DRUM: Learning Demonstration Retriever for Large MUlti-modal Models

Anonymous ACL submission

Abstract

Recently, large language models (LLMs) have demonstrated impressive capabilities in dealing with new tasks with the help of in-context learning (ICL). In the study of Large Vision-Language Models (LVLMs), when implementing ICL, researchers usually adopt the naive strategies like fixed demonstrations across different samples, or selecting demonstrations directly via a visual-language embedding model. These methods do not guarantee the configured demonstrations fit the need of the LVLMs. To address this issue, we propose a novel framework, demonstration retriever for large multi-modal model (DRUM), which fine-tunes the CLIP embedding model to better meet the LVLM's needs. First, we discuss the retrieval strategies for a visual-language task, assuming an embedding model is given. And we propose to concate the image and text embeddings to enhance the retrieval performance. Second, we propose to re-rank the the embedding model's retrieved demonstrations via the LVLM's feedbacks, and calculate a list-wise ranking loss for training the embedding model. Third, we propose an iterative demonstration mining strategy to improve the training of the embedding model. Through extensive experiments on 3 types of visual-language tasks, 7 benchmark datasets, our DRUM framework is proven to be effective in boosting the LVLM's in-context learning performance via retrieving more proper demonstrations.

1 Introduction

In-context learning (ICL) is a simple yet important learning paradigm that given a few input-output pairs (demonstrations), a model can learn to conduct predictions on a new task it never sees before. ICL is a type of emergent capability observed in large-scale pre-trained models (Wei et al., 2022). It is first observed by GPT-3 (Brown et al., 2020), and draws the attention of the whole community of artificial intelligence. And a large branch of literature have shown that large language models (LLMs) have impressive ICL capabilities across a wide range natural language processing (NLP) tasks. ICL is essential for applications, since it can quickly adapt the large pretrained models to a novel task, or a task with personalized needs, with only a few demonstrations. No fine-tuning is needed and the model need not to be deployed again.

Recently, large vision-language models (LVLMs) are being rapidly developed, and its ICL capabilities are also being investigated (Alayrac et al., 2022). The LVLMs like Flamingo (Alayrac et al., 2022) and Qwen-VL (Bai et al., 2023) have demonstrated impressive ICL capabilities on the visual question answering (VQA), few-shot image classification (ImageCLS), and image captioning (ImageCAP) tasks. However, when implementing ICL for LVLMs, researchers usually adopts the naive strategies like fixed demonstrations or demonstrations ranked by a pre-trained visiion-language embedding model. These strategies are sub-optimal, since they do not incorporate the LVLMs' feedbacks on how these demonstrations help them to improve the responses.

To address the above issue, we now present a novel framework, demonstration retriever for large multi-modal model (DRUM). DRUM is targeted at fine-tuning a pre-trained visual-language embedding model so that it learns to retrieve better demonstrations to meet the LVLM's needs when conduct inference. First, assuming the embedding model is given, DRUM discusses the retrieval strategy for any visual-language tasks. And it proposes to retrieve demonstrations based on the joint embedding of input image, prompt and draft response. Second, DRUM asks the inference LVLM to re-rank the embedding model's retrieved demonstrations via the LVLM feedback. In this work, the LVLM feedback on a demonstration is defined as the conditional log-likelihood of the target response when

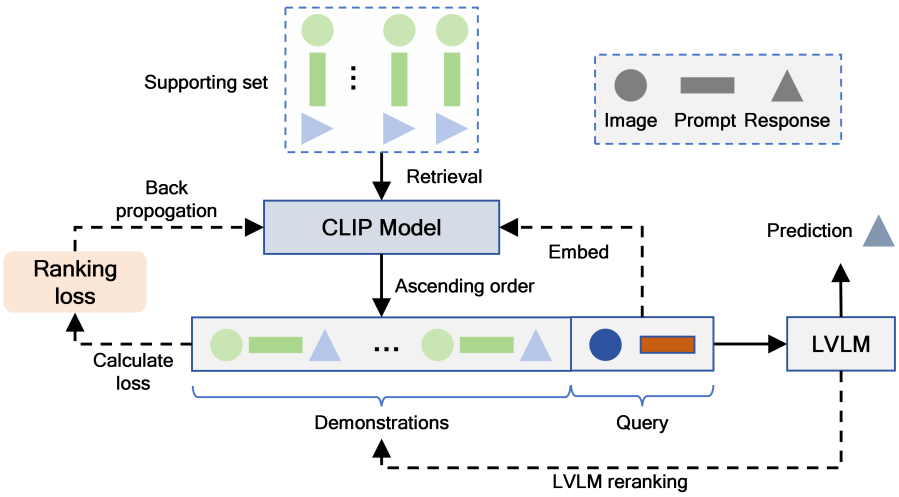


Figure 1: The schematic representation of our DRUM framework. Circles, rectangles, and triangles respectively represent the images, prompts, and responses in the triplet.

the demonstration is added to the prompt. With the LVLM's reranking results, a list-wise ranking loss can be calculated and used as the optimization objective for the embedding model. Third, we propose an iterative demonstration mining strategy which updates the demonstration candidates iteratively, thus improving the training of the embedding model by providing high-quality ranking signals.

We have conducted extensive experiments on 3 types of visual-language tasks, VQA, ImageCLS and ImageCAP, and totally 7 benchmark datasets. The experimental results demonstrate that our DRUM framework is effective in boosting the LVLM's ICL performance. In addition, for commercial LVLMs like GPT-4o, the embedding model fine-tuned by DRUM can also be transferred to them, help them to retrieve better demonstrations.

Our contributions are as follows:

- We propose a novel framework, DRUM, to enhance the ICL capabilities of the LVLMs.
- Extensive experiments have proven that DRUM is effective in boosting the LVLMs' ICL performance on a wide range of vision-language tasks.

2 Related Work

In-Context Learning in NLP. The artificial intelligence community has witnessed significant advancements in the realm of large language models (LLMs) in recent years. As these models and their training corpora expand in scale, LLMs have demonstrated emergent capabilities, such as reasoning, mathematical proficiency, and the ability to follow prompts (Wei et al., 2022). GPT-3 (Brown et al., 2020) was the pioneer in revealing that sufficiently large models can learn to execute new tasks with minimal guidance, a phenomenon termed in-context learning (ICL). Subsequent studies have corroborated the impressive performance of LLMs across various tasks through ICL (Mosbach et al., 2023). The crux of ICL lies in the construction of high-quality in-context demonstration sequences (Li et al., 2023c). However, the bulk of these explorations have concentrated on pure natural language processing tasks and text-centric foundation models, highlighting the necessity to extend this research to encompass other domains.

The research works on in-context learning focus primarily on demonstration sequences. A series of techniques have been investigated, including: (a) utilizing similarity scores to retrieve more relevant in-context examples (Li et al., 2023c), (b) employing machine-generated demonstrations (Li et al., 2023b). The literature has seen a series of studies that reveals certain properties of LLMs when applied to in-context learning. Pan (2023) proposed a decomposition of ICL into the task recognition effect and the task learning effect, and quantified these capabilities of models with varying numbers of shots and scales. Additionally, Lyu et al. (2022) records the "copying effect" phenomenon in LLMs, which is also a type of shortcut inference. Our work complements this line of research by fine-tuning the vision-language embedding model to learn how to retrieve appropriate demonstrations.

LVLM and ICL Inspired by the triumphs of LLMs in natural language processing, the vision-language domain has seen the rise of analogous large vision-language models (LVLMs) (Du et al., 2022). Among these, models such as BLIP2 (Li et al., 2023a), MiniGPT-4 (Zhu et al., 2023), and LLAVA (Liu et al., 2024) are pretrained by aligning image and text data through the use of adapters (Houlsby et al., 2019) to reduce training overhead. While there are several VLMs available, it is worth noting that some of the models are unsuitable for in-context learning, as this capability demands that the LVLM handle inputs that interweave images and text content (Alayrac et al., 2022). Presently, there is scant research on multimodal ICL or ICL for LVLMs, with only a few studies focusing on rudimentary strategies. Yang et al. (2024) examines the impact of ICL on the LVLM's performance in image captioning tasks. Li et al. (2024) analyzes the effects of ICL for LVLMs and proposes various strategies for demonstration retrieval using a pre-trained vision-language embedding model, such as CLIP (Radford et al., 2021). Our work complements this line of research by proposing a novel framework for ICL of the LVLMs.

3 DRUM

We now elaborate on the technical details of our DRUM framework. For the training process of DRUM, we splits the dataset for the current visual-language task into four parts: the support set $\mathcal{D}_{supp}$, the training set $\mathcal{D}_{clip_train}$ used for fine-tuning the image-text embedding model, the validation set $\mathcal{D}_{clip_dev}$ used to validate the embedding model after fine-tuning, and the test set $\mathcal{D}_{test}$ for evaluating the performance of LVLM contextual learning.

3.1 In-context learning

Given a well pre-trained Large Vision-Language Model (LVLM) (denoted as $\mathcal{M}$) e.g., Flamingo (Alayrac et al., 2022), one can use it directly to solve a VL task like VQA with in-context learning, and no fine-tuning is required. To achieve this, we need to prepare a multi-modal in-context sequence

$$\mathcal{S} = \{z_1, ..., z_n\}, \tag{1}$$

where $\mathcal{S}$ consists of n-shot $z_i = (\text{image}_i, \text{prompt}_i, \text{response}_i)$ tuples. Then we concatenate $\mathcal{S}$ to the left of the test sample $x_{test} = (\text{image}_{test}, \text{prompt}_{test})$, and feed into the LVLM for generating the corresponding response:

$$\text{response}_{test} = \{\hat{a}_1, ..., \hat{a}_{T_A}\}, \tag{2}$$

where the t-th ($t \leq T_A$) token $\hat{a}_t$ is sampled from the probability distribution $\mathbf{P}(\cdot)$ over the vocabulary calculated by the LVLM $\mathcal{M}$:

$$\mathbf{P}(\hat{a}_t | \mathcal{S}, x_{test}, \hat{a}_{1:t-1}). \tag{3}$$

3.2 Strategies for sample embedding

Different from retrieving via only images or texts (Li et al., 2024), we retrieve the demonstrations via the concatenation of image embeddings and text embeddings generated by the CLIP model (Radford et al., 2021). We first generate a draft response $\text{response}_{test}^{pred,1}$ to the test sample x_{test} with the help of strategy SIT-IP, and then compare the semantic similarity between (image_{test}, prompt_{test}, $\text{response}_{test}^{pred,1}$) and ($\text{image}_i$, prompt_i, response_i). We denote this strategy as retrieving via similar image prompt and draft response (SIT-IPDR).

We will use SIT-IPDR as the default sample embedding strategy in our experiments. More strategies are presented in Appendix C for completeness. And we will use experiments (Section 4.6) to validate this choice.

3.3 Pilot experiments and motivations

The previous sub-section assumes that an embedding model $\mathcal{E}$ is ready to use for any given VL task which can transform the image and text inputs to embedding vectors. Intuitively, one can directly utilize the pre-trained CLIP models (Radford et al., 2021) to initialize $\mathcal{E}$ and obtain the test sample's image or text embeddings, and conduct search for similar demonstrations based on these embeddings. However, we now conduct a pilot experiment to demonstrate that the original open-sourced CLIP models may not be effective in retrieving demonstrations for a LVLM.

For a task at hand, we first use the CLIP model (base) to construct the demostration vector database on $\mathcal{D}_{supp}$. For a sample $x_q = (\text{image}_q, \text{prompt}_q, \text{response}_q)$ from $\mathcal{D}_{clip_dev}$, the CLIP model will embedd it and retrieve $n = 16$ demonstration candidates $\{z_j\}_{i=1}^n$. These candidates are ranked based on the embeddings' similarity scores:

$$r_0(z_j) = \text{Ranking}(\text{sim}(x_q, z_j) | \{z_j\}_{i=1}^n), \tag{4}$$

where $\text{sim}(x_q, z_j)$ denotes the embedding vectors' cosine similarity when CLIP is the embedding

model, and Ranking is the ranking function (in ascending order).

Note that the intended effect of demonstrations on LVLM is to help the LVLM generate better responses and achieve performance boost. In other words, demonstrations are expected to enhance the likelihood of the ground-truth answer being generated by the LVLM. Thus, it is appropriate for the LVLM to evaluate and rank the demonstration candidates via the log-likelihood function. Formally, the LVLM's ranking of the candidate demonstrations are given by:

$$\begin{aligned} r(z_j) &= \text{Ranking}(s(z_j)|\{s(z_j)\}_{i=1}^n) \\ s(z_j) &= \text{LLH}(\text{response}_q|z_j, \text{image}_q, \text{prompt}_q), \end{aligned} \tag{5}$$

where $\text{LLH}(\cdot|\cdot)$ is the LVLM's conditional log-likelihood function. $s(z_j)$ represents the ground-truth response$_q$'s log-likelihood conditioned on the demonstration candidate z_j and the querying input image$_q$ and prompt$_q$. $s(z_j)$ indicates the importance of z_j for the LVLM to encode the querying sample and generate the ground-truth response. The more important z_j is for the LVLM, the higher $s(z_j)$ will be, and the larger $r(z_j)$ will be.

Since we have two rankings for the same set of demonstration candidates, we can calculate the correlation between these two rankings:

$$\text{corr}_q = \text{Spearman}(\{r(z_j)\}_{i=1}^n, \{r_0(z_j)\}_{i=1}^n), \tag{6}$$

where Spearman is the Spearman rank corelation coefficient (Dodge, 2008). The average correlation score is given by:

$$\text{corr}_{avg} = \frac{\sum_{x_q \in \mathcal{D}_{clip_dev}} \text{corr}_q}{\|\mathcal{D}_{clip_dev}\|}. \tag{7}$$

The average correlation score is calculated on the VizWiz (Gurari et al., 2018), Flicker30K (Plummer et al., 2015) and Hateful-Memes (Kiela et al., 2020) tasks, with the LVLM being the Deepseek-VL2 (tiny). The results are presented in Table 1. From Table 1, we can see that the CLIP model's rankings and the LVLM's rankings actually have very low correlations. For example, the correlation score on the VizWiz task is negative, showing significant discrepancy between the CLIP model's retrieved candidates and the LVLM's needs.

The above observations are consistent with the claims in the previous works (Li et al., 2023c; Rubin et al., 2021): demonstrations retrieved by an open-sourced embedding model may not benefit the most for the LVLM. Thus, it is natural to consider fine-tuning the embedding model $\mathcal{E}$ so that its retrieved demonstrations better fit the LVLM and help to elicit better responses from the LVLM.

Task	corr_{avg}
VizWiz	-0.16
Flicker30K	0.11
Hateful-Memes	0.21

Table 1: The average correlation scores between the CLIP model's rankings and the LVLM's rankings, on the $\mathcal{D}_{clip_dev}$ sets of the VizWiz , Flicker30K and Hateful-Memes tasks.

3.4 Demonstration retriever training

We now elaborate on the core of our DRUM framework, the training approach for the demonstration retriever. Different from Rubin et al. (2021) which design task-specific training signals for several tasks separately, we propose to cast the retriever's training signals into a list-wise ranking loss based on the LVLM's feedback. Then we introduce a training framework in which the retriever iteratively mines high-quality demonstration candidates with the help of the LVLM and learn to rank them in turn. The whole workflow are shown in Algorithm 1. And we now introduce the list-wise ranking training and iterative mining strategy for the demonstration retrievers as follows.

Loss function for the demonstration retriever The objective of training the demonstration retriever is to make the CLIP's ranking (from Equation 4) and the LVLM's ranking (from Equation 5) more consistent. With the demonstration candidates' ranks $\{r(z_j)\}_{i=1}^n$ from the LVLM's feedback, we propose to use the following loss function to inject the ranking signal into the demonstration retriever $\mathcal{E}$:

$$\mathcal{L}_r = \sum_{1 \le i,j \le n, i \ne j} m(i,j) * g(i,j), \tag{8}$$

where $m(i,j)$ is given by

$$m(i,j) = \max(0, \frac{1}{\sqrt{r(z_j)}} - \frac{1}{\sqrt{r(z_i)}}), \tag{9}$$

and $g(i,j)$ is given by:

$$g(i,j) = \log(1 + e^{(\text{sim}(x_q,z_j) - \text{sim}(x_q,z_i))}), \tag{10}$$

We now provide intuitive explanations for the above loss function. For those z_i and z_j where

$r(z_j) \leq r(z_i)$, $\mathcal{L}_r$ will draw $\text{sim}(x_q, z_i)$ up and optimize the retriever towards $\text{sim}(x_q, z_i) > \text{sim}(x_q, z_j)$. For z_i and z_j where $r(z_i) \geq r(z_j)$, this pair will be discarded by the loss function. Additionally, $m(i, j)$ adjusts the weight for each pair of demonstrations, conveying list-wise ranking information into $\mathcal{L}_r$. When the ranks of z_i and z_j are close, e.g., $r(z_i) = 2$ and $r(z_j) = 1$, $m(i, j) \approx 0.292$. In comparison, when z_i has a much higher rank than z_j, e.g., $r(z_i) = 15$ and $r(z_j) = 1$, $m(i, j)$ will be 0.742, larger than 0.292. Thus, when z_i has a much higher rank than z_j, w will be a high weight, and asks $\mathcal{L}_r$ to strongly draw $\text{sim}(x_q, z_i)$ up and away from $\text{sim}(x_q, z_j)$. Since we optimize the retriever on demonstration pairs under different $m(i, j)$, $\mathcal{L}_r$ can help our DRUM method fully incorporate candidates' list-wise ranking signals and learn to retrieve those high-quality and helpful demonstrations.

3.5 Iterative Demonstration Candidate Mining

The selection of demonstration candidates can be a key factor for retriever's training. It is infeasible and possibly harmful to take the entire training set as candidates. In addition, once the embedding model is fine-tuned, it no longer matches the supporting samples' vectors in the vector database. To strike a balance between training time cost and quality, we adapt an iterative strategy to update candidates (Li et al., 2023c). Specifically, we iteratively train the retriever and use it to select candidates in turn. At each iteration, we update each supporting example x_q's candidates as:

$$Z^* = \text{topK}(\{\text{sim}(x_q, z) | z \in D\}, n), \qquad (11)$$

where D is the task's supporting set, n is the number of candidates retrieved. Then we will use the LVLM $\mathcal{M}$ to score and rerank Z^*, and calculate the list-wise ranking loss according to Eq 8. Before the first iteration, the retriever is exactly the pre-trained embedding model, so we initialize candidates based on the similarity calculated with the pretrained embedding model. In summary, Algorithm 1 shows the DRUM's overall training procedure.

Embedding Model Validation The optimization objective of model $\mathcal{E}$ is to minimize the discrepancy between the ranking of retrieved example vectors and the ranking assigned by the large-scale model $\mathcal{M}$ to these examples. Therefore, to validate the training effectiveness of model $\mathcal{E}$, and to select the model checkpoints during training, we follow Equation 7 to compute the average correlation coefficient corr_{avg} of rankings using dataset $\mathcal{D}_{clip_dev}$.

Algorithm 1: DRUM's demonstration ranking training

Input: Embedding model $\mathcal{E}$, large vision-language model $\mathcal{M}$, number of training iterations N_1, number of training steps in each iteration N_2, number of retrieved candidates n

Output: A fine-tuned embedding model $\mathcal{E}$.

Data: support set $\mathcal{D}_{supp}$, model $\mathcal{E}$'s training set $\mathcal{D}_{clip_train}$, model $\mathcal{E}$'s validation set $\mathcal{D}_{clip_dev}$, test set for the LVLM $\mathcal{D}_{test}$;

```
training iteration index i ← 0;
while i < N_1 do
    Embed each training example with E;
    Retrieve n candidates of each training example;
    training step index j ← 0;
    while j < N_2 do
        Sample an querying example x_q from D, and obtain its candidates {z_k}_{k=1}^n;
        Re-rank {z_k}_{k=1}^n by M using Eq 5;
        Calculate L_r using Eq 8;
        Update E;
        j ← j + 1;
    i ← i + 1;
```

4 Experiments

4.1 Datasets

We conduct experiments on three benchmark visual question-answering (VQA) tasks, two image classification (ImageCLS) tasks, and two image captioning (ImageCAP) tasks: VQAv2 (Goyal et al., 2017), VizWiz (Gurari et al., 2018), OK-VQA (Marino et al., 2019), Flowers102 (Nilsback and Zisserman, 2008), Hateful-Memes (Kiela et al., 2020), Flickr30K (Plummer et al., 2015), NoCaps (Agrawal et al., 2019). The introduction and dataset splits of each dataset are detailed in Appendix A.

4.2 Evaluation metrics

Metric for the VQA tasks We follow Alayrac et al. (2022) to use accuracy as the evaluation met-

Retrieval	VQA			ImageCLS		ImageCap	
Methods	VQAv2	VizWiz	OK-VQA	Flowers102	Hateful-Memes	Flicker30K	NoCaps
Null	56.1	24.6	42.3	14.6	55.4	27.7	28.6
Random	66.3	43.2	56.3	31.5	61.3	37.5	39.4
Fixed	66.4	42.6	57.9	32.3	61.1	38.1	39.9
BM25	67.8	34.5	55.8	25.7	56.7	33.9	34.3
Dino	69.5	46.8	59.9	35.7	63.2	39.0	38.8
BGE	68.9	38.7	61.2	26.6	56.8	34.3	35.1
CLIP	69.7	58.2	63.4	36.5	65.4	39.2	40.7
EPR	70.4	61.3	64.9	38.5	66.9	40.3	41.3
DRUM	**73.7**	**64.6**	**67.8**	**40.9**	**70.9**	**41.5**	**43.5**

Table 2: Results on 7 benchmark tasks. Due to randomness, the results from Random, Fixed, EPR, UDR and DRUM are the average scores across five different runs under different random seeds. Best scores are bolded.

ric for VQA task:

$$\mathrm{Acc}_{a_i} = \min(1, \frac{3 \times \sum_{k\in[0,9]} \mathrm{match}(a_i, g_k)}{10}), \tag{12}$$

where a_i denotes the predicted answer of the LVLM, g_k denotes the k-th ground true answer, and the match() function indicates whether two answers match, if they match, the result is 1, otherwise it is 0.

Metric for the image classification tasks For the visual classification tasks, we report the accuracy score.

Metric for the image captioning tasks For evaluation on the image captioning tasks, we report the ROUGE-L score (Lin, 2004).

4.3 Implementation details

Computing infrastures All experiments are conducted on the RTX 4090 GPUs.

LVLM models We employ the Deepseek-VL2 Tiny (Wu et al., 2024) model (3B) as the LVLM to evaluate our DRUM method.

Decoding After receiving the input images and text prompts, the predictions are generated using the language modeling head (LM head) of the LVLM. No other prediction layers outputting numerical or categorical results are installed on the LVLM backbone. For decoding during inference, we use beam search with beam size 3.

ICL Setup for the LVLM Model $\mathcal{M}$ The number of demonstrations obtained for each test sample is set by default to $n = 4$ in this work. The ablation studies also investigate different values of n. After retrieving the examples, model $\mathcal{M}$ concatenates the demonstration sequence in ascending order of similarity scores to the left side of the test sample input. This means that the higher the similarity score an retrieved example has, the closer it is placed to the test sample input. The prompt templates for the LVLM are presented in Appendix B.

Settings for embedding and retrieval This work defaults to using the base-sized CLIP model[1] for image-text embedding. The default retrieval strategy adopted in this work is the SIT-IPDR approach detailed in Section 3.2. Under this strategy, the vector representation of both demonstrations samples and test samples is obtained by concatenating the image vector and the text vector. This work utilizes the Faiss toolkit (Douze et al., 2024) for constructing the vector database and for efficient vector retrieval.

Settings for fine-tuning the embedding model We implements the fine-tuning process of the embedding model $\mathcal{E}$ based on the Huggingface Transformers(Wolf et al., 2020) code library. The number of training epochs N_1 for the embedding model is set to 50, with $N_2 = 100$ steps per epoch. During the fine-tuning of the embedding model, the number of recall examples n is set to 32. For model optimization, we use AdamW (Loshchilov and Hutter, 2019), with a learning rate of 1e-5 and a warmup of 50 steps at the beginning of the model fine-tuning. Other hyperparameters remain consistent with the Transformers code library. After each epoch, the embedding model $\mathcal{E}$ is evaluated according to Equation 7. The fine-tuning employs an early stopping strategy with a maximum patience of 10, meaning that if the evaluation metric corr_{avg} does not improve for 10 consecutive epochs, the training will be stopped.

4.4 Baseline methods

With the same inference LVLM, we compare our DRUM method with existing methods for demon-

[1] https://huggingface.co/openai/clip-vit-base-patch32

stration retrieval by the downstream ICL performance, including: (a) Null, which is not to use any demonstrations. (b) Random, randomly sampling demonstrations from the supporting set. (c) BM25, a prevailing sparse retriever widely used in the literature (Chen et al., 2017). (d) DINO, which is to retrieve demonstrations using the image embedding provided by the DINO model (Caron et al., 2021). (e) BGE, which is to retrieve demonstrations using the text embedding provided by the BGE model (Chen et al., 2024). (f) CLIP, which is to retrieve demonstrations using the image-text embedding provided by the CLIP model (Caron et al., 2021). (g) EPR (Rubin et al., 2021), which builds upon the aforementioned CLIP approach by conducting LVLM feedback evaluation for each example, then transforming the task of re-ranking demonstrations into a classification task, leading to the training of a classifier for evaluating these demonstrations.

4.5 Main Results

We report the performance of different methods on the seven benchmark VL tasks in Table 2. We can see that: (a) DRUM outperforms the baselines with clear margins on most tasks, which shows our method's best demonstration retrieval ability on a wide range of VL tasks. (b) Specially, compared with EPR, DRUM has better overall performance and this shows the effectiveness of our training method. Meanwhile, compared with CLIP, the embedding model which is directly initialized with CLIP-base, DRUM has clear advantages. This straightly demonstrates that our proposed training framework can help DRUM incorporate LVLM's feedback through the DRUM's fine-tuning procedure and retrieve more beneficial demonstrations. The experimental results also reveal that the random baseline achieves the worst performance in most tasks. This phenomenon is intuitive: pairing the current query with irrelevant demonstrations is unhelpful, and sometimes could lead the model to the wrong directions.

4.6 Further analysis

Ablation Study To evaluate the effect of our DRUM's each component, we consider the following variant of DRUM: (a) DRUM-1, which substitute Eq 9 to $m(i,j) = \max(0, \frac{1}{r(z_j)} - \frac{1}{r(z_i)})$. (b) DRUM-2, which substitute Eq 9 to $m(i,j) = \max(0, r(z_i) - r(z_j))$. (c) DRUM-3 removes the weight $m(i,j)$ from Eq 8. (d) DRUM-4, which do not conduct iterative demonstration candidate mining. The results are reported in Table 3.

Method	VizWiz	Hateful-Memes	Flicker30K
DRUM	**64.6**	**70.6**	**41.5**
DRUM-1	64.0	68.7	40.8
DRUM-2	63.9	69.3	40.7
DRUM-3	63.8	68.4	40.1
DRUM-4	63.4	68.2	39.9

Table 3: Results of the ablation study on DRUM's training strategy.

Strategy	VizWiz	Hateful-Memes	Flicker30K
SIT-IPDR	**64.6**	**70.6**	**41.5**
SIT-IP	63.1	68.6	40.7
ST-PDR	61.5	66.2	39.4
ST-P	62.7	67.0	34.7
SI	62.8	68.3	40.8

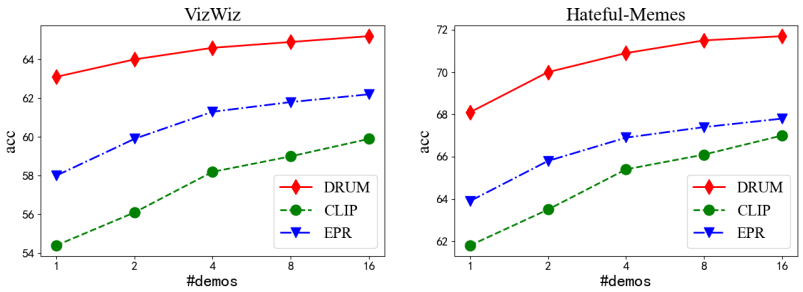
Table 4: Results of the ablation study on the demonstration retrieval strategy.

LVLM $\mathcal{M}$	$\mathcal{E}$	VizWiz	Hateful-Memes	Flicker30K
GPT-4o	CLIP	72.1	76.9	41.1
	EPR	75.6	79.0	42.9
	DRUM	**77.2**	**81.6**	**45.2**
Claude 3 Opus	CLIP	71.5	76.2	38.2
	EPR	73.3	78.3	41.6
	DRUM	**76.1**	**80.2**	**43.4**

Table 5: Experiments on the transfer learning capabilities of DRUM. We using the fine-tuned model $\mathcal{E}$ to retrieve demonstrations for GPT-4o and Claude 3 Opus. $\mathcal{E}$ being CLIP means no fine-tuning is conducted. $\mathcal{E}$ being CLIP + EPR means fine-tuning with the EPR method is conducted. $\mathcal{E}$ being CLIP + DRUM means fine-tuning with the DRUM method is conducted.

The experimental results show that: (a) The comparison between DRUM-1 and DRUM demonstrates the

Ablation on the retrieval strategy This work uses the SIT-IPDR strategy for example retrieval in the main experiment (Table 2). To demonstrate the rationality of the DRUM setup and this strategy, we conduct an ablation study on the demonstration retrieval strategy. Table 4 reports the performance of the DRUM method using SIT-IP, ST-PDR, ST-P, and SI strategies. The experimental results show: (a) The SIT-IPDR strategy outperforms other strategies. This strategy combines image and text information for demonstration retrieval, utilizing the maximum amount of semantic information available in the test sample, thus enabling it to recall the most relevant demonstrations. (b) Retrieving examples based only on the prompt text content (ST-P)



(a) VizWiz (b) Hateful-Memes

Figure 2: The effects of the number of demonstrations on DRUM, EPR, and CLIP.

performs poorly on image classification tasks and image caption generation tasks. The primary reason for this phenomenon is that these types of tasks involve prompts that contain generic task instructions without directly related semantic information. However, by combining the prompt text with the draft response text (ST-PDR), there is a significant improvement in performance. This result shows that the draft response can effectively supplement the semantic information needed for example retrieval.

Transferability across Different LMs Note that during the fine-tuning of the embedding model $\mathcal{E}$ using the DRUM method, the LVLM model $\mathcal{M}$ needs to re-rank the recalled examples based on conditional likelihood function values. Given that different LVLM models have similar training mechanisms and are pre-trained on large amounts of internet data, their internal mechanisms and cognition share similarities. In this part of the experiment, we will use the embedding model $\mathcal{E}$, fine-tuned with feedback from the Deeoseek-VL2 model, for example recall with GPT-4o or Claude 3 Opus models. The experimental results are presented in Table 5.

According to Table 5, the embedding model, fine-tuned with feedback signals from the Deeoseek-VL2 model, is able to recall higher-quality examples, effectively enhancing the performance of powerful commercial LVLM models like GPT-4o or Claude 3 Opus in tasks such as VQA (Visual Question Answering), image classification, and image caption generation. This experiment demonstrates the practical significance of the DRUM method: by fine-tuning an example recall model with feedback from open-source LVLM models, and then applying this example recall model to the contextual learning of commercial LVLM models.

Impact of demonstration quantity In the main experiments (Table 2), we set n to 4. We now compare DRUM with CLIP and EPR under different amounts of demonstrations, and the experimental results are reported in Figure 2.

We can see that DRUM outperforms baselines consistently across varying amounts of demonstrations. Meanwhile, we can draw two conclusions from the results: (a) The number of demonstrations has a greater impact on the generation task, VizWiz, than the classification task, Hateful-Memes. Specifically, as the number of demonstrations increases, VizWiz' performance gets significant improvements while Hateful-Memes' has slight improvements. (b) The quality of demonstrations can be more important than their quantity. Specifically, DRUM with one or two demonstrations still outperforms EPR with 4 demonstrations. These observations again reflect the strong demonstration retrieval ability of DRUM.

5 Conclusion

In this paper, we propose DRUM, a unified approach of demonstration retrieval for large vision-language models. To train DRUM, we cast the LVLM's feedback on a demonstration to a unified list-wise ranking formulation, and propose the ranking training framework with an iterative mining strategy to find high-quality candidates. Experiments on three visual question answering tasks, two visual recognition tasks and two image captioning tasks show that UDR significantly outperforms the baseline demonstration retrieval methods. Further analysis show the effectiveness of each proposed components of the DRUM, and the strong transferability of DRUM across different LVLMs (3B to 175B), unseen datasets, and varying demonstration quantities.

Limitations

We showed that our proposed method can improve the performance of in-context learning on diverse vision-language tasks and different large vision-language models. However, we acknowledge the following limitations: (a) the number of experimented open-sourced LVLMs is limited due to limited computation resources. (b) Other vision-language tasks, like visual information extraction, were also not considered. But our framework can be easily transferred to other LVLM backbone architectures and different types of tasks. It would be of interest to investigate if the superiority of our method holds for other large-scaled backbone models and other types of tasks. And we will explore it in future work.

Ethics Statement

The finding and proposed method aims to improve the in-context learning in terms of better task performances. The used datasets are widely used in previous work and, to our knowledge, do not have any attached privacy or ethical issues. In this work, we have experimented with Deepseek-VL2, a modern large vision language model series. As with all LVLMs, Deepseek-VL2's potential outputs cannot be predicted in advance, and the model may in some instances produce inaccurate, biased or other objectionable responses to user prompts. However, this work's intent is to conduct research on different in-context learning methods for LVLMs, not building applications to general users. In the future, we would like to conduct further tests to see how our method affects the safety aspects of LVLMs.

References

Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. 2019. Nocaps: Novel object captioning at scale. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8948–8957.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2318–2335.

Y Dodge. 2008. *The concise encyclopedia of statistics*. Springer New York.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. 2022. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and

Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. *Advances in neural information processing systems*, 33:2611–2624.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. 2020. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 128(7):1956–1981.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Li Li, Jiawei Peng, Huiyi Chen, Chongyang Gao, and Xu Yang. 2024. How to configure good in-context sequence for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26710–26720.

Rui Li, Guoyin Wang, and Jiwei Li. 2023b. Are human-generated demonstrations necessary for in-context learning? *arXiv preprint arXiv:2309.14681*.

Xiaonan Li, Kai Lv, Hang Yan, Tianya Lin, Wei Zhu, Yuan Ni, Guo Tong Xie, Xiaoling Wang, and Xipeng Qiu. 2023c. Unified demonstration retriever for in-context learning. *ArXiv*, abs/2305.04320.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dollár. 2014. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Xinxi Lyu, Sewon Min, Iz Beltagy, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. Z-icl: Zero-shot in-context learning with pseudo-demonstrations. *arXiv preprint arXiv:2212.09865*.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *arXiv preprint arXiv:2305.16938*.

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.

Jane Pan. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. Master's thesis, Princeton University.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

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.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Xu Yang, Yongliang Wu, Mingzhuo Yang, Haokun Chen, and Xin Geng. 2024. Exploring diverse in-context configurations for image captioning. *Advances in Neural Information Processing Systems*, 36.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

A Datasets

The DRUM method is evaluated on three benchmark visual-question answering (VQA) datasets, two benchmark image captioning (ImageCap) datasets, and two image classification (ImageCLS) tasks. The specific VQA datasets are as follows:

- **VQAv2** (Goyal et al., 2017). This dataset uses images from the MSCOCO dataset (Lin et al., 2014), with textual questions manually crafted by annotators to ensure that each question requires visual information to answer.

- **VizWiz** (Gurari et al., 2018). This dataset contains low-resolution images, and some questions are unanswerable based on the images. It is designed to evaluate whether models can discern answerable questions and avoid hallucination or overconfident responses.

- **OK-VQA** (Marino et al., 2019). This dataset requires models to integrate visual information, textual questions, and external world knowledge to generate answers, posing significant challenges.

The ImageCap datasets include:

- **Flickr30K** (Plummer et al., 2015). This dataset contains images from the Flickr community[2], with each image annotated by crowdworkers to provide five reference captions.

- **NoCaps** (Agrawal et al., 2019). This dataset uses images from the validation and test sets of the Open Images dataset (Kuznetsova et al., 2020), with human-annotated captions.

The ImageCLS tasks employ the following datasets:

- **Flowers102** (Nilsback and Zisserman, 2008). This dataset requires classifying input images into one of 102 common flower categories in the UK.

- **Hateful-Memes** (Kiela et al., 2020). This dataset collects internet memes and categorizes them into "hateful" or "non-hateful" classes.

For each dataset, the original training/validation/test splits were randomly reorganized to form the support set $\mathcal{D}_{supp}$ required by the DRUM workflow, the training set $\mathcal{D}_{clip_train}$ and validation set $\mathcal{D}_{clip_dev}$ for fine-tuning the example retrieval model, and the test set $\mathcal{D}_{test}$ for evaluating the in-context learning performance of the language model. The statistics of each task's dataset are summarized in Table 6.

[2] https://www.flickr.com/

B Prompt templates

Prompt template for the VQA task If we do not use any demonstrations, the prompt template for the VQA task is:

```
<image>
Question: [question]
Instruction: answer with a short phrase.
Answer:
```

in which <image> is the placeholder for the input image, [question] is the input question.

The prompt template for VQA with a group of demonstrations is:

```
<demo_image>
Question: [demo_question]
Answer: [demo_answer]

<demo_image>
Question: [demo_question]
Answer: [demo_answer]

You will be engaged in a two-phase task.
    Phase 1: Absorb the information
    from a series of image-text pairs.
    Phase 2: Use that context, combined
    with an upcoming image and your own
    database of knowledge, to accurately
    answer a subsequent question.
<image>
Question: [question]
Instruction: answer with a short phrase.
Answer:
```

in which <demo_image> is the placeholder for the image in the demonstration sample, [demo_question] is the demonstration question, and [demo_answer] is the corresponding ground-truth answer.

Prompt template for the image captioning task If we do not use any demonstrations, the prompt template for the image captioning task is:

```
<image>
Instruction: write a concise caption for
    the image.
Response:
```

in which <image> is the placeholder for the input image.

The prompt template for VQA with a group of demonstrations is:

Table 6: The vision-language tasks used in the experiments.

Dataset	$\lvert\mathcal{D}_{supp}\rvert$	$\lvert\mathcal{D}_{clip_train}\rvert$	$\lvert\mathcal{D}_{clip_dev}\rvert$	$\lvert\mathcal{D}_{test}\rvert$	Labels	Type	Metric
VQAv2	180k	10k	10k	14k	-	VQA	acc
VizWiz	2.0k	1.0k	0.5k	0.8k	-	VQA	acc
OK-VQA	2.0k	1.0k	0.5k	1.6k	-	VQA	acc
Flickr30K	20.0k	5.0k	1.0k	5.8k	-	ImageCap	rouge-l-ic
NoCaps	2.0k	1.0k	0.5k	1.0k	-	ImageCap	rouge-l-ic
Flowers102	4.0k	1.0k	1.0k	1.2k	102	ImageCLS	acc
Hateful-Memes	6.0k	2.0k	1.5	3.0k	2	ImageCLS	acc

```
<demo_image>
Response: [demo_caption]

<demo_image>
Response: [demo_caption]

You will be engaged in a two-phase task.
     Phase 1: Absorb the information
    from a series of image-text pairs.
    Phase 2: Use that context, combined
    with an upcoming image and your own
    database of knowledge, to accurately
     provide a caption for the following
     image.
<image>
Instruction: write a concise caption for
     the image.
Response:
```

in which <demo_image> is the placeholder for the image in the demonstration sample, [demo_caption] is the ground-truth caption.

Prompt template for the image classification task If we do not use any demonstrations, the prompt template for the image classification task is:

```
<image>
Instruction: assign one of the following
     labels to the input image.
[label_list]
Response:
```

in which <image> is the placeholder for the input image, and the [label_list] is the collection of label names specified in the given classification task.

The prompt template for VQA with a group of demonstrations is:

```
<demo_image>
Response: [demo_label]

<demo_image>
Response: [demo_label]

You will be engaged in a two-phase task.
     Phase 1: Absorb the information
    from a series of image-text pairs.
    Phase 2: Use that context, combined
    with an upcoming image and your own
    database of knowledge, to accurately
     assign a label from the provided
    label list for the following image.
<image>
Instruction: assign one of the following
     labels to the input image.
[label_list]
Response:
```

in which <demo_image> is the placeholder for the image in the demonstration sample, [demo_label] is the ground-truth caption.

C Sample embedding strategies

How to transform a input vision-language sample to an embedding vector is essential for demonstration retrieval. Now, we summarize a series of specific retrieval strategies mentioned in the literature (Li et al., 2024) and new ones proposed in our work.

Random sampling (RS) This strategy simply obeys the uniform distribution to randomly sample n-shot triplets from $\mathcal{D}$ to form the in-context sequence S.

Retrieving via similar image (SI) This method retrieve n images from $\mathcal{D}$ which are most similar to the querying image and then use the corresponding triplets of these retrieved images as the demonstrations. For example, given the test sample x_{test} = (image$_{test}$, prompt$_{test}$), suppose the i-th image image$_i$ is similar to image$_{test}$, then the whole i-th triplet z_i = (image$_i$, prompt$_i$, response$_i$) will be used as one demonstration. Here we assume we have access to an high-quality image embedding model at hand, which can transform each image to a separate vector in the semantic space in which the similarity between two vectors reflect their similarity in contents.

Retrieving via similar texts (ST). Besides retrieving via images, we can also retrieve n triplets which contain the most similar text contents to the querying sample, where the embeddings of these texts are used to calculate the cosine similarity. Here

we assume we have access to an high-quality text embedding model at hand, which can transform a piece of text to a separate vector in the semantic space in which the similarity between two vectors reflect their similarity in contents. We consider three kinds of texts:

- **Retrieving via similar prompts (ST-Q).** We use the prompts in the supporting set as the contents to build the vector database, and use the prompt of the test sample as the input text for retrieving, i.e., comparing the similarity between prompt_{test} and prompt_i.

- **Retrieving via similar prompts & draft response (ST-PDR).** This strategy, since the ground truth answer is not available during inference, we can not retrieve demonstrations with the querying sample's answer. However, note that the LVLM itself can generate a draft response by only generating conditioned onthe prompt or using strategy ST-Q. Thus, we first generate a draft response $\text{response}_{test}^{pred,1}$ to the test sample x_{test}, and then compare the semantic similarity between (prompt_{test}, $\text{response}_{test}^{pred,1}$) and ($\text{prompt}_i$, response_i). Note that generating the draft response $\text{response}_{test}^{pred,1}$ introduces additional latency for the whole system. To ensure small latency, we ask the model to generate at most 2 tokens.

Retrieving via Similar image-texts (SIT). Besides retrieving via only images or texts, we can also retrieve the demonstrations via the concatenation of image embeddings and text embeddings. Note that (Li et al., 2024) neglect this group of strategy. Since the CLIP model can generate two vectors for the text and image contents separately, these two vectors will be concatenated.

Thus, similar to the previous strategies based on text input, we can have the following strategy:

- **Retrieving via similar image and prompts (SIT-IP).** We concatenate the querying image embedding and prompt embedding for retrieval on a vector database, which are constructed by concatenating supporting samples' image embeddings and prompt embeddings.

- **Retrieving via similar image prompt and draft response (SIT-IPDR).** This strategy is introduced Section 3.2 in the main contents.