Leveraging Cross-Modal Neighbor Representation for Improved CLIP Classification

Chao Yi, Lu Ren, De-Chuan Zhan, Han-Jia Ye(✉)
National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{yic,renl,zhandc,yehj}@lamda.nju.edu.cn

Abstract

CLIP showcases exceptional cross-modal matching capabilities due to its training on image-text contrastive learning tasks. However, without specific optimization for unimodal scenarios, its performance in single-modality feature extraction might be suboptimal. Despite this, some studies have directly used CLIP's image encoder for tasks like few-shot classification, introducing a misalignment between its pre-training objectives and feature extraction methods. This inconsistency can diminish the quality of the image's feature representation, adversely affecting CLIP's effectiveness in target tasks. In this paper, we view text features as precise neighbors of image features in CLIP's space and present a novel ***Cr****oss-mo****D****al n****E****ighbor* ***R****epresentation* (CODER) *based on the distance structure between images and their neighbor texts. This feature extraction method aligns better with CLIP's pre-training objectives, thereby fully leveraging CLIP's robust cross-modal capabilities. The key to construct a high-quality* CODER *lies in how to create a vast amount of high-quality and diverse texts to match with images. We introduce the* ***A****uto* ***T****ext* ***G****enerator* (ATG) *to automatically generate the required texts in a data-free and training-free manner. We apply* CODER *to CLIP's zero-shot and few-shot image classification tasks. Experiment results across various datasets and models confirm* CODER*'s effectiveness. Code is available at: https://github.com/YCaigogogo/CVPR24-CODER.*

1. Introduction

In recent years, vision-language models have garnered widespread attention, with CLIP [16] standing out as a notably powerful exemplar. Trained on a vast array of image-text pairs through the image-text contrastive learning tasks, CLIP boasts impressive cross-modal matching capabilities. And it has been applied to fields like image classification [16], object detection [3], semantic segmentation [7, 25], video understanding [10], voice classification [4], text-to-image generation [19, 34], model pretraining [23], and beyond [17, 31].

Some existing works [21, 23, 32] extract image features directly from CLIP's image encoder for intra-modal operation, like calculating similarity between images in few-shot classification [32]. However, these methods overlook CLIP's multi-modal capabilities, leading to a misalignment with CLIP's pre-training objectives. Furthermore, since CLIP isn't optimized for uni-modal scenarios, its performance in intra-modal tasks isn't guaranteed. To optimize the image features extracted by CLIP, we ask:

> Can we leverage CLIP's powerful cross-modal matching capabilities to extract better image features, thereby improving CLIP's performance on downstream tasks?

In this paper, we introduce an enhanced image representation based on the distance between images and their neighboring texts in CLIP's feature space. This idea stems from our re-examination of CLIP's robust zero-shot classification capability from the perspective of nearest neighboring: The previous approaches consider the text features extracted by CLIP as classifiers and use them to get the classification results. Different from this perspective, we interpret CLIP's zero-shot image classification as a 1NN problem, as shown in the left part of Figure 1. We treat texts as the images' neighbors in the CLIP's feature space. Then for each image, CLIP identifies its closest text and assigns this text's class as the image's predicted class. This 1NN approach shows good performance because CLIP's robust image-text matching capabilities ensure images are closer to their semantically related texts. This suggests that the cross-modal distance between an image and its neighboring texts captures information about the image, such as its class.

In zero-shot image classification, CLIP only considers the distance between an image and its nearest neighbor text, which loses the information implied in the distance between the image and other texts. To make full use of this infor-

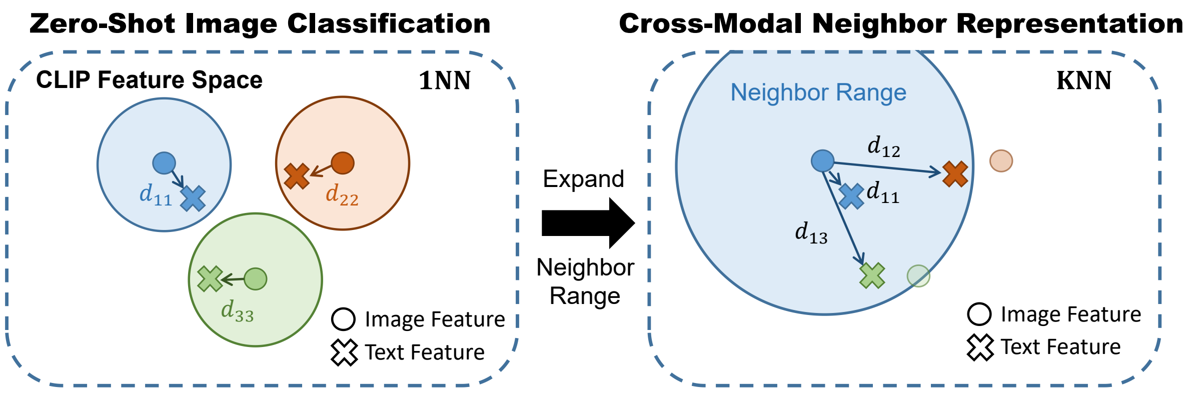

Figure 1. **Illustration of image's CrOss-moDal nEighbor Representation (CODER).** CLIP's powerful text-image matching capabilities endows it with a favorable cross-modal neighbor distance relation. And CLIP's Zero-Shot Image Classification process can be interpreted as using a 1NN algorithm to find the image's nearest text, with the text's class determining the image's predicted class. Inspired by this idea, we expand the image's neighbor range to leverage its distance to all texts for constructing the CODER. Here d_{ij} refers to the distance between the i-th image and the j-th text.

mation, we expand each image's neighbor range to utilize its distance to ***K*** **N**earest **N**eighbor (KNN) texts for constructing image representation, as depicted in the right half of Figure 1. Here K denotes the total number of texts. We refer to this representation as **C**r**O**ss-mo**D**al n**E**ighbor **R**epresentation (CODER). We believe images with closer CODER values are more similar in semantics. This aligns with intuition: If two objects share the same sets of similar and dissimilar items, they're likely similar to each other.

Previous work [24, 33] has noted that dense sampling of neighbor samples is critical for building neighbor representations. This inspires us to use various high-quality texts for dense sampling. Considering that Large Models have rich knowledge and are widely used in many ways [9, 14, 27], we introduce the **A**uto **T**ext **G**enerator (ATG) based on Large Language Models like ChatGPT [13] to automatically generate various high-quality texts. Without the need for image data and the training process, ATG can produce a diverse and effective set of texts based on the target dataset's class names. These diverse, high-quality texts enhance the density of neighboring texts for images in CLIP's feature space, helping to build a better CODER.

We apply CODER to CLIP's zero-shot and few-shot image classification tasks to prove CODER's superiority. For the zero-shot image classification task, we use CODER in a two-stage manner. In the first stage, we use ATG to acquire general class-specific texts for constructing general CODER. Then we employ a heuristic classifier to the image's general CODER. In the second stage, we utilize ATG to generate one-to-one specific texts for comparing two distinct classes, concentrating on their most distinguishing features. These texts are then used to construct the one-to-one specific CODER, which is used to rerank the preliminary classification results. For the few-shot situation, we calculate the similarity between test images and the support set's images using those images' general CODER. By ensembling the similarity and CLIP's original zero-shot classification logits, we determine the classification results. Experiment results on various datasets and different CLIP models confirm that CODER enhances CLIP's performance in both zero-shot and few-shot image classification.

2. Related Work

Vision-Language Models. Vision-Language Models (VLMs) represent a class of multimodal models adept at correlating textual and visual information. Prominent models in this domain include CLIP [16], ALIGN [6], FLAVA [18] and Florence [30], among others. These models typically comprise two main components: an image encoder and a text encoder. Some models also have multimodal fusion modules [8, 22]. VLMs are trained on extensive text-image pairs through tasks like image-text contrastive learning, endowing them with powerful text-image matching capabilities. In this paper, we leverage these capabilities of CLIP to generate our cross-modal neighbor representations (CODER) for images.

Using LLMs as Experts to Improve VLMs. Large Language Models (LLMs) are widely used in many tasks, such as in-context learning. LLMs can serve as experts by outputting their knowledge in text form. This text-based knowledge can be harnessed by VLMs to enhance their capabilities. Previous work has explored using LLMs' knowledge to optimize VLMs' pre-training [1], prediction interpretability [12, 26, 28], and classifier quality [2, 11, 12, 15]. Recently, some work [29, 35] uses LLMs to select the best VLMs from a VLM zoo for specific target tasks. In this paper, we use the semantic information provided by LLMs to optimize CLIP's features during the inference stage in a training-free manner.

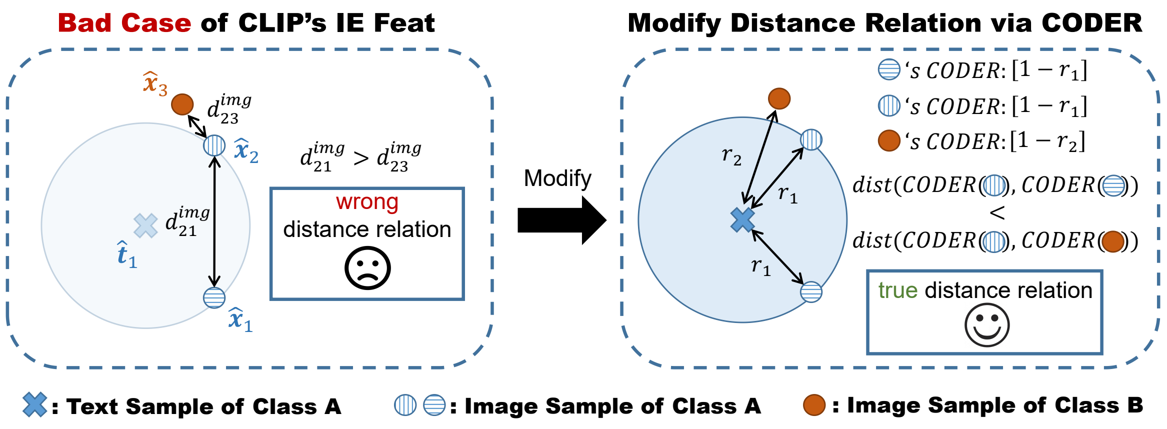

Figure 2. **An example of** CODER **correcting wrong distance relation between images.** d_{ij}^{img} refers to the cosine distance between the i-th and the j-th image. r_1 and r_2 refer to the cosine distance between the text and different images, while $1 - r_1$ and $1 - r_2$ refer to the cosine similarity. The left side of the figure indicates that even though images of the same class share similar distance to a text, this doesn't ensure that their features are closely similar. The right side of the figure shows that CODER corrects the wrong distance relation by utilizing the text-image distance.

3. Notations and Background

Using CLIP to match text and image. CLIP encodes both images and texts into a joint space using its image encoder f^I or text encoder f^T, shown in Equation 1. Here $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{t}}_j$ refers to the feature of image $\boldsymbol{x}_i$ and text $\boldsymbol{t}_j$, respectively. D_1 refers to the dimension of CLIP's feature space.

$$\hat{\boldsymbol{x}}_i = f^I(\boldsymbol{x}_i) \in \mathbb{R}^{D_1},\ \hat{\boldsymbol{t}}_j = f^T(\boldsymbol{t}_j) \in \mathbb{R}^{D_1}. \tag{1}$$

$$\tilde{\boldsymbol{t}} = \underset{j\in[1,\cdots,K]}{\arg\max} \frac{\hat{\boldsymbol{x}}_i^\top \hat{\boldsymbol{t}}_j}{\|\hat{\boldsymbol{x}}_i\| \cdot \|\hat{\boldsymbol{t}}_j\|}. \tag{2}$$

Then we can compute the cosine similarity between the features of various texts and images. These similarity scores represent the matching degree between different images and texts. Given an image $\boldsymbol{x}_i$, its best matched text $\tilde{t}$ is identified by the highest similarity scores with the image, as shown in Equation 2. Here K refers to the number of texts.

4. Cross-Modal Neighbor Representation

4.1. Understand the Advantage of CODER.

We construct the CODER for the current image by tapping into the precise image-text distance relationship within the CLIP feature space. This process is shown in Equation 3. We use the CODER construct function ϕ to build image's CODER $\phi(\hat{\boldsymbol{x}}_i)$ based on its original feature $\hat{\boldsymbol{x}}_i$.

$$\phi(\hat{\boldsymbol{x}}_i) = \left[\psi\left(d\left(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{t}}_1\right)\right), \cdots, \psi\left(d\left(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{t}}_K\right)\right)\right] \in \mathbb{R}^K. \tag{3}$$

The specific implementation of ϕ depends on the similarity or distance function d and the subsequent mapping function ψ applied to this distance. In this paper, we use cosine similarity for d and identity mapping for ψ. Then we can rewrite Equation 3 as Equation 4. We emphasize that the implementation of d and ψ can be further researched.

$$\phi(\hat{\boldsymbol{x}}_i) = \left[\frac{\hat{\boldsymbol{x}}_i^\top \hat{\boldsymbol{t}}_1}{\|\hat{\boldsymbol{x}}_i\| \cdot \|\hat{\boldsymbol{t}}_1\|}, \cdots, \frac{\hat{\boldsymbol{x}}_i^\top \hat{\boldsymbol{t}}_K}{\|\hat{\boldsymbol{x}}_i\| \cdot \|\hat{\boldsymbol{t}}_K\|}\right] \in \mathbb{R}^K. \tag{4}$$

We highlight CODER's advantages over CLIP's original image features using an example. Figure 2's left side depicts a bad case for CLIP. For simplicity, we consider a situation where test images share the same neighboring text in CLIP's feature space. While CLIP's cross-modal pre-training ensures accurate image-text distances, it doesn't always capture precise distances between images. This results in cases where the distance for same-class images d_{21}^{img} exceeds that of different-class images d_{23}^{img}. To solve this problem, CODER uses CLIP's accurate text-image distance to build image features. As images of the same class have similar distance to their neighboring texts, their CODER align more closely. Thus, CODER addresses the wrong distance relation between images.

We then focus on the key element of building a good CODER. Previous studies have emphasized that dense sampling of neighboring samples is vital for algorithms based on nearest neighbor. For example, only when the training samples are densely sampled will the error rate of the KNN classifier remain within twice that of the Bayes optimal classifier. And some studies [24, 33] have observed that greater sampling density of neighboring samples can lead to better neighbor representations. Inspired by these works, we try to optimize our CODER by increasing the number of texts. For increasing the number of texts, we use our **A**uto **T**ext **G**enerator (ATG) to achieve this objective.

4.2. Use Auto Text Generator to Generate Texts

Accurate and diverse class-specific texts provide a comprehensive description of object classes from various perspectives, enhancing the sampling density of images' neighbor texts for constructing better CODER. To automatically generate a plethora of high-quality class-specific texts adaptive to target tasks, we introduce the **A**uto **T**ext **G**enerator (ATG). It uses different query prompts to extract diverse insights from external experts like ChatGPT [13]. Then it leverages that knowledge to construct various high-quality texts. These texts are instrumental in generating high-quality CODER.

In our implementation, ATG can construct five types of texts: (1) Class Name-based Texts; (2) Attribute-based Texts; (3) Analogous Class-based Texts; (4) Synonym-based Texts; (5) One-to-One Texts. The first two types of texts are proposed by previous work[12, 16], while the latter three are innovations introduced in this paper. We will then delve deeper into the design rationale and generation process of these last three types of texts.

(1) **Analogous Class-based Texts**. If someone describes a clouded leopard as resembling a cheetah, you can envision its appearance even if you've never seen a clouded leopard. This scenario illustrates how inter-object similarities help humans leverage their experience with known object classes to recognize new ones. Inspired by this, we query ChatGPT to obtain analogous categories for a given object by asking:

```
Q: What other categories are {class}
visually similar to?
```

Given that the generated class names might exist within the target dataset's class names, we address this issue by filtering out those generated class names whose cosine similarity with existing class names surpass a defined threshold. Next, we insert each analogous class name into some templates like "`a {target class} similar to {analogous class}`" to form a complete text.

(2) **Synonym-based Texts**. An object can have several names. For example, both "forest" and "woodland" refer to "land covered with trees and shrubs". Due to varying frequencies of synonyms in CLIP's training data, the model might favor more common terms and undervalue lesser-known synonyms, despite their equivalent meanings. To mitigate this bias, we query WordNet for synonyms of the current class. Subsequently, we insert the obtained synonym class names into templates like "`a photo of {synonym class}`".

(3) **One-to-One Texts**: Due to similar classes often share common features, the ATG may generate nearly identical attribute descriptions for these classes. For example, the ATG might generate attributes such as "two pairs of wings" for both "butterfly" and "dragonfly". Such identical attributes can hinder CLIP's ability to distinguish closely related classes. To tackle this problem, we introduce the one-to-one texts. We use the following query prompt to guide ChatGPT in generating the most distinguishing features between similar classes A and B:

```
Q: What are different visual features
between a {class 1} and a {class 2} in a
photo?  Focus on their key differences.
```

Using the prompt, we generate distinguishing attributes that differentiate butterfly from dragonfly. For butterfly, some of the exemplified attributes include: ["Butterflies typically have larger and more colorful wings compared to dragonflies."]. For dragonfly, some of the descriptors produced are: ["They have transparent wings that are typically held out horizontally when at rest."]. From these newly created attributes, we can make two main observations: (1) The attributes underscore the key differences between butterfly and dragonfly, such as their wing characteristics. (2) The new attributes accentuate the comparison between the two classes, evident from terms like "larger", and "compared to" found within the descriptors.

Finally, we insert the obtained one-to-one texts into some templates like "`Because of {1v1 text}, {class 1} is different from {class 2}`".

4.3. Use CODER on Downstream Tasks

We apply CODER in zero-shot and few-shot image classification tasks to enhance the performance of CLIP.

Zero-Shot Image Classification. In the first stage, we utilize a general text set generated by ATG to construct the general CODER for images. These texts can be considered as descriptions of general characteristics for a certain class in comparison to most other classes. We can derive preliminary classification results based on the general CODER of the images. In the second stage, we employ ATG to create distinct one-to-one specific text sets for the top five classes, each pair-wise. These texts serve as descriptions of the key distinguishing features between the classes. Utilizing these, we then develop corresponding one-to-one specific CODER for images. Subsequently, we rerank the initial top five classification results using these one-to-one specific CODER. Figure 3 shows the process of our zero-shot image classification method.

We first utilize all texts generated by ATG, excluding the one-to-one texts, as the general text set $\mathcal{P} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_K]$ to construct the general CODER for test images. This process is shown in Equation 5 and Equation 6. Here $\boldsymbol{W} \in \mathbb{R}^{D_1 \times K}$ refers to the texts' features in the CLIP feature space. And $\boldsymbol{s}_i \in \mathbb{R}^K$ represents the CODER of test image $\boldsymbol{x}_i$. K refers to the total number of texts in $\mathcal{P}$.

$$\boldsymbol{W} = \left[\frac{\hat{\boldsymbol{t}}_1}{\|\hat{\boldsymbol{t}}_1\|}, \cdots, \frac{\hat{\boldsymbol{t}}_K}{\|\hat{\boldsymbol{t}}_K\|}\right] \in \mathbb{R}^{D_1 \times K}. \tag{5}$$

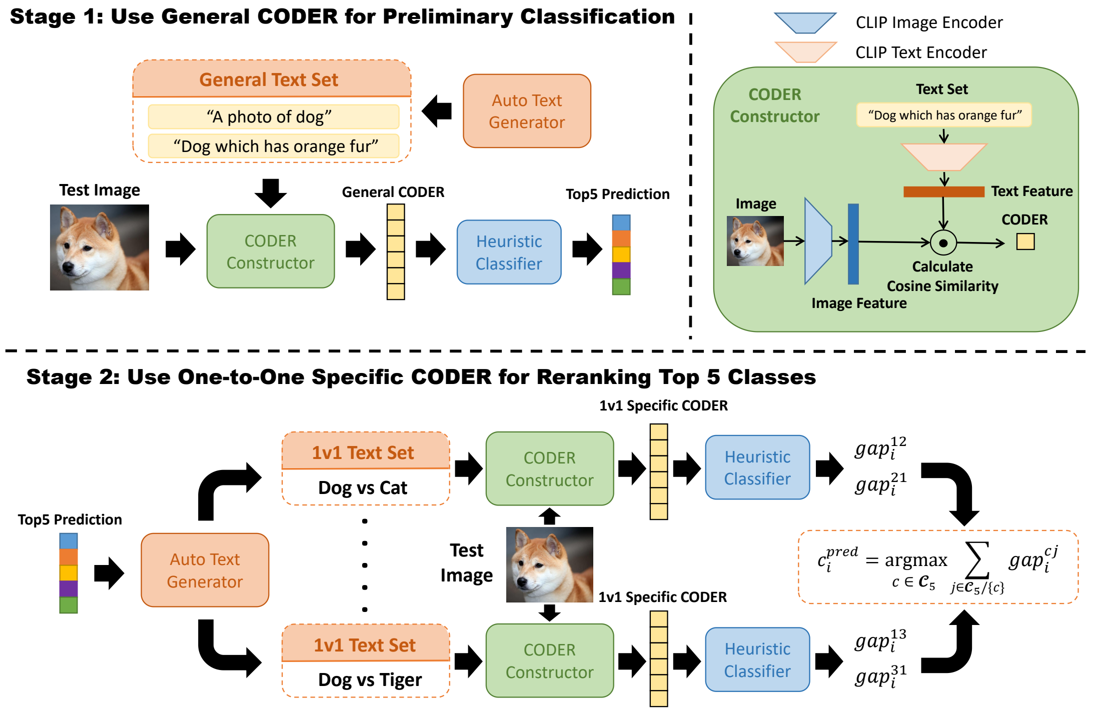

Figure 3. **Illustration of two-stage zero-shot image classification process based on image's** CODER. In the first stage, we use the Auto Text Generator to create a General Text Set, which contains general descriptions of classes. This set is utilized to construct the image's general CODER, and we use it for preliminary classification. In the second stage, we construct One-to-One Text Sets in pairs for the top five predicted classes of the preliminary classification results, focusing on attributes where the two specific classes differ most. We build one-to-one specific CODER for the image based on these One-to-One Text Sets and use heuristic classifier to get each class's classification score. Then we rerank the top five preliminary results based on the classification score gaps gap_i^{cj} between classes. Here gap_i^{cj} represents the difference obtained by subtracting the score of the class c from that of the class j for image $\boldsymbol{x}_i$.

$$\boldsymbol{s}_i = \phi\left(\hat{\boldsymbol{x}}_i\right) = \frac{\hat{\boldsymbol{x}}_i^\top}{\|\hat{\boldsymbol{x}}_i\|} \boldsymbol{W} \in \mathbb{R}^K. \tag{6}$$

Then, we employ a heuristic classifier h on the test image's general CODER $\boldsymbol{s}_i$ to obtain the preliminary classification logits vector $\boldsymbol{o}_i$. We use $\boldsymbol{s}_{ij}$ to represent the portion of the test image's general CODER $\boldsymbol{s}_i$ corresponding to the text of the j-th category. $\boldsymbol{s}_{ij}$ consist of several parts, as shown in Equation 7. Here $\boldsymbol{s}_{ij}^{ori}$, $\boldsymbol{s}_{ij}^{att}$, $\boldsymbol{s}_{ij}^{ana}$, $\boldsymbol{s}_{ij}^{syn}$ refers to $\boldsymbol{s}_{ij}$'s portion of class name-based texts, attribute-based texts, analogous class-based texts and synonym-based texts, respectively. $\oplus$ refers to the vector concatenation operation.

$$\boldsymbol{s}_{ij} = \boldsymbol{s}_{ij}^{ori} \oplus \boldsymbol{s}_{ij}^{att} \oplus \boldsymbol{s}_{ij}^{ana} \oplus \boldsymbol{s}_{ij}^{syn}. \tag{7}$$

For each class, the heuristic classifier h first gets the largest element in the test image's CODER portion corresponding to the class name-based texts and synonym-based texts. This step is intuitive: Humans can recognize an object by knowing just one of its names. Then h calculates the mean of $\max(\boldsymbol{s}_{ij}^{ori} \oplus \boldsymbol{s}_{ij}^{syn})$ and all elements in the $\boldsymbol{s}_{ij}$'s portion corresponding to the attribute-based texts and synonym-based texts. This value can be seen as the preliminary classification logit value o_{ij} belonging to class j. The process is shown in the following Equation:

$$o_{ij} = h(\boldsymbol{s}_{ij}) = \operatorname{mean}\left(\boldsymbol{s}_{ij}^{att} \oplus \boldsymbol{s}_{ij}^{ana} \oplus \left[\max(\boldsymbol{s}_{ij}^{ori} \oplus \boldsymbol{s}_{ij}^{syn})\right]\right). \tag{8}$$

After obtaining the preliminary classification results $\boldsymbol{o}_i$, we extract the top five classes from the prediction results, forming a set named $\mathcal{C}_5$. Subsequently, we draw all pairwise combinations from these five classes in set $\mathcal{C}_5$, creating a new set $\mathcal{P}_{1v1}$ comprised of these pairs. This process can be described by Equation 9 and Equation 10:

$$\mathcal{C}_5 = \operatorname{Top5}\left(\boldsymbol{o}_i\right) = \left\{c_1, c_2, c_3, c_4, c_5\right\}. \tag{9}$$

$$\mathcal{P}_{1v1} = \left\{\{c_a, c_b\} \,|c_a, c_b \in \mathcal{C}_5, a \neq b\right\}. \tag{10}$$

In the second stage, we use ATG to generate the one-to-one specific texts for each pair in $\mathcal{P}_{1v1}$, resulting in a total of $C_5^2 = 10$ text sets. For each one-to-one specific text set, we can construct a one-to-one specific CODER for the image. We first use CLIP's text encoder f^T to extract the text features of the one-to-one text set $\boldsymbol{W}^{a,b} \in \mathbb{R}^{D_1 \times (K_1+K_2)}$. Here K_1 and K_2 represent the number of one-to-one texts for each of the two classes. Equation 11 shows the composition of $\boldsymbol{W}^{a,b}$, where $\hat{\boldsymbol{t}}_i^a$ and $\hat{\boldsymbol{t}}_i^b$ refers to the i-th one-to-one text feature of class a and class b, respectively.

$$\boldsymbol{W}^{a,b} = \left[\frac{\hat{\boldsymbol{t}}_1^a}{\|\hat{\boldsymbol{t}}_1^a\|}, \cdots, \frac{\hat{\boldsymbol{t}}_{K_1}^a}{\|\hat{\boldsymbol{t}}_{K_1}^a\|}, \frac{\hat{\boldsymbol{t}}_1^b}{\|\hat{\boldsymbol{t}}_1^b\|}, \cdots, \frac{\hat{\boldsymbol{t}}_{K_2}^b}{\|\hat{\boldsymbol{t}}_{K_2}^b\|}\right]. \quad (11)$$

Then we use the one-to-one text features $\boldsymbol{W}^{a,b}$ to construct the image's one-to-one specific CODER $\boldsymbol{s}_i^{a,b} \in \mathbb{R}^{K_1+K_2}$:

$$\boldsymbol{s}_i^{a,b} = \frac{\hat{\boldsymbol{x}}_i^\top}{\|\hat{\boldsymbol{x}}_i\|}\boldsymbol{W}^{a,b} = \boldsymbol{s}_{ia}^{a,b} \oplus \boldsymbol{s}_{ib}^{a,b}. \quad (12)$$

Here $\boldsymbol{s}_{ia}^{a,b} \in \mathbb{R}^{K_1}$ and $\boldsymbol{s}_{ib}^{a,b} \in \mathbb{R}^{K_2}$ represent the part of CODER pertaining to the one-to-one texts for class a and class b, respectively.

Based on the one-to-one specific CODER $\boldsymbol{s}_i^{a,b}$ of the image $\boldsymbol{x}_i$, we employ a heuristic classifier h to obtain image's classification scores for class a and b, denoted as $o_{ia}^{a,b}$ and $o_{ib}^{a,b}$, respectively:

$$\begin{aligned} o_{ia}^{a,b} &= h(\boldsymbol{s}_{ia}^{a,b}) = \text{mean}\left(\boldsymbol{s}_{ia}^{a,b}\right), \\ o_{ib}^{a,b} &= h(\boldsymbol{s}_{ib}^{a,b}) = \text{mean}\left(\boldsymbol{s}_{ib}^{a,b}\right). \end{aligned} \quad (13)$$

Then we calculate the image's classification score gap for each class relative to another class:

$$gap_i^{ab} = o_{ia}^{a,b} - o_{ib}^{a,b}, \quad gap_i^{ba} = o_{ib}^{a,b} - o_{ia}^{a,b}. \quad (14)$$

For each class, we sum up all the score gaps between it and other classes. Then we select the class with the largest sum of score gaps as the final predicted class $\hat{c}_i$ of the image:

$$\hat{c}_i = \underset{c \in \mathcal{C}_5}{\text{argmax}} \sum_{j \in \mathcal{C}_5/\{c\}} gap_i^{cj}. \quad (15)$$

We use the method based on score gaps instead of the traditional voting method to reorder initial prediction results. The reason is that score gaps provide quantified information about the relative advantages between classes. Unlike voting, which only considers each class's number of wins in pairwise classification, score gaps better capture the model's uncertainty in the classification tasks. For example, a small score gap might indicate that the model is uncertain about the correct class for the image, suggesting that the classification result may be unreliable. This can lead to errors in the outcomes derived from voting methods.

This two-phase image classification process illustrates that by changing the neighbor text set, we can dynamically build image's feature that focus on different semantics according to task requirements. In coarse-grained image classification, we focus on the general semantics of a class, such as whether an animal has wings. In fine-grained image classification, we aim to focus on more detailed features that differentiate between two specific classes, such as the color of wings when distinguishing between dragonfly and butterfly. However, the original image features extracted by the CLIP's image encoder cannot dynamically adapt to specific classification task requirements.

Few-Shot Image Classification. For Few-Shot Image Classification, we improve the Tip-Adapter [32] by replacing the original CLIP's image features with our CODER to calculate the similarity between the test image and the support set's images. We refer to the improved method as **C**r**O**ss-Mo**D**al N**E**ighbor **R**epresentation CLIP Adapter (CODER-Adapter). Given the N-way M-shot support image set $\mathcal{I}$, we first calculate the CODER of the images in the support set, represented by $\boldsymbol{S}_{train} \in \mathbb{R}^{NM \times K}$. Equation 16 shows the calculation process, where $f^I(\mathcal{I}) \in \mathbb{R}^{NM \times D_1}$ refers to the support images' original features and $\boldsymbol{W} \in \mathbb{R}^{D_1 \times K}$ refers to the features of texts in the general text set $\mathcal{P}$, as defined in Equation 5.

$$\boldsymbol{S}_{train} = f^I(\mathcal{I})\,\boldsymbol{W}. \quad (16)$$

Then we perform one-hot encoding on the support images' labels $\boldsymbol{L}$ to get one-hot labels matrix $\boldsymbol{L}_{train} \in \mathbb{R}^{NM \times C}$. Each row of $\boldsymbol{L}_{train}$ is a one-hot vector.

$$\boldsymbol{L}_{train} = \text{OneHot}(\boldsymbol{L}). \quad (17)$$

For the test image $\boldsymbol{x}_i$, we also construct its CODER $\boldsymbol{s}_i \in \mathbb{R}^K$ similarly to the support set's images.

$$\boldsymbol{s}_i = f^I(\boldsymbol{x}_i)\,\boldsymbol{W}. \quad (18)$$

We then calculate the affinity matrix $\boldsymbol{A} \in \mathbb{R}^{1 \times NM}$ between the test image's CODER $\boldsymbol{s}_i \in \mathbb{R}^K$ and the support set images' CODER $\boldsymbol{S}_{train} \in \mathbb{R}^{NM \times K}$ using the Equation 19. Here Norm$(\cdot)$ refers to the data normalization operation like L2 or min-max normalization. β and T are hyperparameters to control the sharpness of $\boldsymbol{A}$'s distribution.

$$\boldsymbol{A} = \exp\left(-\beta \cdot \left(1 - \frac{\text{Norm}\left(\boldsymbol{s}_i \boldsymbol{S}_{train}^\top\right)}{T}\right)\right). \quad (19)$$

The affinity $\boldsymbol{A} \in \mathbb{R}^{1 \times NM}$ serves as a weight factor for $\boldsymbol{L}_{train}$. By calculating the weighted sum of images' labels in $\boldsymbol{L}_{train}$, we can refine the CLIP's original zero-shot prediction results $\boldsymbol{o}_i^{zs}$ using Equation 20. Here α controls the degree of correction.

$$\boldsymbol{o}_i = \alpha \cdot \boldsymbol{A}\boldsymbol{L}_{train} + \boldsymbol{o}_i^{zs}. \quad (20)$$

Finally, we select the class corresponding to the largest logits in $\boldsymbol{o}_i$ as the predicted class of image $\boldsymbol{x}_i$.

5. Experiments

5.1. Zero-Shot Image Classification Performance

For zero-shot image classification, we compare our method with two baselines: Vanilla CLIP [16] and VCD [12]. Ta-

Table 1. Accuracy gains over VCD and CLIP baseline. The "CODER" column shows the classification accuracy before rerank stage, while the "CODER*" column shows the accuracy after rerank stage. Δ represents the improvement of our method over VCD.

	ImageNet					CUB200					EuroSAT				
Architecture	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ
ViT-B/32	59.07	62.96	64.38	**66.86**	3.90	51.81	51.78	53.00	**55.04**	3.26	44.89	46.57	52.83	**54.93**	8.36
ViT-B/16	63.53	68.10	69.61	**71.46**	3.36	55.75	57.54	58.30	**59.92**	2.38	49.52	56.46	55.30	**60.54**	4.08
ViT-L/14	70.58	74.96	76.15	**77.38**	2.42	62.10	63.29	64.64	**65.76**	2.47	53.54	59.57	62.98	**68.24**	8.67
ViT-L/14@336px	71.81	76.11	77.27	**78.49**	2.38	63.48	64.79	66.29	**67.36**	2.57	54.93	59.46	64.94	**69.96**	10.50

	Describable Textures					Places365					Food101				
Architecture	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ
ViT-B/32	41.28	45.27	48.78	**52.71**	7.44	36.47	38.92	**40.37**	40.27	1.45	80.27	84.13	84.72	**85.50**	1.37
ViT-B/16	43.19	45.64	48.67	**55.69**	10.05	37.31	39.87	41.39	**41.70**	1.83	85.99	89.25	89.29	**89.82**	0.57
ViT-L/14	52.18	57.18	60.74	**61.86**	4.68	37.06	38.55	41.21	**41.59**	3.04	89.85	93.33	93.66	**93.93**	0.60
ViT-L/14@336px	52.12	57.45	61.12	**62.87**	5.42	37.27	39.88	42.09	**42.50**	2.62	91.10	94.06	94.48	**94.75**	0.69

	Caltech101					Oxford Pets					ImageNetV2				
Architecture	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ	CLIP	VCD	CODER	CODER*	Δ
ViT-B/32	79.90	89.80	91.24	**91.42**	1.62	81.63	85.91	88.47	**89.26**	3.35	51.79	55.29	**56.75**	56.42	1.46
ViT-B/16	80.18	92.28	**94.07**	93.95	1.79	83.95	89.34	91.50	**92.01**	2.67	57.25	61.53	62.94	**62.97**	1.44
ViT-L/14	79.78	93.95	**95.79**	95.68	1.84	87.90	93.18	94.19	**94.36**	1.18	64.31	69.21	70.35	**70.37**	1.16
ViT-L/14@336px	80.36	94.35	96.31	**96.37**	2.02	87.82	93.62	94.22	**94.88**	1.26	65.61	70.41	**71.45**	71.28	1.04

ble 1 displays zero-shot image classification experiment results. By analyzing the results, we can draw the following conclusions: (1) Our proposed CODER consistently boosts CLIP's zero-shot image classification accuracy across various datasets and model architectures; (2) The rerank stage based on the one-to-one specific CODER can further enhance CLIP's performance, demonstrating the effectiveness of our proposed two-stage zero-shot classification method.

5.2. Few-Shot Image Classification Performance

For few-shot image classification, we use ResNet 50[5] as the backbone for CLIP's image encoder, consistent with previous methods. We use two CLIP's training-free few-shot image classification methods TIP-Adapter [32] and TIP-X [20] as our baselines. TIP-Adapter leverages features extracted by CLIP's image encoder for calculating the similarity between test images and support set's images. While TIP-X uses features based on similarity scores between images and texts generated by CUPL [15] to do it.

Figure 4 shows the few-shot image classification experiment results. We use the experimental results of TIP-Adapter and TIP-X presented in [20] and ensure that our random seeds are consistent with them. Our method surpasses the previous CLIP few-shot training-free image classification methods on most datasets across different shot scenarios. We notice that CODER-Adapter's performance on EuroSAT is unsatisfactory, which results in our method not having a significant advantage in average accuracy across 11 datasets compared to previous methods. But this meets our expectations. Since EuroSAT has only 10 classes and 95 texts generated by ATG, this small number of texts fails to satisfy CODER's need for dense text sampling, impacting the adapter's performance. This highlights the importance of dense sampling for CODER.

Table 2. The accuracy of the zero-shot image classification experiments using different texts. Meaning of symbols: **P**: Using class name-based texts. **Att**: Using attribute-based texts. **Ana**: Using analogous class-based texts. We use CLIP ViT-B-32.

Text	Caltech101	Oxford Pets	Describable Textures	EuroSAT
P	79.90	81.63	41.28	44.89
P+Att	89.80	85.91	45.27	46.57
P+Att+Ana	**91.01**	**88.55**	**48.73**	**52.77**

Table 3. The accuracy of the 16-shot image classification experiments using different texts. We use CLIP ResNet-50.

Text	Caltech101	Oxford Pets	Describable Textures	EuroSAT
P	90.34	89.83	62.35	58.44
P+Att	90.47	**90.3**	63.06	65.36
P+Att+Ana	**90.99**	90.26	**64.18**	**67.11**

Contrasting to previous work [20, 32], we innovatively interpret the superiority of CLIP's cross-modal relative feature representation from the perspective of neighbor representation. Additionally, by drawing an analogy to neighbor-based algorithms like KNN, which require dense sampling of neighbor samples to ensure superior performance, we provide an intuitive explanation for our need to use a more diverse and higher-quality text set to construct our CODER. Motivated by this conclusion, we develop the Auto Text Generator for the automated creation of varied, high-quality texts to meet CODER's dense sampling needs. The experiments demonstrate the validity of our method.

5.3. The Importance of Dense Sampling for CODER

Nearest neighbor algorithms like KNN depend on dense sampling for good performance. We view our CODER as a cross-modal neighbor representation. Hence, as the diversity and quantity of high-quality cross-modal neighbor texts increase, the constructed CODER should correspond-

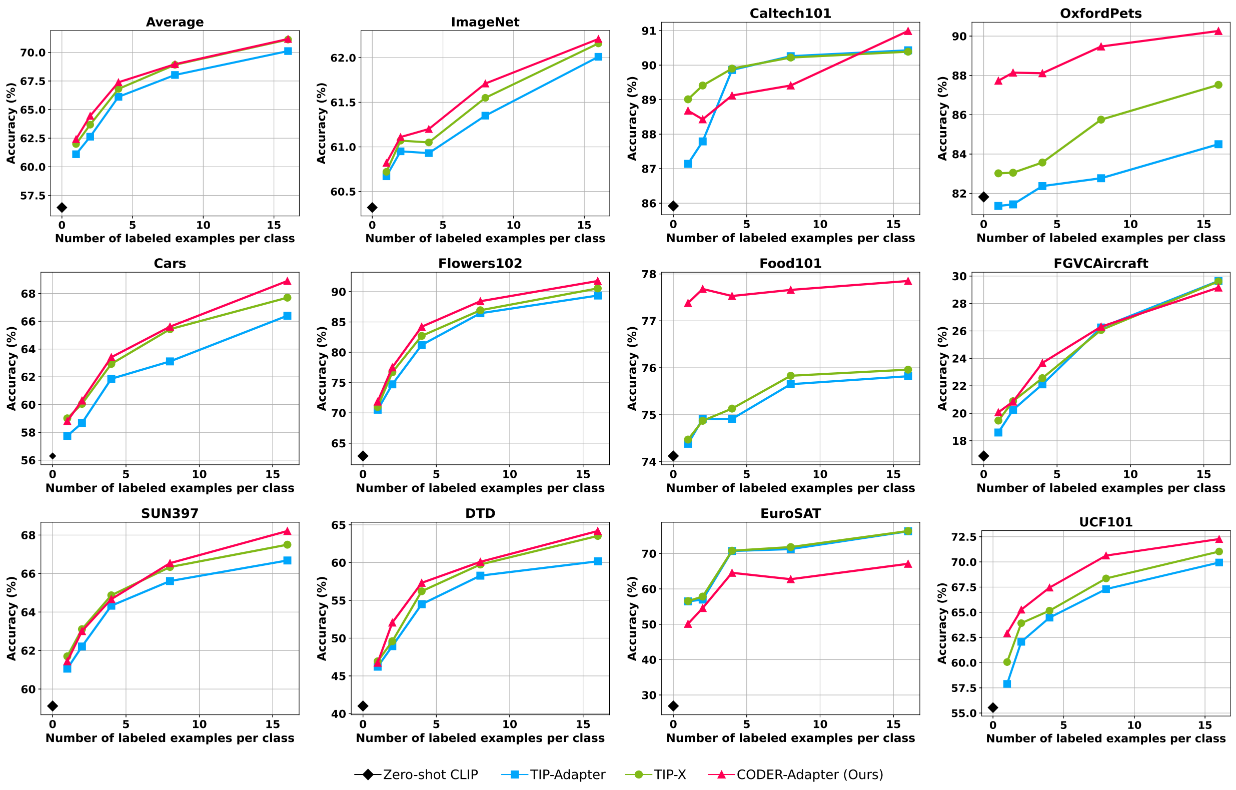

Figure 4. Results for the training-free few-shot regime across 11 datasets. We compare the CODER-Adapter with the previous CLIP few-shot image classification methods. Our CODER-Adapter achieves the best performance on most datasets.

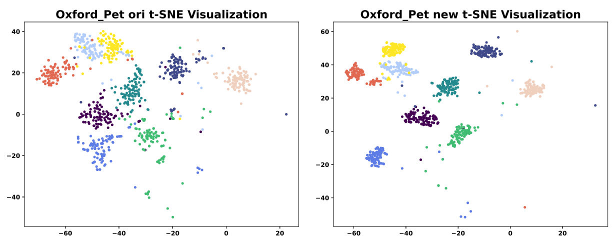

Figure 5. The t-SNE Visualization of the Oxford-Pets dataset. **Left**: Original CLIP Images' Features; **Right**: CODER.

ingly improve. We validate this idea through experiments. We use three different text sets for constructing the image's CODER: (1) CLIP's original class name-based texts; (2) class name-based texts and VCD's attribute-based texts; (3) our analogous class-baed texts added to (2). Our experiments in zero-shot and few-shot image classification, as detailed in Table 2 and Table 3, show that CODER performance enhances with the increasing diversity and number of texts. This highlights the importance of dense neighbor text sampling in improving CODER quality.

5.4. Visualization results of CODER

Figure 5 presents a t-SNE visualization comparing original image features from the CLIP image encoder with those of our proposed CODER on the Oxford-Pets dataset. Our CODER achieves tighter clustering of same-class images' features and clearer separation of different classes.

6. Conclusion

In this paper, we address the misalignment between CLIP's image feature extraction method and its pre-training paradigm. We present a novel perspective based on the nearest neighbors to comprehend CLIP's strong zero-shot image classification capability. Our key insight is that CLIP's effective text-image matching capability embeds image information in image-text distances. This leads us to propose the **Cr**Oss-mo**D**al n**E**ighbor **R**epresentation (CODER), utilizing these image-text distances for image representation. We introduce the **A**uto **T**ext **G**enerator to automatically generate texts, ensuring dense sampling of neighbor texts for better CODER construction. Experiment results in zero-shot and few-shot image classification show the superiority of our method.

Acknowledgments

This work is partially supported by National Key R&D Program of China (2022ZD0114805), NSFC (62376118, 62006112, 62250069, 61921006), Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Lijie Fan, Dilip Krishnan, Phillip Isola, Dina Katabi, and Yonglong Tian. Improving CLIP training with language rewrites. In *NeurIPS*, 2023. 2

[2] Yunhao Ge, Jie Ren, Andrew Gallagher, Yuxiao Wang, Ming-Hsuan Yang, Hartwig Adam, Laurent Itti, Balaji Lakshminarayanan, and Jiaping Zhao. Improving zero-shot generalization and robustness of multi-modal models. In *CVPR*, pages 11093–11101, 2023. 2

[3] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *ICLR*, 2022. 1

[4] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending clip to image, text and audio. In *ICASSP*, pages 976–980, 2022. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 7

[6] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, pages 4904–4916, 2021. 2

[7] Boyi Li, Kilian Q. Weinberger, Serge J. Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022. 1

[8] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Gotmare, Shafiq R. Joty, Caiming Xiong, and Steven Chu-Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *NeurIPS*, pages 9694–9705, 2021. 2

[9] Li Li, Jiawei Peng, Huiyi Chen, Chongyang Gao, and Xu Yang. How to configure good in-context sequence for visual question answering. *CoRR*, abs/2312.01571, 2023. 2

[10] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of CLIP for end to end video clip retrieval and captioning. *Neurocomputing*, 508:293–304, 2022. 1

[11] Chengzhi Mao, Revant Teotia, Amrutha Sundar, Sachit Menon, Junfeng Yang, Xin Wang, and Carl Vondrick. Doubly right object recognition: A why prompt for visual rationales. In *CVPR*, pages 2722–2732, 2023. 2

[12] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. In *ICLR*, 2023. 2, 4, 6

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, pages 27730–27744, 2022. 2, 4

[14] Yingzhe Peng, Xu Yang, Haoxuan Ma, Shuo Xu, Chi Zhang, Yucheng Han, and Hanwang Zhang. ICD-LM: configuring vision-language in-context demonstrations by language modeling. *CoRR*, abs/2312.10104, 2023. 2

[15] Sarah M. Pratt, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. *CoRR*, abs/2209.03320, 2022. 2, 7

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 1, 2, 4, 6

[17] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can CLIP benefit vision-and-language tasks? In *ICLR*, 2022. 1

[18] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. FLAVA: A foundational language and vision alignment model. In *CVPR*, pages 15617–15629, 2022. 2

[19] Ming Tao, Bing-Kun Bao, Hao Tang, and Changsheng Xu. GALIP: generative adversarial clips for text-to-image synthesis. In *CVPR*, pages 14214–14223, 2023. 1

[20] Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. Sus-x: Training-free name-only transfer of vision-language models. *CoRR*, abs/2211.16198, 2022. 7

[21] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: semantically-aware object sketching. *ACM Transactions on Graphics*, 41(4):86:1–86:11, 2022. 1

[22] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a foreign language: BEIT pretraining for vision and vision-language tasks. In *CVPR*, pages 19175–19186, 2023. 2

[23] Longhui Wei, Lingxi Xie, Wengang Zhou, Houqiang Li, and Qi Tian. MVP: multimodality-guided visual pre-training. In *ECCV*, pages 337–353, 2022. 1

[24] Jianxin Wu. Balance support vector machines locally using the structural similarity kernel. In *PAKDD*, pages 112–123, 2011. 2, 3

[25] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas M. Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 1

[26] An Yan, Yu Wang, Yiwu Zhong, Chengyu Dong, Zexue He, Yujie Lu, William Yang Wang, Jingbo Shang, and Julian J. McAuley. Learning concise and descriptive attributes for visual recognition. In *ICCV*, pages 3067–3077. IEEE, 2023. 2

[27] Xu Yang, Yongliang Wu, Mingzhuo Yang, Haokun Chen, and Xin Geng. Exploring diverse in-context configurations for image captioning. In *NeurIPS*, 2023. 2

[28] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *CVPR*, pages 19187–19197, 2023. 2

[29] Chao Yi, Zhan De-Chuan, and Ye Han-Jia. Bridge the modality and capacity gaps in vision-language model selection. *CoRR*, abs/2403.13797, 2024. 2

[30] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *CoRR*, 2021. 2

[31] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by CLIP. In *CVPR*, pages 8542–8552, 2022. 1

[32] Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaption of CLIP for few-shot classification. In *ECCV*, pages 493–510, 2022. 1, 6, 7

[33] Guobing Zhou, Jianxin Wu, and song Zhou. A nearest-neighbor-based clustering method. *Journal of Software*, 26 (11):2847–2855, 2015. 2, 3

[34] Yufan Zhou, Ruiyi Zhang, Changyou Chen, Chunyuan Li, Chris Tensmeyer, Tong Yu, Jiuxiang Gu, Jinhui Xu, and Tong Sun. Towards language-free training for text-to-image generation. In *CVPR*, pages 17886–17896, 2022. 1

[35] Orr Zohar, Shih-Cheng Huang, Kuan-Chieh Wang, and Serena Yeung. Lovm: Language-only vision model selection. In *NeurIPS*, 2023. 2