation		Test	
	$\lvert\mathcal{A}\rvert$ / $\lvert\mathcal{O}\rvert$	$\lvert\mathcal{C}_s\rvert$	$\lvert\mathcal{X}_s\rvert$	$\lvert\mathcal{C}_s\rvert$ / $\lvert\mathcal{C}_u\rvert$	$\lvert\mathcal{X}_{val}\rvert$	$\lvert\mathcal{C}_s\rvert$ / $\lvert\mathcal{C}_u\rvert$	$\lvert\mathcal{X}_{te}\rvert$
MIT-States	115 / 245	1,262	30,338	300 / 300	10,420	400 / 400	12,995
UT-Zappos	16 / 12	83	22,998	15 / 15	3,214	18 / 18	2,914
C-GQA	413 / 674	5,592	26,920	1,040 / 1,252	7,280	888 / 923	5,098

Table 4: Statistics of Datasets for CZSL. $\lvert\mathcal{C}_u\rvert$ here is the number of unseen compositions in the *closed world* setting.

B.2 Evaluation Metrics

We compute the prediction accuracy for recognizing seen and unseen compositions, i.e., the generalized CZSL, in both *closed world* and *open world* scenarios. Specifically, due to the inherent bias towards seen classes, we follow the current standard (Purushwalkam et al., 2019; Nayak et al., 2023) to add a scalar bias to the prediction scores of unseen classes and vary the bias from $-\infty$ to $+\infty$ to get a seen-unseen accuracy curve, which indicates the seen accuracy on the x-axis and unseen accuracy on the y-axis. Correspondingly, we can report the best seen accuracy (**S**), where the bias is set to $-\infty$ and the models only predict on the seen labels, and report the best unseen accuracy (**U**), where the bias is set to $+\infty$ and the models only predict on the unseen labels. Also, we calculate the best harmonic mean (**H**), where a harmonic mean value is first computed for each point on the curve to balance the seen accuracy (acc_s) and unseen accuracy (acc_u) as $(2 \times acc_s \times acc_u)/(acc_s + acc_u)$, and then the highest value across all the selected points is reported. Finally, we compute the Area Under the accuracy Curve (**AUC**) as a comprehensive metric.

B.3 Implementation Details

We implement our models with PyTorch and use Adam as the optimizer, with the learning rate set to 5e-5, 5e-5, 1e-5, and the batch size set to 32, 32, 16 for MIT-States, UT-Zappos, C-GQA, respectively, the weight decay is set to 5e-5 for all datasets. The GNN module is implemented as GCN with $K = 2$. n for selecting the representative auxiliary compositions is set to 5. The initialized value of α, β, γ are all set to 1, and later optimized together with other parameters. The optimum configurations w.r.t these hyper-parameters are determined by AUC on the validation set. All the experiments are run on a single NVIDIA Tesla A100 GPU with 40GB memory.

B.4 Experiments on C-GQA

To deal with the low-resolution and small-size images in C-GQA, we tried to add more trainable parameters, such as inserting more adapters in CLIP's image encoder or fully fine-tuning the whole image encoder, to adapt CLIP to this kind of image. More specifically, we add a downsample-upsample style adapter, which is similar to the adapter used in CAILA and Troika, after the self-attention layer and feed-forward layer in each vision transformer block (i.e., the positions ③ and ② in Figure 3(c)), except for the last three transformer blocks where our V-Adapters have already been there. In addition, we also try to unfreeze the whole image encoder to fully update its parameters, where our V-Adapters are added in the last three layers for feature disentanglement. The results are presented in the second and third lines of Table 5, respectively. Moreover, since a single A100 GPU with 40GB memory can not afford these extra adapters or fully fine-tuned parameters with CLIP-large that contains 24 vision transformer (ViT) layers, we instead use CLIP-base with ViT-B/32 as its image encoder, and re-run baselines for a fair comparison. Regarding that some baselines with CLIP-large still perform worse than our methods with CLIP-base, we omit re-implementing them to save computation costs.

From Table 5, we can see that introducing more trainable parameters indeed achieves superior performance, in comparison with the vanilla models that only include 3 V-Adapters in the last three layers of the frozen image encoder. Especially, the fine-tuning method performs better. As a result, we implement our DCDA[PRG] and DCDA[PRG+N] with the whole image encoder fully tunable, the resulting models together with our V-Adapters achieve very competitive performance on C-GQA compared with the SOTA CAILA. Moreover, we

Models	S	U	H	AUC
Frozen Encoder + 3 V-Adapters (default)	34.8	23.4	21.6	6.9
Extra Adapters + 3 V-Adapters	38.8	26.6	23.8	8.7
Full Fine-Tuning + 3 V-Adapters	38.5	**28.8**	**25.3**	**9.4**
Full Fine-Tuning + 1 V-Adapters	38.9	26.4	24.6	8.8
Full Fine-Tuning + 2 V-Adapters	**39.2**	27.3	24.9	9.0
Full Fine-Tuning + 4 V-Adapters	38.8	25.6	24.2	8.5
Full Fine-Tuning + 6 V-Adapters	31.9	13.7	14.1	3.5

Table 5: Ablation on adding more trainable parameters in the vision encoder on C-GQA in *closed world*. All the results are tested with the "[PRG+N]" sampling method.

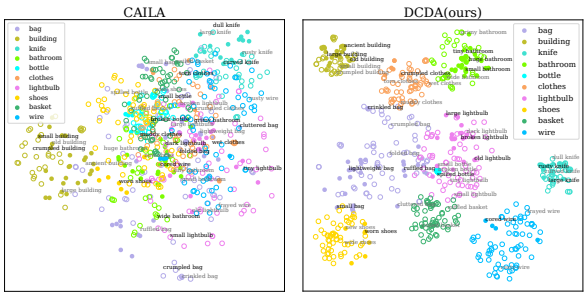

Figure 5: t-SNE visualizations of disentangled object representations of images in the test set of MIT-States, learned by CAILA and our DCDA. Solid and hollow circles represent images of seen and unseen compositions, respectively. Best viewed in color.

also vary the number of transformer layers with V-Adapters inserted, starting from the top transformer layers, the results are as shown in the last four lines of Table 5. From Table 5, we have similar observations as in Figure 4, i.e., too few (e.g., only one) V-Adapter is not enough to disentangle the primitive features, while too many (e.g., 6) V-Adapters may overfit to disentangle the training data, resulting in poor generalization. To sum up, the fine-tuned image encoder and three V-adapters lead to a balance between adapting CLIP to the C-GQA dataset and disentangling its image features.

C Supplementary Case Studies

We use examples from MIT-States to analyze the disentanglement of primitive features learned by CAILA and our DCDA (e.g., DCDA[PRG]), especially those visual ones. Specifically, we first randomly sample a set of seen and unseen compositions from the test set of MIT-States whose annotated attributes or objects have high diversity. Here, we use the number of associated objects (resp. attributes) to roughly measure the diversity of an attribute (resp. object) as a wider range of objects (resp. attributes) would lead to more diverse appearances of attributes (resp. objects). For example, in Figure 2, attribute *broken* describes 40 objects in the training set ranging from *car*, *drum* to *furniture* with different damaged states; similar to the object *knife* in Figure 5, which is paired with 9 attributes in the training set. In addition, we also manually select 2 $\sim$ 3 attributes (resp. objects) whose associated objects (resp. attributes) are fewer but look greatly different, e.g., the attribute *worn* in Figure 2.

Then, for each sampled composition, we randomly extract at least 3 testing images and visualize their visually disentangled attribute and object representations learned by our DCDA and CAILA in Figure 2 and Figure 5, respectively, where different colors indicate different attribute or object labels. More specifically, for DCDA, we extract the features learned by our V-Adapters, i.e., $\boldsymbol{H}^v_{*\to A}$ and $\boldsymbol{H}^v_{*\to O}$; while for CAILA, we save the features learned by its attribute and object-specific vision encoding blocks.

From Figure 2 and Figure 5, it can be seen that the attribute embeddings or object embeddings learned by our model are clustered into different groups w.r.t different attributes or objects in each vector space. For example, in Figure 2, *broken car*, *broken drum* and *broken furniture* are in the same cluster in the attribute space, i.e., their learned attribute features are similar even though they show different *broken* state w.r.t different objects, similar to *curved knife* and *large knife* in Figure 5. Also, *broken car* is a seen composition, while *broken drum* and *broken furniture* are two unseen compositions. However, the attribute embeddings as well as object embeddings learned by CAILA scatter in the attribute and object space, respectively. All of these illustrate that our DCDA captured similar visual features specific to each primitive, which is discriminative and generalizable.

Moreover, we also find that the attribute and object embeddings of the same composition are divided into different clusters with different neighbors in two spaces, for example, *large knife* and *dull knife* are two neighbors from the cluster of *knife* in the object space, while they fall into the clusters of *large* and *dull* in the attribute space with neighbors *large building* and *dull brass*, respectively. This indicates that our method indeed disentangles the attribute and object features into different representation spaces.