HomoGraphAdapter: A Homogeneous Graph Neural Network as an Effective Adapter for Vision-Language Models

Chuan He[1,2] Zhuozhao Li[1†] Song Guo[3†] Xiaocheng Lu[3] Jinxiang Lai[3]
[1]Southern University of Science and Technology, Shenzhen, China
[2]The Hong Kong Polytechnic University, Hong Kong SAR, China
[3]Hong Kong University of Science and Technology (HKUST), Hong Kong SAR, China
chuan.he@connect.polyu.hk

Abstract

Vision-Language Models (VLMs), such as CLIP, have exhibited significant advancements in recognizing visual concepts through natural language guidance. However, adapting these models to downstream tasks remains challenging. Existing adaptation methods either overlook the structural knowledge between the text and image modalities or create overly complex graphs containing redundant information for alignment, leading to suboptimal classification performance and increased computational overhead. This paper proposes a novel adapter-tuning methodology named Homogeneous Graph Adapter (HomoGraphAdapter), which transforms diverse textual and visual descriptions into a unified set of node representations and establishes edges between nodes for inter-modal and cross-modal semantic alignment. We leverage a straightforward homogeneous Graph Neural Network (GNN) to adapt positive and negative classifiers across text and image modalities. The classifiers comprehensively enhance the performance for few-shot classification and OOD generalization. Compared with the SOTA approach HeGraphAdapter, HomoGraphAdapter improves classification accuracy by an average of 1.51% for 1-shot and 0.74% for 16-shot on 11 datasets, while also reducing both precomputation time and training time.

1 Introduction

Pre-trained Vision-Language Models (VLMs), especially CLIP (Radford et al., 2021) and its variants (Jia et al., 2021; Zhang et al., 2025), have opened a new chapter for various computer vision tasks. To efficiently adapt the CLIP model to downstream tasks, researchers typically employ one of two main strategies: prompt-tuning or adapter-tuning.

Prompt-tuning (Zhou et al., 2022b,a; Khattak et al., 2023) methods freeze CLIP's backbone encoders and introduce learnable vectors (referred to as soft prompts) that are fine-tuned using task-specific labeled data. For instance, CoOp (Zhou et al., 2022b) introduces a learnable prompt for the text encoder, while CoCoOp (Zhou et al., 2022a) designs a lightweight meta-network to generate text prompts for each image. While prompt-tuning methods are both parameter-efficient and computationally lightweight, they face several challenges, including overfitting to the learned prompts, high sensitivity to prompt initialization, and reduced robustness when encountering domain shifts. *Adapter-tuning* methods (Gao et al., 2024; Zhang et al., 2022; Huang et al., 2022) focus on fine-tuning output textual or visual features by introducing lightweight adapters as additional learnable components within the network. For instance, TaskRes (Huang et al., 2022) introduces residual vectors with textual features to learn task-specific classifiers. CLIP-Adapter (Gao et al., 2024) introduces an MLP-based adapter module that can be applied to textual or visual features. However, these adapters can still suffer from overfitting on small datasets and often lack the capacity to handle datasets with a large number of classes.

GraphAdapter (Li et al., 2024b) takes an initial step toward integrating knowledge graphs into CLIP-based models by using Graph Neural Networks (GNNs) to refine textual and visual features. It builds a separate knowledge subgraph for each modality and then applies two distinct GNNs to adjust the corresponding node features. However, since these two subgraphs do not interact, GraphAdapter overlooks crucial cross-modal structural information. HeGraphAdapter (Zhao et al., 2024) addresses this gap by introducing Heterogeneous Graph Learning for CLIP. It proposes a Heterogeneous Graph Adapter that fully exploits cross-modal information by building a single het-

†Corresponding authors.

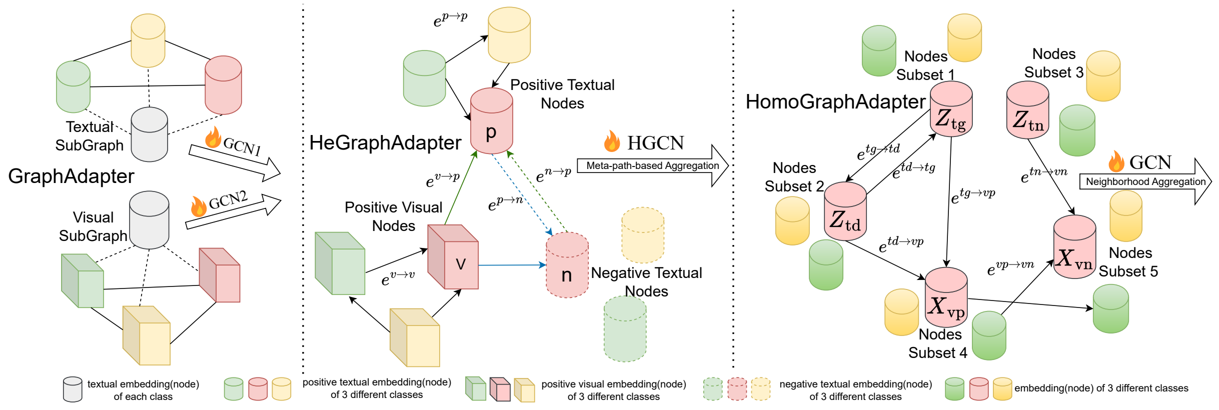

Figure 1: This figure illustrates the difference among the 3 existing graph-based adapter-tuning methods for CLIP. GraphAdapter constructs two separate subgraphs and adapts text features of CLIP with dual sub-graphs. HeGraphAdapter constructs a heterogeneous graph involving different nodes and edges with edge features through HGCN. Our HomoGraphAdapter constructs a unified homogeneous graph with only node features and edge connections by index. Different colors in the graph represent nodes corresponding to different classes.

erogeneous graph. While this approach preserves cross-modal dependencies, it may also introduce redundant information by creating edges between nodes from different classes and assigning potentially unnecessary edge attributes—factors that do not necessarily improve alignment between textual and visual embeddings. Therefore, existing graph-adapter methods either overlook cross-modal structural knowledge or adopt graph designs that become overly complex and include superfluous details.

To address these problems, we propose HomoGraphAdapter as an effective adapter for CLIP in data-limited scenarios. HomoGraphAdapter regards the textual or visual embedding of a specific class as a node in a homogeneous graph and builds edges only between nodes that exhibit semantic alignment. In this homogeneous graph, all nodes in this graph represent the same type, and all edges define a single type of relationship. Homogeneous graph learning not only significantly enhances feature alignment but also improves adaptation efficiency.

The contributions of this paper are summarized as follows:

- **HomoGraphAdapter** proposes *Homogeneous Graph Learning* for CLIP by encapsulating diverse textual or visual embeddings as nodes of the same type in a homogeneous graph and adjusting node features with graph's structure knowledge during few-shot adaptation.
- **HomoGraphAdapter** effectively transfers the knowledge of the CLIP model into positive and negative classifiers across dual modalities through homogeneous graph learning.
- **HomoGraphAdapter** achieves superior performance in terms of top-1 accuracy and demonstrates higher training efficiency compared to existing graph-based adapter-tuning methods.

2 Related Work

Textual classifier and Prompt-tuning methods for CLIP. The CLIP model (Radford et al., 2021) computes the cosine similarity between image embeddings and textual embeddings, assigning the predicted class to the text with the highest similarity score. In this setup, textual embeddings for all classes, generated by inputting positive prompts like "a photo of a ..." into CLIP's text encoder, can be interpreted as a positive textual classifier. Conversely, as suggested by prior studies (Wang et al., 2023; Tian et al., 2023; Nie et al., 2024; Li et al., 2024a), negative prompts such as "a photo of no ...", "a photo without ...", and "a photo not containing ..." yield a negative classifier, where the class corresponding to the text with the lowest similarity score is identified as the predicted class. In a similar vein, prompt-tuning methods (Zhou et al., 2022b,a; Khattak et al., 2023) enhance hard prompts by integrating a learnable vector as a soft prompt to CLIP's text encoder. These methods adapt the textual embeddings of CLIP to enhance alignment with visual embeddings, enabling more flexible and task-specific adaptations.

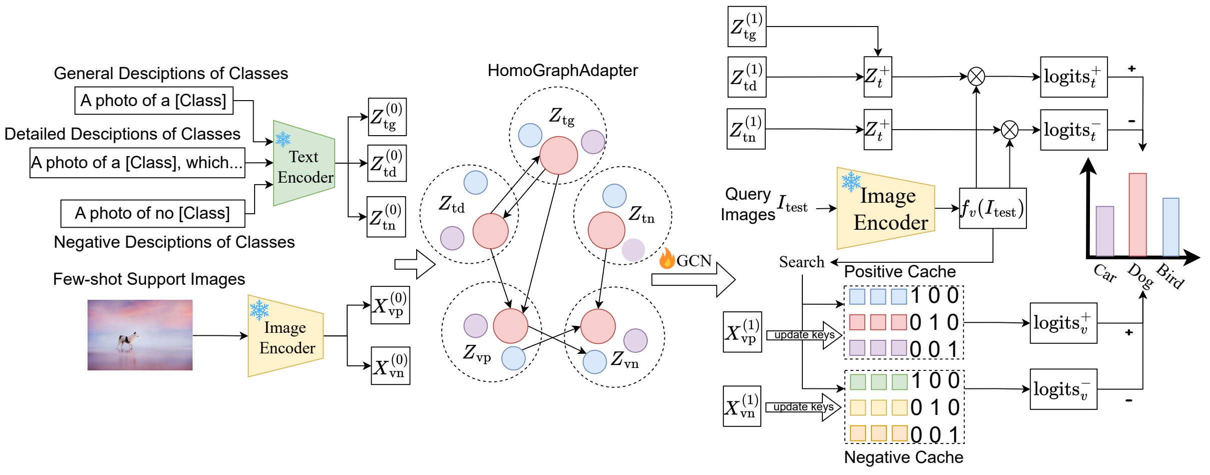

Figure 2: Overall framework of HomoGraphAdapter. First, multiple sets of textual and visual embeddings with diversified semantic meanings are generated with CLIP's two backbone encoders. Through the process of Homogeneous Graph Learning, they are adapted and form multiple classifiers in dual modality. The logits of the classifiers are calculated and then combined for the final label prediction.

Adapter-tuning methods and Visual classifier for CLIP. Most adapter-tuning approaches (Gao et al., 2024; Zhu et al., 2023; Udandarao et al., 2023; Tang et al., 2024) focus on developing lightweight adapters to adjust the image embeddings of CLIP. TaskRes (Huang et al., 2022) presents task-specific residual modules for adjusting visual embeddings. Meanwhile, CLIP-Adapter (Gao et al., 2024) introduces an MLP-based module designed to adapt visual features to textual features. Tip-Adapter (Zhang et al., 2022) is the first to complement the original textual classifier with an additional visual classifier, constructing a key-value (KV) cache of few-shot visual features as keys and their corresponding one-hot labels as values. The visual classifier assesses image-to-image similarities between the test image and representative images of each class. While the original Tip-Adapter operates as a training-free method, Tip-Adapter-F enhances this approach by adding an extra weight parameter for the keys in the cache model. However, the above methods generally struggle to capture cross-modal and intra-modal feature relationships and dependencies, limiting the understanding of few-shot data.

Graph Learning for CLIP: Graph learning captures the inherent structural relationships between textual and visual features by representing them within a graph. This integration of visual and language modalities facilitates the extraction of task-specific semantic knowledge, thereby enhancing cross-modal and intra-modal understanding. GraphAdapter (Li et al., 2024b) pioneers this direction by constructing separate knowledge subgraphs—one for each modality—and employing two distinct GNNs to adapt textual features. With the extensive study on heterogeneous graph learning in recommendation systems (Ying et al., 2018) and computer vision (Cao et al., 2022), HeGraphAdapter (Zhao et al., 2024) is the first to introduce it for the few-shot adaptation of CLIP. It formalizes the structural knowledge among various types of nodes and their relationships within a unified graph.

However, the core challenge of graph learning for the few-shot adaptation of CLIP lies in constructing a graph that captures the structural information most essential for feature alignment and classification. This involves carefully integrating node features and edge connections to reflect the task-relevant relationships. While heterogeneous graphs excel at modeling complex relationships among various entities, CLIP inherently deals with only two modalities—text and image—where the potential for highly intricate interrelations is limited. Consequently, HeGraphAdapter may be less than optimal in scenarios emphasizing task-specific learning efficiency with limited data.

3 Methodology

In this section, we introduce the methodology of HomoGraphAdapter, which establishes and learns class-specific structural knowledge in a unified homogeneous graph. Figure 1 contrasts our approach with two existing graph adapter methods. The key advantage of HomoGraphAdapter is its ability to leverage diversified feature representations from natural language and few-shot images, effectively learning discriminative classifiers using a unified homogeneous GNN. Unlike HeGraphAdapter (Zhao et al., 2024), which pre-computes edge features between every pair of neighboring nodes, HomoGraphAdapter only retains edge connections (i.e., index pairs of nodes). By eliminating redundant edge information, HomoGraphAdapter effectively encodes graph structure to learn higher-quality classifiers for downstream classification tasks.

3.1 Overall Framework

The entire framework of HomoGraphAdapter is illustrated in Figure 2, encompassing feature generation, cache construction, homogeneous graph learning, logit calculation, and label prediction. In the forward process of HomoGraphAdapter, we start by encoding three sets of textual descriptions—general, detailed, and negative—to obtain corresponding textual embeddings. Few-shot support images are then used to generate positive and negative visual embeddings. These textual and visual embeddings form the nodes of a homogeneous graph, which is subsequently processed by a one-layer homogeneous GCN to adapt node features and update the classifiers. Finally, for each test image, we compute logits by evaluating image-text and image-image similarities and combine them to produce the final label prediction.

3.2 Homogeneous Graph Data

To extract structural knowledge, we construct a homogeneous graph $G = (\mathcal{V}, \mathcal{E})$. In this representation, each node feature corresponds to the textual or visual embedding of a specific class. Edge connections represent the structural knowledge that facilitates both intra-modal and cross-modal feature alignment and classification.

Subsets of Nodes. To illustrate the graph learning process, we divide $\mathcal{V}$ into five subsets $\{Z_{\text{tg}}, Z_{\text{td}}, Z_{\text{tn}}, X_{\text{vp}}, X_{\text{vn}}\}$, as shown in Figure 3. Each subset of nodes is a set of representations that describe C different categories, where C is the category number in the downstream task.

Nodes Z_{tg} from positive general text descriptions of categories. CLIP can make zero-shot predictions using a general template such as 'A photo of a [category]'. For a C-way downstream

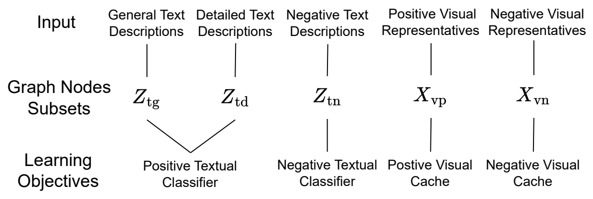
Input
General Text Descriptions
Detailed Text Descriptions
Negative Text Descriptions
Positive Visual Representatives
Negative Visual Representatives
Graph Nodes Subsets
Z_{tg}
Z_{td}
Z_{tn}
X_{vp}
X_{vn}
Learning Objectives
Positive Textual Classifier
Negative Textual Classifier
Postive Visual Cache
Negative Visual Cache

Figure 3: This figure briefly illustrates the flow of the proposed **HomoGraphAdapter**. Multiple text descriptions and images are transformed into subsets of node features, which are then adapted to create more effective classifiers.

dataset, we insert each of the C category names into this template to form C textual inputs, which are then tokenized and encoded by CLIP's text encoder f_t. The resulting embeddings represent the initial $Z_{\text{tg}}^{(0)}$ in the graph.

$$\begin{aligned} Z_{\text{tg}}^{(0)} &= \{z_k\}_{k=1}^{C}, \text{where} \\ z_k &= f_t(\text{'A photo of a [category]}_k\text{'}) \end{aligned}$$

Nodes Z_{td} from positive detailed text descriptions of categories. Previous studies (Goswami et al., 2024; Roy and Etemad, 2023) suggest that utilizing a pre-trained Large Language Model (LLM) such as GPT (Floridi and Chiriatti, 2020) to generate lengthy, detailed text descriptions for each class can get more discriminative textual classifiers. However, lengthy text may not always reflect the exact visual concepts in the corresponding images. Our approach balances this by integrating general and detailed textual descriptions through a unified graph adapter, thereby producing a more robust classifier. Similarly, the output embeddings correspond to the initial $Z_{\text{td}}^{(0)}$ in the graph.

$$\begin{aligned} Z_{\text{td}}^{(0)} &= \{z_k\}_{k=1}^{C}, \text{ where} \\ z_k &= f_t\,(\text{'A photo of a [category]}_k\text{, which ...'}) \end{aligned}$$

Nodes Z_{tn} from negative text descriptions of categories. Although CLIP does not natively support negative learning (Radenovic et al., 2023), recent work (Zhao et al., 2024) demonstrates that incorporating a negative textual classifier can enhance performance. Accordingly, HomoGraphAdapter learns such a classifier through homogeneous graph learning. Specifically, we generate negative text prompts for each class (e.g., 'A photo of no [category]') and encode them using CLIP's text encoder. The resulting embeddings represent the initial $Z_{\text{tn}}^{(0)}$ in the graph.

$$\begin{aligned} Z_{\text{tn}}^{(0)} &= \{z_k\}_{k=1}^{C}, \text{ where} \\ z_k &= f_t(\text{`A photo of no [category]}_k\text{'}) \end{aligned}$$

Nodes X_{vp} from positive visual representatives. As introduced by Tip-Adapter (Zhang et al., 2022), we maintain a visual cache where representative images act as keys and their corresponding one-hot labels serve as values. The visual cache measures image-image similarities between test image features x_{test} and keys X_{key} using an affinity function:

$$A = \exp\left(-\beta\left(1 - x_{\text{test}} X_{\text{key}}^{\top}\right)\right).$$

Similarly, we precompute a positive visual cache X_{cache}^{+}. In the initial cache, the keys are the C-way K-shot image features, and the values are the one-hot labels $\mathcal{L}$ of image features correspondingly:

$$X_{\text{key}}^{+} = \begin{bmatrix} f_t(x_1^1) & f_t(x_1^2) & \cdots & f_t(x_1^K) \\ f_t(v_2^1) & f_t(v_2^2) & \cdots & f_t(v_2^K) \\ \vdots & \vdots & \ddots & \vdots \\ f_t(v_C^1) & f_t(v_C^2) & \cdots & f_t(v_C^K) \end{bmatrix}$$

$$\mathcal{L} = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

Nodes $X_{\text{vp}}^{(0)}$ are initially set to the mean of X_{key}^{+} for each class. The updated node features $X_{\text{vp}}^{(1)}$ will be added to X_{key}^{+}.

Nodes X_{vn} from negative visual representives. As suggested in previous work (Wang et al., 2023; Sun et al., 2022; Huang et al., 2024; Kim et al., 2019), negative learning can enhance classifier performance. In the visual domain, we also maintain a negative cache to enhance the classification performance of CLIP. In the negative cache, the higher the similarity to the keys, the lower the probability that the image is classified to that class. Similarly, we precompute a negative visual representation for each class.

Specifically, for each class, we select top k dissimilar classes (based on cosine similarity) from the remaining $(C-1)$ classes. The average of the representative features from the k classes generates the negative visual representation for a specific class. All the visual representations serve as keys of the negative visual cache $X_{\text{key}}^{-} \in \mathbb{R}^{CK \times d}$. The values of the negative visual cache are the one-hot labels $\mathcal{L} \in \mathbb{R}^{CK \times C}$ of image features correspondingly. Similarly, the negative visual nodes $X_{\text{vn}}^{(0)}$ are initialized using the mean of X_{key}^{-} for each class. The learned node features are then added to update the keys in the negative cache.

Edge Connections. To align nodes and learn more discriminative classifiers, we establish directed edges among different node subsets, allowing feature adaptation across the graph. We define six subsets of edge indices $\mathcal{E}^{\text{tg}\to\text{td}}, \mathcal{E}^{\text{td}\to\text{tg}}, \mathcal{E}^{\text{tg}\to\text{vp}}, \mathcal{E}^{\text{td}\to\text{vp}}, \mathcal{E}^{\text{tn}\to\text{vn}}, \mathcal{E}^{\text{vp}\to\text{vn}}$. Each provides connections from a source subset to a target subset, as shown in Figure 1. The first five subsets of edges contain C-way intra-class connections from the source node subset to the target node subset, linking nodes that belong to the same class. The final subset of edges contains kC-way inter-class connections, linking nodes that belong to different classes.

$\mathcal{E}^{\text{tg}\to\text{td}}$ and $\mathcal{E}^{\text{td}\to\text{tg}}$ connect positive textual nodes Z_{tg} and Z_{td} to learn a more robust positive classifier than the original ones. $\mathcal{E}^{tg\to vp}$ and $\mathcal{E}^{td\to vp}$ denote the cross-modal linking edges from text to image. Due to the limited availability of image data, the initial visual features generated may lack sufficient representativeness. These edges aim to leverage natural language guidance to adjust the visual features. Similarly, $\mathcal{E}^{tn\to vn}$ denotes the cross-modal edges from the negative textual subset Z_{tg} to the negative visual subset X_{vn}.

$\mathcal{E}^{vp\to vn}$ denotes the cross-linking edges from one class positive visual nodes X_{vp} to another class in negative visual nodes X_{vn} to learn better negative visual representatives. As mentioned in §3.2, we select the top k dissimilar classes for each class. Here, we connect the top k classes from X_{vp} to the corresponding class in X_{vn}. The hyperparameter k is positively correlated with the number of classes C in the dataset.

3.3 Homogeneous Graph Learning

We have constructed a homogeneous graph where both the initial node features and their edge connections are encapsulated. In this subsection, we apply a one-layer homogeneous Graph Convolutional Network(GCN) (Kipf and Welling, 2016) to refine the node features and adapt the classifiers for the downstream task. During graph learning, we first add self-loops to the adjacency matrix and then perform neighborhood aggregation.

$$x_i^{(1)} = \sum_{j \in N(i) \cup \{j\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(W^T \cdot x_j^{(0)} \right) + b$$

Given any single node i with node feature $x_i^{(0)}$ from the whole node set $\mathcal{V}$, its neighboring node features are denoted as $x_j^{(0)}$ where $j \in N(i) \cup \{j\}$. The GCN network aggregates messages from neighboring nodes by first transforming each neighbor's feature with a weight matrix W, normalized by the degree of the nodes , and then summing the results. A bias vector b is added to the aggregated output. Notably, we only apply the bias vector on nodes Z_t^- and disable the bias term on other nodes.

3.4 Adapted Classifiers and Logits

After graph learning, node features are updated into $Z_{\text{tg}}^{(1)}, Z_{\text{td}}^{(1)}, Z_{\text{tn}}^{(1)}, X_{\text{vp}}^{(1)}, X_{\text{vn}}^{(1)}$. This section introduces how multiple classifiers are formed and how logits are calculated and combined.

Positive Textual Classifier Z_t^+. Z_t^+ is the weighted combination of positive classifiers derived from the long and short text descriptions of classes. For Z_t^+, we fuse the updated node features with the original node features via the weighted sum fusion strategy. In this way, the original features are modulated for classification in a residual way. Given some test images I_{test}, we calculate Z_t^+ and corresponding logits:

$$\begin{aligned} Z_{\text{tg}} &= (1-a) \cdot Z_{\text{tg}}^{(0)} + a \cdot Z_{\text{tg}}^{(1)} \\ Z_{\text{td}} &= (1-b) \cdot Z_{\text{td}}^{(0)} + b \cdot Z_{\text{td}}^{(1)} \\ Z_t^+ &= \text{Normalize}\left(c \cdot Z_{\text{tg}} + (1-c) \cdot Z_{\text{td}}\right) \\ \text{logits}_t^+ &= 100 \cdot f_v(I_{\text{test}}) \cdot {Z_t^+}^\top \end{aligned} \tag{1}$$

Here, a, b, and c represent the three hyperparameters used as fusion weights. In experiments, a and b are set to a value between 0.0 and 0.2, while c is set to 0.45.

Negative Textual Classifier Z_t^-. Z_t^- is derived from the negative text descriptions. Because the model learns a bias vector for these nodes, we skip the residual fusion strategy to avoid interference from the original features. Instead, we normalize the updated embeddings and compute logits by measuring how dissimilar they are to the visual features $f_v(I_{\text{test}})$.

$$\begin{aligned} Z_t^- &= \text{Normalize}(Z_{\text{tn}}^{(1)}) \\ \text{logits}_t^- &= 100 \cdot (1 - f_v(I_{\text{test}}) \cdot {Z_t^-}^\top) \end{aligned} \tag{2}$$

Positive Visual Cache X_{cache}^+. For X_{cache}^+, the keys are enhanced by incorporating the learned positive visual features as bias. Given images I_{test} to be classified, we calculate its image-image affinities and obtain the logits:

$$\begin{aligned} X_{\text{key}}^+ &\leftarrow \text{Normalize}(X_{\text{key}}^+ + X_{\text{vp}}^{(1)}), \\ \text{logits}_v^+ &= A(f_v(I_{\text{test}}) \cdot {X_{\text{key}}^+}^T) \cdot \mathcal{L}, \quad \text{where} \\ A(x) &= \exp\left(-\beta\left(1-x\right)\right). \end{aligned} \tag{3}$$

Negative Visual Cache X_{cache}^-. Similarly, for X_{cache}^-, we update the keys by incorporating the learned negative visual features. Analogous to Equation 3, we compute the negative affinities to obtain the classification logits:

$$\begin{aligned} X_{\text{key}}^- &\leftarrow \text{Normalize}(X_{\text{key}}^- + X_{\text{vn}}^{(1)}), \\ \text{logits}_v^- &= A'\left(1 - f_v(I_{\text{test}}) \cdot {X_{\text{key}}^-}^\top\right) \cdot \mathcal{L}, \quad \text{where} \\ A'(x) &= \exp\left(-\beta'\left(1-x\right)\right). \end{aligned} \tag{4}$$

The final logits for label prediction are the weighted combination of the above 4 logits:

$$\begin{aligned} \text{logits}_f =& \theta_1 \cdot \text{logits}_t^+ + \theta_2 \cdot \text{logits}_t^- + \theta_3 \cdot \text{logits}_v^+ + \\ & \theta_4 \cdot \text{logits}_v^- \end{aligned} \tag{5}$$

where $\theta_1, \theta_2, \theta_3, \theta_4$, β and β' are hyper-parameters. In experiments, θ_1 is set to 1, and θ_2 is set to a value between 0 and 0.5. The initial values for θ_3, θ_4, β and β' are set to certain values during training. During testing, the optimal values are determined through a search for the best results.

4 Experiments

In this section, we evaluate HomoGraphAdapter on few-shot classification tasks across 11 benchmark datasets, including ImageNet (Deng et al., 2009), StandfordCars (Krause et al., 2013), UCF101 (Soomro, 2012), Caltech101 (Fei-Fei et al., 2004), Flowers102 (Nilsback and Zisserman, 2008), SUN397 (Xiao et al., 2010), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), FGVCAircraft (Maji et al., 2013), OxfordPets (Parkhi et al., 2012), and Food101 (Bossard et al., 2014). These datasets include vision tasks such as remote sensing classification, action recognition, texture classification, and fine-grained classification. We compare HomoGraphAdapter with two state-of-the-art graph adapter methods: GraphAdapter (Li et al., 2024b) and HeGraphAdapter (Zhao et al., 2024), as well as three typical adapter methods

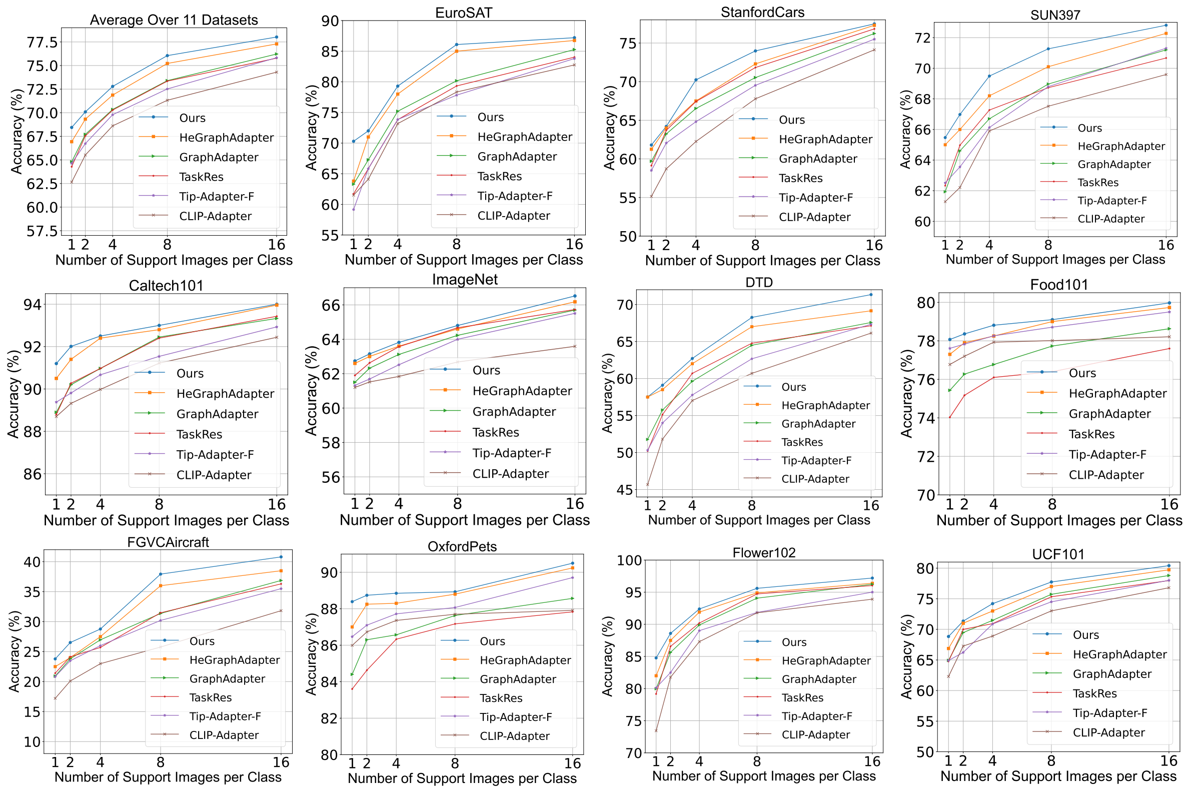

Figure 4: Performance illustration of HomoGraphAdapter with comparison of the state-of-the-art methods on the 1/2/4/8/16-shot adaptation for 11 image classification benchmark datasets.

for CLIP: CLIP-Adapter (Gao et al., 2024), Tip-Adapter (Zhang et al., 2022), and TaskRes (Yu et al., 2023).

Implementation Details. ResNet-50 (He et al., 2016) is used as the default backbone of the CLIP model. We fine-tune HomoGraphAdapter with few-shot labeled data sampled from the downstream dataset. We use AdamW optimizer (Kingma, 2014) with a cosine scheduler. The training epochs are set to 20 for most datasets, with the following exceptions: EuroSAT is set to 100 epochs, FGVCAircraft to 30 epochs, and OxfordFlowers to 70 epochs. For building the visual cache and training, the batch size is set to 256; for testing, it is set to 64. The learning rate is set to 5×10^{-4}. Because tasks differ in complexity, certain hyperparameters vary across datasets. Further details on the hyperparameters are provided in the Appendix. All experiments are conducted on a single NVIDIA A100 GPU.

4.1 Few-shot Classification

For few-shot classification, the model is trained using sampled few-shot data from the training set and is directly tested on the testing set of the same dataset. We conduct experiments with 1-shot, 2-shot, 4-shot, 8-shot, and 16-shot settings, and the comparison metric is the Top-1 classification accuracy. The performance on all few-shot settings is illustrated in Figure 4. The 16-shot performance results on the 11 datasets are displayed in Table 1.

Overall, HomoGraphAdapter outperforms all baselines in average accuracy across 11 datasets for every few-shot settings. The results outperform the second-best method by an average of 1.51% in the 1-shot setting and 0.74% in the 16-shot setting. Notably, the performance improvement is more significant in the 1-shot settings for EuroSAT and OxfordPets, while for FGVCAircraft and DTD, it is more pronounced in the 16-shot setting. Moreover, for datasets with many classes, HomoGraphAdapter achieves 66.58% on ImageNet (1,000 classes) and 72.81% on SUN397 (397 classes). On datasets with a few classes, HomoGraphAdapter demonstrates leading performance with 87.20% on EuroSAT (10 classes).

4.2 Efficiency Comparison

In Table 2, we report the pre-computation time, training time, epochs, and the number of trainable parameters for HomoGraphAdapter on the ImageNet dataset in a 16-shot setting. With just 2.07 million trainable parameters, HomoGraphAdapter

Table 1: Performance in top-1 accuracy of different methods on 11 image classification datasets under the 16-shot setting.

Method	Caltech101	DTD	EuroSAT	FGVCAircraft	Flowers102	Food101	ImageNet	OxfordPets	StanfordCars	SUN397	UCF101	Average
Zero-shot CLIP	84.52	40.33	41.80	16.98	65.46	77.31	60.33	85.51	54.26	58.56	61.44	58.77
CoOp	91.61	63.11	82.36	31.01	94.39	73.80	62.95	87.30	72.51	69.11	75.70	73.07
CoCoOp	90.90	57.53	70.77	22.40	79.14	79.68	62.71	89.93	62.22	67.21	70.81	68.48
CLIP-Adapter	92.44	66.14	82.76	31.83	93.91	78.21	63.59	87.91	74.12	69.59	76.80	74.30
Tip-Adapter-F	92.93	67.33	83.80	35.50	95.01	79.50	65.51	89.71	75.50	71.31	78.01	75.83
TaskRes	93.43	67.13	84.03	36.30	96.03	77.60	65.73	87.83	76.83	70.67	77.97	75.78
GraphAdapter	93.37	67.90	85.55	36.93	96.13	78.67	65.72	88.59	76.20	71.30	78.67	76.25
HeGraphAdapter	93.96	69.15	86.75	38.49	96.39	79.73	66.18	90.24	77.30	72.28	79.73	77.30
HomoGraphAdapter	**94.20**	**71.34**	**87.20**	**40.80**	**97.20**	**79.97**	**66.58**	**90.50**	**77.48**	**72.81**	**80.41**	**78.04**
(Ours)	(+0.24)	(+2.19)	(+0.45)	(+2.31)	(+0.81)	(+0.24)	(+0.40)	(+0.26)	(+0.18)	(+0.53)	(+0.68)	(+0.74)

Table 2: Efficiency comparison with other methods on 16-shot ImageNet. The metrics include training time, number of training epochs, and number of trainable parameters.

Method	**Pre-Computation Time**	**Training (One Epoch)**	**Training Epochs**	**Params (M)**	**Accuracy (%)**
CLIP	-	-	-	-	60.33
CLIP-Adapter	-	15.1 sec	200	**0.52**	63.59
Tip-Adapter-F	**2.9 min**	**12.3 sec**	**20**	16.38	65.51
GraphAdapter	6.7 min	15.9 sec	**20**	4.15	65.70
HeGraphAdapter	16 min	14.1 sec	30	10.37	66.18
HomoGraphAdapter	3.2 min	12.6 sec	**20**	2.07	**66.58**

ranks as the second smallest among adapter-tuning methods. HomoGraphAdapter requires only 12.6 seconds to train per epoch, totaling 20 epochs, making it the second fastest among methods involving precomputation. Among graph-based methods, it achieves the shortest pre-computation time. GraphAdapter and HeGraphAdapter both require pairwise cosine similarity calculations to generate edge features, whereas HomoGraphAdapter relies only on edge indices without edge features. Consequently, HomoGraphAdapter only needs to compute positive and negative visual caches, reducing pre-computation overhead.

Overall, compared with existing graph-based methods GraphAdapter and HeGraphAdapter, HomoGraphAdapter requires less pre-computation time and training time while achieving better classification accuracy, highlighting its effectiveness and efficiency.

4.3 Ablation Studies

In this section, we perform ablation studies to evaluate the effectiveness of the four classifiers used in HomoGraphAdapter. We assess the 16-shot performance across four variations of HomoGraphAdapter on ImageNet. The graph nodes are comprised of five distinct subsets of nodes: $Z_{\text{tg}}, Z_{\text{td}}, Z_{\text{tn}}, X_{\text{vp}}, X_{\text{vn}}$. In the first variant $Z_{\text{tg}}Z_{\text{td}}$, we remove the remaining nodes $Z_{\text{tn}}X_{\text{vp}}X_{\text{vn}}$ and their corresponding edges. Other variants are implemented in a similar fashion. As shown in Table 3, each subset of nodes and classifiers contributes positively to the final performance, confirming the importance of all components in HomoGraphAdapter.

Table 3: Ablation studies on the graph nodes of HomoGraphAdapter. We implement four variants and report the performances of the 16-shot adaptation for ImageNet.

Graph Nodes	Textual Classifiers		Visual Caches		16-shot Acc
	Positive	Negative	Positive	Negative	
$Z_{\text{tg}}Z_{\text{td}}$	✓				64.11
$Z_{\text{tg}}Z_{\text{td}}X_{\text{vp}}$	✓		✓		65.68
$Z_{\text{tg}}Z_{\text{td}}Z_{\text{tn}}X_{\text{vp}}$	✓	✓	✓		66.47
$Z_{\text{tg}}Z_{\text{td}}Z_{\text{tn}}X_{\text{vp}}X_{\text{vn}}$	✓	✓	✓	✓	66.58

Additional ablation studies are detailed in the Appendix.

5 Conclusions

This paper proposes a homogeneous graph learning approach to tune CLIP in the data-limited conditions. We introduce a unified, single-layer homogeneous GNN that encapsulates diverse natural language and visual embeddings within a homogeneous graph comprising only one node type and one edge type. This design jointly enhances positive learning and negative learning across both modalities. Extensive experiments validate the effectiveness and efficiency of our method. In the future, we plan to explore how graph learning can be extended to fine-tune other pre-trained multimodality models.

Limitations

Two key limitations are identified in HomoGraphAdapter:

Performance Limitations with CLIP: We conduct all experiments using the pre-trained CLIP model, which inherently constrains the potential performance enhancement from negative classifiers. This limitation stems from the fact that the CLIP model was trained without negative descriptions, resulting in its inability to effectively differentiate between relevant and irrelevant inputs. As a consequence, the absence of negative examples during the training phase may significantly impede the model's overall capacity to enhance its classification accuracy and robustness in real-world applications. This could lead to suboptimal performance when the model encounters ambiguous or similar data points that require a clear distinction.

Task-Specific Adaptation Challenges: Similar to most adapter-tuning methods, the model fine-tuned by HomoGraphAdapter on a specific downstream task cannot be directly applied to another task without undergoing additional adaptation. This limitation highlights the necessity for tailored tuning processes that cater to the unique characteristics of each task, which can be both resource-intensive and time-consuming. Such processes often require substantial computational resources and expert intervention, reducing the efficiency of deploying models across varying tasks.

Looking ahead, future efforts should focus on fully exploring the potential of integrating negative learning with positive learning. This approach could yield more nuanced representations and significantly improve model performance across a wider array of tasks. By incorporating negative examples into the training process, we can enhance the model's ability to distinguish between similar classes, thereby refining its decision-making capabilities and overall effectiveness.

Furthermore, improving the robustness of adapter-tuning approaches is essential for ensuring that models consistently perform well under diverse conditions, such as noisy data or shifts in domain. This could involve the development of more versatile adapter architectures or the implementation of techniques that facilitate rapid adaptation to new tasks. Ultimately, such advancements would increase the efficiency and applicability of these models in real-world scenarios, making them more versatile and reliable in various applications.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62202216), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2023A1515010244), the Shenzhen Science and Technology Program (Grant No. 20231121101752002), and the Center for Computational Science and Engineering at Southern University of Science and Technology. This research was also supported by the Hong Kong RGC General Research Fund (152244/21E, 152169/22E, 152228/23E, 162161/24E), the Research Impact Fund (No. R5011-23F, No. R5060-19), the Collaborative Research Fund (No. C1042-23GF), the NSFC/RGC Collaborative Research Scheme (No. CRS_HKUST602/24), the Areas of Excellence Scheme (No. AoE/E-601/22-R), and InnoHK (HKGAI).

References

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer.

Pingping Cao, Zeqi Zhu, Ziyuan Wang, Yanping Zhu, and Qiang Niu. 2022. Applications of graph convolutional networks in computer vision. *Neural computing and applications*, 34(16):13387–13405.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. 2014. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. 2024. Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, 132(2):581–595.

Koustava Goswami, Srikrishna Karanam, Prateksha Udhayanan, KJ Joseph, and Balaji Vasan Srinivasan. 2024. Copl: Contextual prompt learning for vision-language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18090–18098.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226.

Shiyuan Huang, Jiawei Ma, Guangxing Han, and Shih-Fu Chang. 2022. Task-adaptive negative envision for few-shot open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7171–7180.

Zhenyu Huang, Mouxing Yang, Xinyan Xiao, Peng Hu, and Xi Peng. 2024. Noise-robust vision-language pre-training with positive-negative learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR.

Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. 2023. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122.

Youngdong Kim, Junho Yim, Juseung Yun, and Junmo Kim. 2019. Nlnl: Negative learning for noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 101–110.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561.

Tianqi Li, Guansong Pang, Xiao Bai, Wenjun Miao, and Jin Zheng. 2024a. Learning transferable negative prompts for out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17584–17594.

Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. 2024b. Graphadapter: Tuning vision-language models with dual knowledge graph. *Advances in Neural Information Processing Systems*, 36.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Jun Nie, Yonggang Zhang, Zhen Fang, Tongliang Liu, Bo Han, and Xinmei Tian. 2024. Out-of-distribution detection with negative prompts. In *The Twelfth International Conference on Learning Representations*.

Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.

Filip Radenovic, Abhimanyu Dubey, Abhishek Kadian, Todor Mihaylov, Simon Vandenhende, Yash Patel, Yi Wen, Vignesh Ramanathan, and Dhruv Mahajan. 2023. Filtering, distillation, and hard negatives for vision-language pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6967–6977.

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.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR.

Shuvendu Roy and Ali Etemad. 2023. Consistency-guided prompt learning for vision-language models. *arXiv preprint arXiv:2306.01195*.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

K Soomro. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Ximeng Sun, Ping Hu, and Kate Saenko. 2022. Dualcoop: Fast adaptation to multi-label recognition with limited annotations. *Advances in Neural Information Processing Systems*, 35:30569–30582.

Yuwei Tang, Zhenyi Lin, Qilong Wang, Pengfei Zhu, and Qinghua Hu. 2024. Amu-tuning: Effective logit bias for clip-based few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23323–23333.

Yumin Tian, Yuanbo Li, Di Wang, Xiao Liang, Ronghua Zhang, and Bo Wan. 2023. Enhancing clip-based text-person retrieval by leveraging negative samples. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 271–283. Springer.

Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. 2023. Sus-x: Training-free name-only transfer of vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2725–2736.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. 2019. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32.

Hualiang Wang, Yi Li, Huifeng Yao, and Xiaomeng Li. 2023. Clipn for zero-shot ood detection: Teaching clip to say no. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1802–1812.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.

Tao Yu, Zhihe Lu, Xin Jin, Zhibo Chen, and Xinchao Wang. 2023. Task residual for tuning vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10899–10909.

Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. 2025. Long-clip: Unlocking the long-text capability of clip. In *European Conference on Computer Vision*, pages 310–325. Springer.

Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. 2022. Tip-adapter: Training-free adaption of clip for few-shot classification. In *European conference on computer vision*, pages 493–510. Springer.

Yumiao Zhao, Bo Jiang, Xiao Wang, Qin Xu, and Jin Tang. 2024. Hegraphadapter: Tuning multi-modal vision-language models with heterogeneous graph adapter. *arXiv preprint arXiv:2410.07854*.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022a. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022b. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348.

Xiangyang Zhu, Renrui Zhang, Bowei He, Aojun Zhou, Dong Wang, Bin Zhao, and Peng Gao. 2023. Not all features matter: Enhancing few-shot clip with adaptive prior refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2605–2615.

HomoGraphAdapter: A Homogeneous Graph Neural Network as an Effective Adapter for Vision-Language Models

Supplementary Material

In the appendix, we provide additional implementation details and experiment results of our HomoGraphAdapter.

A Hyperparameter Values and Sensitivity Analysis

Table A.4 summarizes the values of all the hyperparameters—k, a, b, c, θ_2, θ_3, θ_4, β and β' for each dataset. We also present a sensitivity analysis of these hyperparameters. As shown in Figure A.5, we adjust one hyperparameter while keeping the others constant. The model's performance initially increases with higher hyperparameter values, but eventually declines after reaching a peak. The results indicate that all parameters contribute effectively. Moreover, HomoGraphAdapter maintains stable performance when adjusting individual hyperparameters, as it incorporates multiple classifiers from both textual and visual modalities. For example, when adjusting parameter k from 16 to 128, the performance increases from 66.48 to 66.58. When k is increased from 128 to 512, the performance drops from 66.58 to 66.53. The influence is limited because parameter k only affects the negative visual classifier.

B Experiment Results Across Different Backbones

We also implement HomoGraphAdapter and conduct experiments under various CLIP encoder backbones, including ResNet-101 (He et al., 2016), ViT-B/32 (Dosovitskiy, 2020), and ViT-B/16 (Dosovitskiy, 2020). Specifically, we trained our model on the 16-shot ImageNet dataset and assessed its performance on ImageNet and Out-Of-Distribution (OOD) ImageNet datasets, including ImageNet-V2 (Recht et al., 2019) and ImageNet-Sketch (Wang et al., 2019).

As shown in Table B.5, HomoGraphAdapter consistently achieves superior performance compared with the baselines across all backbones. Compared with the second-best method HeGraphAdapter, HomoGraphAdapter attains an average performance gain of 1.2% on Source and 1.3% on Target across 4 different backbones. The results indicate that HomoGraphAdapter consistently improves classification accuracy for the downstream task across different CLIP backbones and varying distribution conditions.

C Visualizations of HomoGraphAdapter

In this section, we present visualizations of the positive and negative textual and visual classifiers in HomoGraphAdapter to validate our findings and offer a clearer perspective on our research. For the EuroSAT dataset, t-SNE visualization of positive and negative visual representatives in cache models is presented in Figure C.6. For the Food101 dataset, the Grad-CAM visualization of learned positive and negative textual embeddings for the ground-truth class is demonstrated in Figure C.7.

D Details of Multiple Text Prompts

The use of multiple text prompts, including both positive and negative ones, has gained significant traction in the community for prompt-tuning and adapter-tuning methods. To save on pre-computation time, we do not generate the sentences on the fly. For positive general text descriptions and negative text descriptions of categories mentioned in §3 of the main text, we utilize multiple pre-defined prompt templates to generate them. The pre-defined prompt templates for each dataset are shown in Table D.6. We fill these templates with the class names from the dataset to generate text prompts, which are then used as input to the text encoder, and we average their embeddings.

For the positive detailed descriptions mentioned in §3 of the main text, we directly use the JSON files for each category in the dataset, generated by GPT-3 in (Roy and Etemad, 2023). Due to their length, we showcase the generated sentences for one class in Eurosat in Table D.7.

Table A.4: Values of hyperparameters that achieve peak performance during fine-tuning on the 11 datasets.

Dataset	Caltech	FGVC	Eurosat	DTD	Cars	Flowers102	Pets	UCF101	Food101	SUN397	ImageNet
k	12	16	2	10	16	4	8	4	24	64	128
a	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.05
b	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.05
c	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
θ_1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
θ_2	0.1	0.3	0.3	0.15	0.25	0.25	0.2	0.25	0.25	0.1	0.25
θ_3	1.0	4.5	1.5	1.8	2.0	1.5	1.5	1.5	0.8	1.5	1.2
β	1.5	2.5	1.0	2.8	2.5	2	1.5	1.5	0.8	1.5	1.8
θ_4	0.3	0.6	0.1	0.3	0.6	0.5	0.25	0.3	0.25	0.3	0.3
β'	1.0	1.5	0.4	1.0	2.0	1.5	0.8	1.0	0.8	1.0	1.8

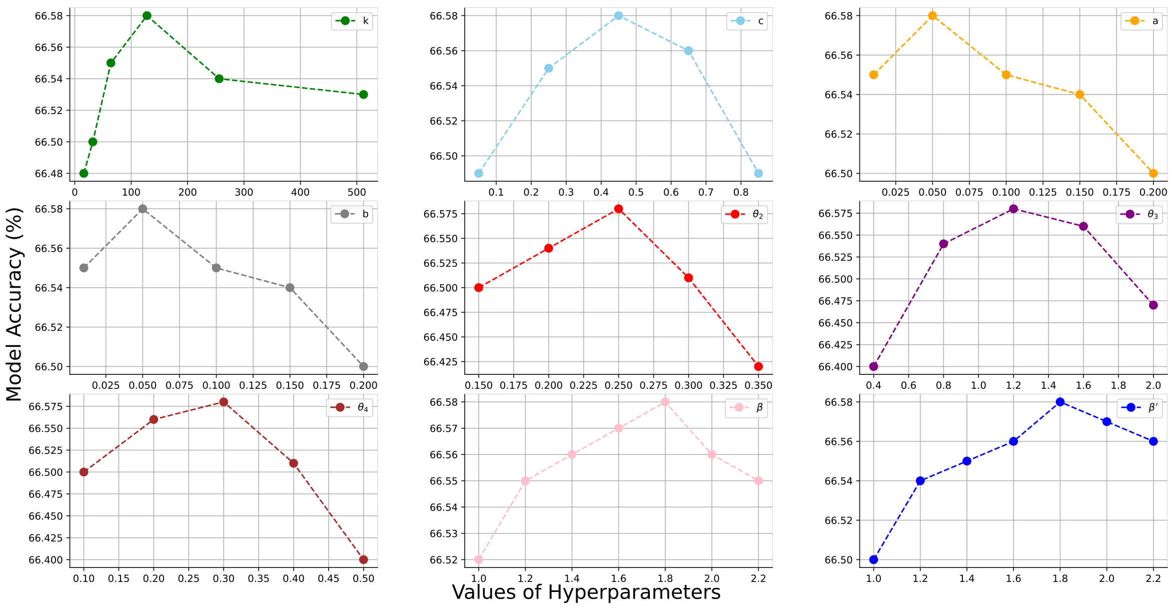

Figure A.5: Ablation studies on the sensitivity of hyperparameters are conducted while fine-tuning on the 16-shot ImageNet dataset. Each individual hyperparameter is adjusted while keeping the others constant to observe changes in accuracy.

Figure C.6: t-SNE visualizations of positive and negative visual representatives in cache models of HomoGraphAdapter. Dots in different colors represent embeddings of different categories. From top left to right, distributions indicate the variation of keys in the positive cache during fine-tuning on EuroSAT dataset. From bottom left to right, distributions indicate the variation of keys in the negative visual cache during fine-tuning on EuroSAT dataset.

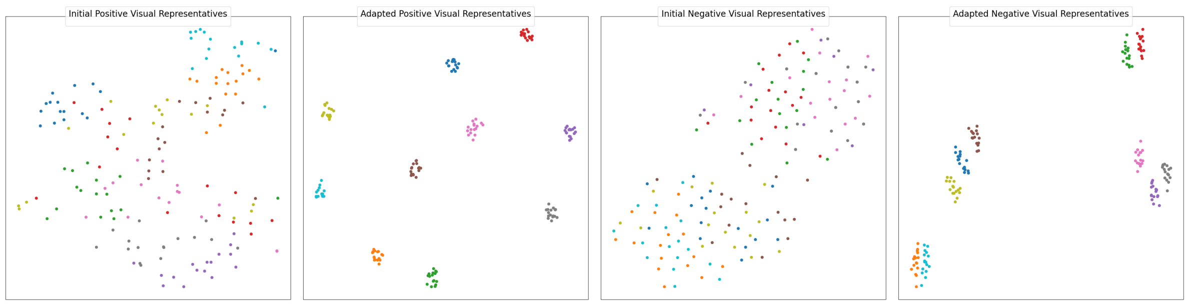

Table B.5: The generalization experiments across various CLIP backbones. The model is optimized on the ImageNet dataset (denoted as 'Source') with a 16-shot setting and tested on OOD datasets (denoted as 'Target'). The best performance is marked in bold, while the second best is underlined.

Method	Backbone	Source	Target	
		ImageNet	ImageNet-V2	ImageNet-Sketch
Zero-shot CLIP	ResNet-50	60.33	53.27	35.44
Linear Probe CLIP	ResNet-50	56.13	45.61	19.13
CoOp	ResNet-50	62.95	55.40	34.67
TaskRes	ResNet-50	64.75	56.47	35.83
GraphAdapter	ResNet-50	65.70	56.58	<u>35.89</u>
HeGraphAdapter	ResNet-50	<u>66.06</u>	<u>56.99</u>	35.24
HomoGraphAdapter	ResNet-50	**66.58**	**57.89**	**36.40**
Zero-shot CLIP	ResNet-101	61.62	54.81	38.71
Linear Probe CLIP	ResNet-101	59.75	50.05	26.80
CoOp	ResNet-101	66.60	58.66	39.08
TaskRes	ResNet-101	67.70	59.50	41.70
GraphAdapter	ResNet-101	68.23	59.60	40.83
HeGraphAdapter	ResNet-101	<u>68.60</u>	<u>59.82</u>	<u>41.88</u>
HomoGraphAdapter	ResNet-101	**69.65**	**60.52**	**42.18**
Zero-shot CLIP	ViT-B/16	66.73	60.83	46.15
Linear Probe CLIP	ViT-B/16	65.85	56.26	34.77
CoOp	ViT-B/16	71.92	64.18	46.71
TaskRes	ViT-B/16	73.07	65.30	49.13
GraphAdapter	ViT-B/16	73.68	<u>65.57</u>	48.57
HeGraphAdapter	ViT-B/16	<u>73.82</u>	65.39	<u>49.31</u>
HomoGraphAdapter	ViT-B/16	**74.70**	**66.24**	**49.88**
Zero-shot CLIP	ViT-B/32	62.05	54.79	40.82
Linear Probe CLIP	ViT-B/32	59.58	49.73	28.06
CoOp	ViT-B/32	66.85	58.08	40.44
TaskRes	ViT-B/32	68.20	59.20	42.50
GraphAdapter	ViT-B/32	68.80	59.00	41.70
HeGraphAdapter	ViT-B/32	<u>68.85</u>	<u>59.62</u>	<u>42.77</u>
HomoGraphAdapter	ViT-B/32	**69.85**	**59.85**	**43.35**

Figure C.7: Grad-CAM (Selvaraju et al., 2017) visualizes similarity heatmaps using the learned positive and negative textual embeddings for the ground-truth class in the Food101 dataset. From left to right, the images display the input image, the positive heatmap, and the negative heatmap.

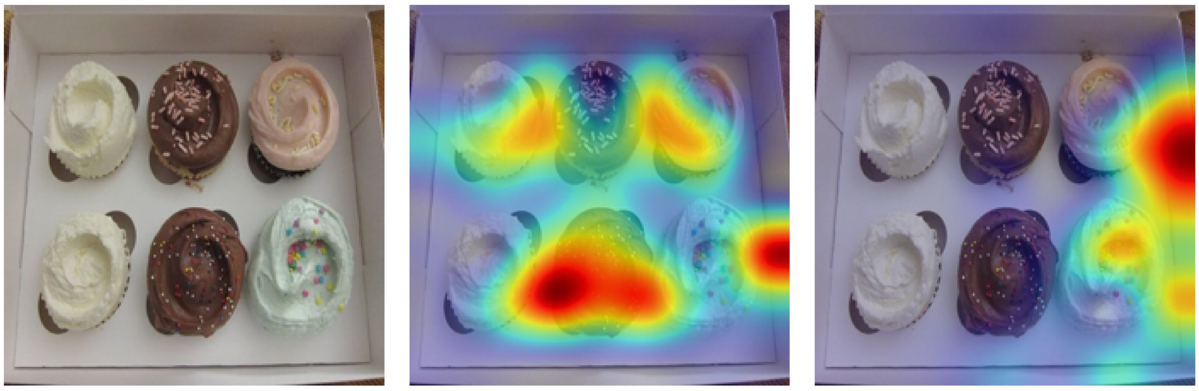

Table D.6: Pre-defined positive and negative prompt templates designed for each dataset.

Dataset	Classes	Positive Prompt Templates	Negative Prompt Templates
Caltech101	101	'A photo of a [Category].'	'A photo of no [Category].'
EuroSAT	10	'A centered satellite photo of [Category]'	'A centered satellite photo of no [Category]'
FGVCAircraft	100	'A photo with [Category] aircraft.'	'A photo without [Category] aircraft.'
SUN397	397	'A photo of a [Category].'	'A photo of no [Category].'
StanfordCars	196	'A photo of a [Category].'	'A photo of no [Category].'
UCF101	101	'A photo of a person doing [Category].'	'A photo of a person not doing [Category].'
Flowers102	102	'A photo of a [Category], a type of flowers.'	'A photo of no [Category], a type of flowers.'
Food101	101	'A photo of a [Category], a type of food.'	'A photo of no [Category], a type of food.'
DTD	47	'[Category] texture.'	'not [Category] texture.'
OxfordPets	37	'A photo of a [Category], a type of pets.'	'A photo of no [Category], a type of pets.'
ImageNet ImageNet-V2 ImageNet-Sketch	1000	'Itap of a [Category].', 'A bad photo of the [Category].', 'A origami [Category].', 'A photo of the large [Category].', 'A [Category] in a video game.', 'Art of the [Category].', 'A photo of the small [Category].', 'An image of a [Category] with bright and natural lighting.'	'Itap without any [Category].', 'A bad photo with no [Category] in it.', 'A origami piece that isn't a [Category].', 'A photo with no large [Category].', 'A video game scene without a [Category].', 'Art that doesn't include a [Category].', 'A photo with no small [Category].', 'A landscape devoid of any [Category].', 'An image completely lacking a [Category].', 'A scene with no trace of [Category].', 'An empty space without any [Category].', 'A picture where [Category] is conspicuously absent.', 'A setting that is free from any [Category].'

Table D.7: Descriptive sentences for the class "Pasture Land" in Eurosat generated by GPT-3.

Dataset	Classes	Long detailed text descriptions
Eurosat	10	'A centered satellite photo of Pasture Land would look like large green fields with animals grazing on them.' 'A centered satellite photo of Pasture Land would look like a large green field with some areas of brown or bare earth in between.' 'A centered satellite photo of Pasture Land would look like a large green field broken up by areas of trees, bushes, or other foliage.' 'A centered satellite photo of Pasture Land would look like large green fields with small areas of brown or bare earth in between.' 'A centered satellite photo of Pasture Land would look like a large green field with small patches of brown or bare earth in between.' 'A centered satellite photo of Pasture Land would look like a large green or tan field with small patches of brown or bare earth in between.' 'A centered satellite photo of Pasture Land would look like a large green or brown field with small patches of different colors in between.' 'A centered satellite photo of Pasture Land would look like a large field of green with small brown or black spots (cows).' 'A centered satellite photo of Pasture Land would look like large green fields with some areas of brown or bare earth in between.' 'A centered satellite photo of Pasture Land would look like large green fields with small patches of brown or bare earth in between.'