Supplementary - ProAPO: Progressively Automatic Prompt Optimization for Visual Classification

Xiangyan Qu[12] Gaopeng Gou[12†] Jiamin Zhuang[12] Jing Yu[3]
Kun Song[4] Qihao Wang[12] Yili Li[12] Gang Xiong[12]

[1]Institute of Information Engineering, Chinese Academy of Sciences [2]School of Cyber Security, Chinese Academy of Sciences
[3]School of Information Engineering, Minzu University of China [4]University of Science and Technology Beijing

{quxiangyan, gougaopeng, zhuangjiamin}@iie.ac.cn, jing.yu@muc.edu.cn
songkun@xs.ustb.edu.cn, wangqihao22@mails.ucas.ac.cn, {liyili, xionggang}@iie.ac.cn

We first describe detailed processes of building the prompt library and sampling strategies in our method:

- Appendix A: Details of building prompt library.
- Appendix B: Details of prompt sampling strategy.
- Appendix C: Details of group sampling strategy.

Then, we show more experiments to show the effectiveness of our ProAPO:

- Appendix D: More implementation details.
- Appendix E: Results on different backbones.
- Appendix F: More comparisons with SOTA methods.
- Appendix G.1: Ablation of progressive optimization.
- Appendix G.2: More ablation of operators.
- Appendix G.3: More ablation of group sampling.
- Appendix G.4: Ablation of cost computation.
- Appendix H.1: Effect of shot numbers.
- Appendix H.2: Effect of scalar α in score function.
- Appendix H.3: Effect of sampled numbers in prompt sampling.
- Appendix H.4: Effect of quality of prompt library.
- Appendix I: More qualitative results.

We also provide detailed results for experiments appearing in the main paper:

- Appendix J.1: Results of transfering to adapter-based methods.
- Appendix J.2: Results of transferring to different backbones.
- Appendix K.1: Analysis of single vs ensemble prompts.
- Appendix K.2: Improvement by iterative optimization.
- Appendix L.1: Ablation of edit and evolution operators.
- Appendix L.2: Ablation of two sampling strategies.
- Appendix L.3: Ablation of different score functions.

†Corresponding author

A. Details of Building Prompt Library

A.1. Details of Building Template Library

The template library aims to collect a set of templates that provide task-specific contextual information, which can address issues of semantic ambiguity caused by class names. It contains processes for collecting templates, generating dataset domains, and adding dataset domains to templates.

Collecting templates. We utilize two ways to collect templates. First, pre-defined templates, such as Template-80 [29], FILIP-8 [41], and DEFILIP-6 [6] can be used. Second, similar to PN [20], we query LLMs to create diverse templates by the following prompt:

> "Hi, ChatGPT! I would like your help to prompt for image classification using CLIP. As a human-level prompt engineer, your task is to create a set of Templates like the following for visual classification. For example:
>
> a photo of a {}."

Generating dataset domain by LLMs. Inspired by previous description-based methods [22, 30], we query LLMs to generate dataset domain information to provide task-specific context. For this purpose, we use the prompt:

> "Hi, ChatGPT! I would like your help in generating dataset domain information for image classification based on the dataset paper. A few words are good. Please return directly without explanation.
>
> {uploaded PDF}."

Here, {uploaded PDF} represents the uploading of the paper of the dataset to LLMs. Generated dataset domain information is summarized in Tab. 1.

Adding dataset domain to templates. We supplement templates with dataset domain information in the following four ways: (1) Add "a type of {domain}". (2) Re-

Dataset	Domain Information
IN-1K [7]	real scenario; natural scene
Caltech [9]	object; everyday objects; common items
Cars [17]	car; vehicles; auto-mobile
CUB [37]	bird; wildlife; ornithology
DTD [5]	textures; patterns; surface; material
ESAT [13]	land cover; remote sensing; satellite photo; satellite imagery; aerial or satellite images; centered satellite photo
FGVC [21]	aircraft; airplane; plane; airliner
FLO [25]	flower; floral; botanical; bloom
Food [2]	food; dishes; cuisine; nourishment
Pets [27]	pet; domestic animals; breed; dog or cat
Places [45]	place; scene
SUN [40]	place; scene
UCF [34]	action; human action; human activities; person doing

Table 1. **Generated dataset domain information.**

place "{class}" with "{domain}:{class}". (3) Replace "photo" with "{domain}". (4) Replace "photo" with "{domain} photo". Taking "a photo of a {class}" as an example, we modify the templates with the above four ways to add dataset domain information as follows:

1. a photo of a {class}, a type of {domain}.
2. a photo of a {domain}: {class}.
3. a {domain} of a {class}.
4. a {domain} photo of a {class}.

Here, {class} and {domain} denote category name and dataset domain information, respectively.

A.2. Details of Building Description Library

Description Library aims to provide a set of visual descriptions for each category, enhancing visual semantics for fine-grained recognition in prompts. It contains processes for generating visual descriptions and category synonyms and integrating descriptions with the best templates.

Generating category synonym. Except for descriptions, we also replace class names from the dataset with their synonyms to create diverse class-specific prompts. For this purpose, we use the following prompt to ask LLMs to generate category synonyms:

"Hi, ChatGPT! I would like your help in generating category synonyms. As a {domain} expert, I will provide you with a category name. Your task is to provide synonyms for the current category. If it has subclasses, return them as well. Please return directly without explanation.

User: I want to give the synonyms of {class}.
Assistant: "

Method	Prompts
DCLIP [23]	Q: What are useful visual features for distinguishing a {class} in a photo? A: There are several useful visual features to tell there is a {class} in a photo:
CuPL-Base [28]	Describe what a {class} looks like. Describe a {class}. What are the identifying characteristics of a {class}?
CuPL-Full [28]	Describe what a {class} looks like. How can you identify a {class}? What does a {class} look like? Describe an image from the internet of a {class} A caption of an image of a {class}:
GPT4Vis [39]	I want you to act as an image description expert. I will give you a word and your task is to give me 20 sentences to describe the word. Your description must accurately revolve around this word and be as objective, detailed and diverse as possible. In addition, the subject of your description is a some kind of object photograph. Output the sentences in a json format which key is the the word and the value is a list composed of these sentences. Do not provide any explanations. The first word is "{class}".
AdaptCLIP [31]	What characteristics can be used to differentiate {class} from other {domain} based on just a photo? Provide an exhaustive list of all attributes that can be used to identify the {domain} uniquely. Texts should be of the form "{domain} with {characteristic}".

Table 2. **Prompts for generating visual descriptions.**

Generating visual descriptions for each category. Similar to previous description methods [23, 28, 31, 39], we instruct LLM to generate visual descriptions for each category by several prompts, which are summarized in Tab. 2.

Integrating descriptions with the best templates. We use the following prompt to integrate descriptions with templates: "{template}. {description.}".

After the above processes, we collect diverse visual descriptions for each category c, denoted as VD(c). For each group iteration, we select the descriptions for categories in the specific group as the description library. Moreover, the prompt sampling strategy also utilizes these descriptions for class-specific initialization.

B. Details of Prompt Sampling Strategy

The detailed prompt sampling strategy is summarized in Alg. 1. Visual descriptions of each class VD(c) are collected by the above process (see Appendix A.2). We utilize

the candidate prompt P_t^* with the best templates as an initial point. The RANDOMSAMPLE$(\cdot)$ operator denotes randomly selecting a set of elements from a given set. We randomly sample descriptions for each category to create multiple candidate prompts (Lines 2-8). After T_{sample}-times steps, we select the candidate prompt $\hat{P}_0$ with the highest score for description initialization (Line 9). It ensures that subsequent optimization is around the optimal initial point. We set $T_{sample} = 32$ for all datasets in the default setting.

Algorithm 1 Prompt Sampling Strategy.

Require: $\mathcal{D} \leftarrow \{(x, y)\}_n$: training samples, $F : \mathcal{D} \times P \rightarrow \mathbb{R}$: score function, $\mathcal{C}$: class labels, VD(c): visual descriptions of class c, P_t^*: the prompt candidate with the best template

1: $\mathcal{U} \leftarrow \{P_t^*\}$
2: **for** $i = 1$ to T_{sample} **do**
3: $\quad P_i \leftarrow P_t^*$
4: $\quad$ **for all** class $c \in \mathcal{C}$ **do**
5: $\quad\quad P_i \leftarrow P_i \cup$ RANDOMSAMPLE(VD(c))
6: $\quad$ **end for**
7: $\quad \mathcal{U} \leftarrow \mathcal{U} \cup \{P_i\}$
8: **end for**
9: $\hat{P}_0 \leftarrow \arg\max_{P \in \mathcal{U}} F(\mathcal{D}, P)$
10: **return** the candidate prompt with the highest score $\hat{P}_0$

C. Details of Group Sampling Strategy

The detailed group sampling strategy is summarized in Alg. 2. It contains processes of obtaining misclassified categories and selecting the worst and salient groups.

Obtaining misclassified categories. In Lines 1-8 of Alg. 2, we collect misclassified set for each category by MISCLASS$(\cdot)$ operator. Given an image x, if the prediction pred(x) is not its corresponding label y, we will add pred(x) to the misclassified set for category y. In fact, we also ablate the K-means clustering algorithm to group categories (in Appendix G.3). Results show that the misclassified set achieves better performance than the K-means algorithm.

Selecting the worst groups aims to select categories with the lowest top-n_{wst} accuracy and corresponding misclassified categories. We first compute the accuracy for each category in Line 11. Then, we sort the categories by accuracy and retain the top-n_{wst} worst categories in Line 15. Finally, n_{wst} groups are added to the set $\mathcal{G}$ in Lines 18-20.

Selecting the salient groups aims to select categories with the top-n_{sln} performance gains and its misclassified categories after adding descriptions. In Line 13, we compute the accuracy gains after adding the descriptions. Then, we sort the categories by accuracy gain and retain the top-n_{sln} accuracy gain categories in Line 16. At last, n_{sln} groups are added to the set $\mathcal{G}$ in Lines 21-23.

Algorithm 2 Group Sampling Strategy.

Require: $\mathcal{D} \leftarrow \{(x, y)\}_n$: training samples, $F : \mathcal{D} \times P \rightarrow \mathbb{R}$: score function, $\mathcal{C}$: class labels, VD(c): visual descriptions of class c, P_t^*: prompt candidate with the best template, pred(x): prediction for image x

1: **for all** class $c \in \mathcal{C}$ **do**
2: $\quad$ MISCLASS$(c) \leftarrow \emptyset$
3: **end for**
4: **for all** training sample $(x, y) \in \mathcal{D}$ **do**
5: $\quad$ **if** pred$(x) \neq y$ **then**
6: $\quad\quad$ MISCLASS$(y) \leftarrow$ MISCLASS$(y) \cup \{\text{pred}(x)\}$
7: $\quad$ **end if**
8: **end for**
9: **for all** class $c \in \mathcal{C}$ **do**
10: $\quad$ **Select Class Images**: DATA$(c) \leftarrow \{(x, y) \mid y = c\}_{(x,y)\in\mathcal{D}}$
11: $\quad$ **Compute Accuracy**: ACC$(c) \leftarrow F($DATA$(c), P_t^*)$
12: $\quad$ **Add Descriptions**: $P_c \leftarrow P_t^* \cup$ VD(c)
13: $\quad$ **Compute Accuracy Gain**: ACCGAIN$(c) \leftarrow F($DATA$(c), P_c) -$ ACC(c)
14: **end for**
15: **Sort Class by Accuracy**: $\mathcal{C}_{wst}$, retaining the classes with the lowest top-n_{wst} accuracy
16: **Sort Class by Accuracy Gain**: $\mathcal{C}_{sln}$, retaining the classes with the top-n_{sln} accuracy gain
17: **Initialize Group Set**: $\mathcal{G} \leftarrow \emptyset$
18: **for all** class $c \in \mathcal{C}_{wst}$ **do**
19: $\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{$MISCLASS$(y)\}$
20: **end for**
21: **for all** class $c \in \mathcal{C}_{sln}$ **do**
22: $\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{$MISCLASS$(y)\}$
23: **end for**
24: **return** sampled groups $\mathcal{G}$

Finally, we collect $S = n_{wst} + n_{sln}$ groups for subsequent description optimization.

D. More Implementation Details

D.1. Hyperparameter Settings

In Tab. 3, we show the searched hyperparameter settings for thirteen datasets. All results are average with four seeds. Except for $1, 2, 3$ as seeds like CoOp [47], we add 42 as our fourth seed to further evaluate the stability of our method. In the default setting, we use the same LLMs as the description methods, *i.e.*, GPT-3 [3] for CuPL [28] and DCLIP [23], GPT-4 [26] for GPT4Vis [39] and AdaptCLIP [31].

D.2. More Related Work

Large-scale vision-language models like CLIP [29] have shown promising performance on various tasks. They align visual and textual spaces to a joint space via training on mil-

Dataset	T	M	N	α	n_{wst}	n_{sln}	T_{sample}
IN-1K [7]	4	8	8	1e3	4	4	32
Caltech [9]	2	8	8	1e2	2	2	32
Cars [17]	4	8	8	1e4	4	4	32
CUB [37]	4	8	8	1e2	4	4	32
DTD [5]	4	8	8	1e3	4	4	32
ESAT [13]	4	8	8	1e3	3	3	32
FGVC [21]	4	8	8	1e3	4	4	32
FLO [25]	4	8	8	1e3	4	4	32
Food [2]	4	8	8	1e3	2	2	32
Pets [27]	2	8	8	1e4	2	2	32
Places [45]	4	8	8	1e2	3	3	32
SUN [40]	2	8	8	1e4	4	4	32
UCF [34]	4	8	8	1e3	3	3	32

Table 3. **Hyperparameters settings for thirteen datasets.**

lions of image-text pairs from the web. Other work [1, 6, 8, 14, 18, 19, 24, 35, 41] has furthered this paradigm to learn more accurate semantic alignment in joint space. In this work, we advance VLMs for downstream tasks by progressively learning optimal class-specific prompts with minimal supervision and no human intervention.

E. Results on Different Backbones

Settings. In Tab. 4, we show results of our ProAPO in different backbones, including ResNet50, ResNet101, ViT-B/32, ViT-B/16, ViT-L/14 for CLIP [29], ViT-B/32 for OpenCLIP [4], ViT-B/16 for EVA02 [8], and ViT-B/16 for SigLIP [43]. We compare our ProAPO with vanilla VLMs and the SOTA description method CuPL [28].

Results. We see that our ProAPO consistently improves vanilla CLIP and CuPL in thirteen datasets across all backbones. Compared to vanilla VLMs, our ProAPO enhances them by at least 3.4% average accuracy in thirteen datasets. Moreover, we see notable performance improvement on several fine-grained datasets, such as DTD [5], ESAT [13], FLO [25], and UCF [34]. It further verifies that class-specific descriptions provide helpful knowledge for fine-grained recognition. Besides, iterative optimization by our ProAPO also enhances the description method CuPL.

More interesting findings. We find that as the backbones of VLMs become larger, the performance improvement by ProAPO gradually decreases. For example, from ViT-B/32 to ViT-B/16 to ViT-L/14, the gain for CLIP is from 5.7% to 5.6% to 5.0%. Moreover, similar results appear in different models with the same backbone, *i.e.*, the vanilla model with better results achieves a lower performance increase. For example, from CLIP [29] to OpenCLIP [4] on ViT-B/32 backbone, the gain is from 5.7% to 4.7%, and from CLIP [29] to EVA02 [8] to SigLIP [43], the gain is from 5.6% to 4.0% to 3.4%. We argue that the model with the higher result has more knowledge, which may be affected less by prompt quality. Overall, our ProAPO continues to improve the performance of VLMs.

F. More Comparisons with SOTA Methods

In this section, we compare our ProAPO with more SOTA prompt tuning methods. These methods adapt VLMs from both visual and textual views.

Comparison of test-time prompt tuning methods. In Tab. 5, our ProAPO outperforms SOTA test-time prompt tuning methods on 11 datasets. Notably, we adapt VLMs solely from the textual view, while TPT methods introduce textual and visual views (*i.e.*, augmented images), which further verifies the effectiveness of our method.

Comparison of vector-based prompt-tuning methods Since recent prompt-tuning methods adapt VLMs using both visual and textual views, we combine ProAPO with an adapter (*i.e.*, APE [48]) for a fair comparison. **(1) Higher performance in low-shot.** In Tab. 5, ProAPO consistently outperforms these methods, which verifies that optimizing prompts in natural language is more effective in low-shot tasks. **(2) Better transferability and interpretability**. Unlike vector-based prompt-tuning methods that search in a continuous space, ProAPO benefits from the discrete nature of natural language, leading to better interpretability and easily transfers across different backbones (shown in **??**). **(3) Lower performance in high-shot**. However, in Tab. 11, ProAPO shows a sub-optimal result compared to CoOp [47] in high-shot settings. This is due to the limited language search space and iteration steps.

Comparison of AWT [49]. First, since AWT uses augmented visual and textual views to adapt VLMs, we compare ProAPO with AWT under the augmented textual view for a fair comparison. In Tab. 5, the result shows our ProAPO improves AWT-text by 6.1% on average, verifying that our progressive optimization improves prompt quality. In addition, we introduce a common adapter-based method to our ProAPO and compare it with AWT-Adapter in the one-shot setting. We see that our ProAPO achieves comparable results. These results suggest that ProAPO and AWT are complementary.

Comparison of iCM [12]. iCM is somewhat similar to ours, optimizing class-specific prompts with chat-based LLMs. However, it uses the whole validation set as supervision. In Tab. 6, we see that our ProAPO outperforms iCM significantly even under the one-shot supervision. This is because our ProAPO address challenges in class-specific prompt optimization by an offline generation algorithm to reduce LLM querying costs, an entropy-constrained fitness score to prevent overfitting, and two sampling strategies to find an optimal initial point and reduce iteration times.

Module	IN-1K	Caltech	Cars	CUB	DTD	ESAT	FGVC	FLO	Food	Pets	Places	SUN	UCF	Avg (11)	Avg (13)
CLIP [29] - ResNet50	57.9	84.5	53.9	44.7	38.8	28.6	15.9	60.2	74.0	83.2	38.2	58.0	56.9	55.6	53.4
CuPL [28]	61.2	88.3	55.3	48.7	49.5	38.2	18.9	67.0	80.1	86.1	41.2	63.1	63.3	61.1	58.5
ProAPO (ours)	**61.5**	**90.3**	**58.0**	**50.7**	**52.3**	**51.7**	**21.1**	**75.1**	**81.8**	**88.7**	**41.8**	**63.7**	**66.0**	**64.6**	**61.8**
Δ	+ 3.6	+ 5.8	+ 4.1	+ 6.0	+ 13.5	+ 23.1	+ 5.2	+ 14.9	+ 7.8	+ 5.5	+ 3.6	+ 5.7	+ 9.1	+ 9.0	+ 8.4
CLIP [29] - ResNet101	61.4	89.9	63.3	49.6	40.3	31.7	18.3	64.3	83.4	86.9	37.9	59.0	61.2	60.0	57.5
CuPL [28]	61.4	91.0	61.2	45.3	49.7	28.7	18.6	59.0	82.7	86.6	**40.6**	62.3	56.4	59.8	57.2
ProAPO (ours)	**63.6**	**92.3**	**64.4**	**52.2**	**51.6**	**45.9**	**21.2**	**69.6**	**84.9**	**89.6**	**40.6**	**63.5**	**64.0**	**64.6**	**61.8**
Δ	+ 2.2	+ 2.4	+ 1.1	+ 2.6	+ 11.3	+ 14.2	+ 2.9	+ 5.3	+ 1.5	+ 2.7	+ 2.7	+ 4.5	+ 2.8	+ 4.6	+ 4.3
CLIP [29] - ViT-B/32	62.1	91.2	60.4	51.7	42.9	43.9	20.2	66.0	83.2	86.8	39.9	62.1	60.9	61.8	59.3
CuPL [28]	64.4	92.9	60.7	53.3	50.6	50.5	20.9	69.5	84.2	87.0	43.1	66.3	66.4	64.9	62.3
ProAPO (ours)	**64.7**	**94.4**	**61.7**	**55.4**	**53.5**	**63.0**	**23.0**	**74.3**	**85.3**	**91.0**	**43.3**	**66.6**	**69.0**	**67.9**	**65.0**
Δ	+ 2.6	+ 3.2	+ 1.3	+ 3.7	+ 10.6	+ 19.1	+ 2.8	+ 8.3	+ 2.1	+ 4.2	+ 3.4	+ 4.5	+ 8.1	+ 6.1	+ 5.7
CLIP [29] - ViT-B/16	66.9	93.2	65.5	55.3	44.3	51.0	24.4	70.6	88.4	89.0	40.8	62.5	67.7	65.8	63.0
CuPL [28]	69.6	94.3	66.1	57.2	53.8	55.7	26.6	73.9	88.9	91.2	43.4	**69.0**	70.3	69.0	66.1
ProAPO (ours)	**69.9**	**95.2**	**67.7**	**59.0**	**55.8**	**65.3**	**28.3**	**82.7**	**89.5**	**92.7**	**43.8**	68.9	**73.1**	**71.7**	**68.6**
Δ	+ 3.0	+ 2.0	+ 2.2	+ 3.7	+ 11.5	+ 14.3	+ 3.9	+ 12.1	+ 1.1	+ 3.7	+ 3.0	+ 6.4	+ 5.4	+ 5.9	+ 5.6
CLIP [29] - ViT-L/14	73.5	95.1	76.8	62.5	52.1	61.5	33.4	79.5	93.1	93.3	40.7	67.6	75.0	72.8	69.5
CuPL [28]	76.7	96.2	77.6	61.4	62.6	62.4	36.1	79.7	93.4	93.8	43.8	73.2	78.3	75.5	71.9
ProAPO (ours)	**76.8**	**97.1**	**78.8**	**65.1**	**64.8**	**74.3**	**38.3**	**87.3**	**93.9**	**94.6**	**44.4**	**73.4**	**80.1**	**78.1**	**74.5**
Δ	+ 3.3	+ 2.0	+ 2.0	+ 2.6	+ 12.7	+ 12.8	+ 4.9	+ 7.8	+ 0.8	+ 1.3	+ 3.7	+ 5.3	+ 5.1	+ 5.8	+ 5.0
OpenCLIP [4] - ViT-B/32	66.2	94.7	88.2	65.6	51.3	49.4	23.0	71.2	82.4	90.7	41.5	68.1	65.0	68.2	65.9
CuPL [28]	66.7	94.4	86.6	65.9	62.4	50.1	25.5	69.5	81.7	90.8	43.3	69.1	65.8	69.3	67.1
ProAPO (ours)	**67.0**	**95.8**	**88.7**	**67.3**	**65.1**	**66.0**	**27.5**	**81.8**	**83.2**	**91.9**	**43.4**	**69.7**	**70.2**	**73.3**	**70.6**
Δ	+ 0.8	+ 1.1	+ 0.5	+ 1.7	+ 13.8	+ 16.6	+ 4.5	+ 10.6	+ 0.8	+ 1.2	+ 1.9	+ 1.6	+ 5.2	+ 5.1	+ 4.7
EVA02 [8] - ViT-B/16	74.6	**97.2**	79.2	60.8	49.7	68.0	24.6	75.6	89.5	92.2	42.9	70.7	68.6	71.8	68.7
CuPL [28]	75.4	96.7	79.2	61.8	59.1	61.7	27.5	75.2	89.3	92.1	44.0	72.5	71.9	72.8	69.7
ProAPO (ours)	**75.5**	97.0	**80.0**	**62.8**	**61.3**	**74.2**	**29.7**	**89.1**	**89.6**	**93.5**	**44.5**	**72.5**	**75.2**	**76.2**	**72.7**
Δ	+ 0.9	-0.2	+ 0.8	+ 2.0	+ 11.6	+ 6.2	+ 5.1	+ 13.5	+ 0.1	+ 1.3	+ 1.6	+ 1.8	+ 6.6	+ 4.4	+ 4.0
SigLIP [43] - ViT-B/16	75.8	97.3	90.5	62.3	62.8	44.6	43.6	85.5	91.5	94.1	41.6	69.5	74.9	75.5	71.8
CuPL [28]	76.0	98.0	90.5	63.0	64.9	42.8	45.1	87.0	90.7	94.5	43.5	69.9	73.4	75.7	72.3
ProAPO (ours)	**76.4**	**98.3**	**91.7**	**66.2**	**69.1**	**55.8**	**47.1**	**93.3**	**92.2**	**94.9**	**44.3**	**71.7**	**75.9**	**78.8**	**75.2**
Δ	+ 0.6	+ 1.0	+ 1.2	+ 3.9	+ 6.3	+ 11.2	+ 3.5	+ 7.8	+ 0.7	+ 0.8	+ 2.7	+ 2.2	+ 1.0	+ 3.3	+ 3.4

Table 4. **Results of our ProAPO on different backbones. Avg (11)** and **Avg (13)** denote average results across 11 datasets (excluding CUB [37] and Places [45]) and all 13 datasets, respectively. Δ denotes performance gains compared to vanilla VLMs.

G. More Ablation Results

G.1. Ablation of Template and Description Optimization

In Tab. 7, we ablate key components in template and description optimization on the ResNet50 backbone.

(1) Ablation of Template Optimization. In the main paper (Sec. 4.3), we show that prompt ensembling is better than a single prompt. Moreover, dataset domain information also plays a significant role in template optimization. Without domain information, we see a performance drop in our ATO by an average of 1.0% (from 58.3% to 59.3 %) on thirteen datasets. This is because domain information provides contextual information, which can mitigate issues of semantic ambiguity caused by class names.

(2) Ablation of Description Optimization. Without label synonyms to increase description diversity, a performance degradation appears by an average of 1.4% (from 60.4% to 61.8%) on thirteen datasets. It verifies the effectiveness of optimization class names, which are usually ignored in previous description methods [23, 28, 31, 39].

(3) Template VS Description Optimization. Compared with template optimization, we see a notable performance improvement with description optimization, especially in CUB [37], DTD [5], ESAT [13], FLO [25], and UCF [34] datasets. It demonstrates that optimizing

Module (ViT-B/16)	**IN-1K**	**Caltech**	**Cars**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**SUN**	**UCF**	**Avg (11)**
Vanilla CLIP [29]	66.9	93.2	65.5	44.3	51.0	24.4	70.6	88.4	89.0	62.5	67.7	65.8
Test-Time Prompt Tuning Methods												
TPT [33]	69.0	94.2	66.9	47.8	42.4	24.8	69.0	84.7	87.8	65.5	68.0	65.5
DiffTPT [10]	70.3	92.5	67.0	47.0	43.1	25.6	70.1	87.2	88.2	65.7	68.2	65.9
PromptAlign [32]	71.4	94.0	68.5	47.2	47.9	24.8	72.4	86.7	90.8	67.5	69.5	67.3
Self-TPT-v [50]	**73.0**	94.7	68.8	49.4	51.9	27.6	71.8	85.4	91.3	68.2	69.5	68.3
Vector-based Prompt Tuning Methods												
UPT [42]	69.6	93.7	67.6	45.0	66.5	28.4	75.0	84.2	82.9	68.8	72.0	68.5
CoCoOp [46]	69.4	93.8	67.2	48.5	55.3	12.7	72.1	85.7	91.3	68.3	70.3	66.8
MaPLe [15]	69.6	92.6	66.6	52.1	71.8	26.7	83.3	80.5	89.1	64.8	71.8	69.9
ALIGN [38]	69.8	94.0	68.3	54.1	53.2	29.6	81.3	85.3	91.4	69.1	74.4	70.1
PromptSRC [16]	68.1	93.7	69.4	56.2	73.1	27.7	85.9	84.9	92.0	69.7	74.8	72.3
Description-Based Methods												
w/o adapters												
CuPL [28]	69.6	94.3	66.1	53.8	55.7	26.6	73.9	88.9	91.2	69.0	70.3	69.0
AWT-text [49]	68.9	95.2	66.0	52.0	52.6	26.1	74.5	89.4	91.2	68.4	69.8	68.6
ProAPO (ours)	69.9	95.2	67.7	55.8	65.3	28.3	82.7	89.5	92.7	68.9	73.1	71.7
ProAPO w/ AWT-text	69.4	95.3	67.8	54.3	67.1	27.4	82.1	89.6	93.2	68.5	73.1	71.6
w/ adapters												
AWT-Adapter [49]	72.1	95.1	**73.4**	59.4	**76.3**	**33.9**	85.6	85.9	92.9	**72.7**	**78.4**	75.1
ProAPO w/ APE [48]	71.3	**95.8**	70.9	**60.6**	72.4	33.2	**91.4**	**89.9**	**93.4**	71.0	77.6	**75.2**

Table 5. **Comparison of our ProAPO with more SOTA methods under one-shot supervision. Avg (11)** denote average results across 11 datasets.

Module (ViT-B/32)	**IN-1K**	**Caltech**	**CUB**	**DTD**	**ESAT**	**FLO**	**SUN**	**UCF**	**Avg (8)**
Vanilla CLIP	62.1	91.2	51.7	42.9	43.9	66.0	62.1	60.9	60.1
Automatic Prompt Optimization Methods									
iCM [12] (w/ validation set)	64.5	92.7	**56.1**	51.4	56.3	72.2	66.2	67.0	65.8
ProAPO (w/ 1-shot)	**64.7**	**94.4**	55.4	**53.5**	**63.0**	**74.3**	**66.6**	**69.0**	**67.6**

Table 6. **Comparison of our ProAPO with iCM [12].** Avg (8) denotes average results across 8 datasets.

class-specific prompts can find discriminative information for fine-grained classification.

G.2. More Ablation of Operators

To further explore whether each operator has a role in searching the optimal result, we show the number of each operator causing the new optimal score during the iterations in Tab. 8. We see that each operator in iterative optimization may generate a better prompt. It further demonstrates that each operator is helpful in ProAPO. Notably, the crossover operator has the highest times to update the optimal score, which demonstrates that it makes the model search for the optimal prompt faster with limited iterations.

G.3. More Ablation of Group Sampling

In Tab. 9, we ablate how to select categories in the group sampling strategy. We consider the settings for optimizing all categories in one group, selecting random categories and the best categories with their misclassified categories in groups. In rows a)-c) of Tab. 9, we see notable performance degradation compared to full ProAPO. It further demonstrates that optimizing salient and worst groups can achieve comparable results with all categories and save iteration costs. Moreover, we also consider replacing misclassified categories with a K-Means clustering algorithm. A performance drop appears in row d), which verifies the effectiveness of selecting misclassified categories in groups.

Module (ResNet50)	**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**
Vanilla CLIP	57.9	84.5	53.9	44.7	38.8	28.6	15.9	60.2	74.0	83.2	38.2	58.0	56.9	55.6	53.4
Template Optimization Methods															
PN [20]	59.6	89.1	56.2	-	44.8	49.0	18.1	67.2	78.3	88.1	-	61.0	60.2	61.1	-
ATO (w/o dataset domain)	60.4	88.9	56.8	47.0	45.0	43.7	17.9	67.4	79.9	87.8	40.0	61.2	61.5	61.0	58.3
ATO	61.3	89.4	57.4	49.2	45.4	46.4	18.4	68.1	80.5	88.5	40.2	61.8	63.9	61.9	59.3
Description Optimization Methods															
ProAPO (w/o synonyms)	**61.5**	89.7	**58.3**	49.7	46.6	46.8	20.5	74.6	81.0	**88.8**	40.9	62.3	64.8	63.2	60.4
ProAPO (ours)	**61.5**	**90.3**	58.0	**50.7**	**52.3**	**51.7**	**21.1**	**75.1**	**81.8**	88.7	**41.8**	**63.7**	**66.0**	**64.6**	**61.8**

Table 7. **Ablation of template and description optimization.** Avg (11) and Avg (13) denote average results across 11 datasets (excluding CUB [37] and Places [45]) and all 13 datasets, respectively. ATO denotes our automatic template optimization algorithm.

Dataset	Add	Del	Rep	Cross	Mut	**Total**
IN-1K [7]	3	4	5	5	2	19
Caltech [9]	5	5	6	12	3	31
Cars [17]	7	8	5	8	3	31
CUB [37]	9	4	10	6	2	31
DTD [5]	5	3	8	8	2	26
ESAT [13]	2	4	6	8	1	21
FGVC [21]	6	2	6	5	3	22
FLO [25]	5	3	11	5	4	28
Food [2]	5	3	4	5	2	19
Pets [27]	4	2	5	6	2	19
Places [45]	3	2	8	12	4	29
SUN [40]	4	2	3	5	2	16
UCF [34]	5	6	8	6	2	27
Sum	63	48	85	91	32	319

Table 8. **Number of times for each operator that update the optimal score. Total** denotes the total number of iterations when achieving the highest score.

G.4. Ablation of Cost Computation

In Tab. 10, we detail the time each process consumes on ImageNet. Compared to previous LLM-generated description methods, we similarly query LLMs one-time to generate descriptions (*i.e.*, process of building prompt library). In addition, we introduce iterative processes to refine prompts and two sampling strategies to save costs. With a few additional costs (15 min v.s. 60 min), our ProAPO improves previous methods by at least 2.7% on average. This further verifies the efficiency of our method.

H. More Hyperparameter Analysis

H.1. Effect of Shot Numbers

In Tab. 11, we show the effect of the number of training samples per category. Specifically, we conduct experiments with 1, 2, 4, 8, and 16 shots. Moreover, we introduce the performance of the optimal prompt searched in the test set as the upper bound of ProAPO. Compared with CoOp [47], ProAPO achieves remarkable performance when shots ≤ 2, which demonstrates the effectiveness of our method under low-shot settings. Since we only adapt VLMs in a training-free way, the performance increases finitely as the training samples increase. We attribute two key directions for further performance improvement in high-shot settings. First, our result is still far from the upper bound (66.1 % in 16 shots VS 67.2 % for the upper bound). We need to improve the prompt generation algorithm and the score function to find better candidate prompts within the limited iterations. Second, the upper bound of our ProAPO is much smaller than the prompt tuning method. We need to use a larger natural language search space (*e.g.*, more diverse descriptions, or more query times of LLMs) to further increase the upper bound of the optimal result.

H.2. Effect of Scalar in Score Function

In Tab. 12, we show the effect of α in **??**. We see that performance improves as the α increases. This is because the entropy constraint provides more information to select better candidate prompts. We see a stable result when $\alpha \in [5e2, 5e3]$, which means a better trade-off between accuracy and entropy constraint. However, a high α may be biased to the train set, thus harming the performance.

H.3. Effect of Sampled Numbers in Prompt Sampling Strategy

In Fig. 1, we show the effect of sampled numbers T_{sample} of Alg. 1. The $T_{sample} = 0$ means that the prompt sampling strategy is not used. As the number of T_{sample} increases, we see a slight performance gain when $T_{sample} < 4$. After $T_{sample} \geq 4$, a consistent improvement appears because the initial search point achieves a higher score than the baseline. We achieve stable results when $T_{sample} \geq 32$.

Module (ViT-B/32)	**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**	**Times**
CuPL	64.4	92.9	60.7	53.3	50.6	50.5	20.9	69.5	84.2	87.0	43.1	66.3	66.4	64.9	62.3	-
a) w/ all categories in one group	64.5	93.3	60.9	53.5	51.6	52.2	22.2	70.8	84.5	87.9	42.3	**66.7**	**69.4**	65.8	63.1	**20 min**
b) w/ random selected group	64.3	93.7	**61.8**	55.2	48.7	59.5	22.6	72.9	85.2	90.8	42.6	65.4	68.4	66.7	63.9	15 min
c) w/ performance best group	64.1	93.0	61.2	54.4	47.4	56.8	20.7	68.2	85.1	88.6	42.4	65.0	65.4	65.0	62.5	15 min
d) w/ K-Means algorithm	64.6	93.8	**61.8**	55.1	49.4	59.6	22.8	74.0	**85.3**	90.7	42.7	65.4	69.0	67.0	64.2	17 min
ProAPO (full model)	**64.7**	**94.4**	61.7	**55.4**	**53.5**	**63.0**	**23.0**	**74.3**	**85.3**	**91.0**	**43.3**	66.6	69.0	**67.9**	**65.0**	15 min

Table 9. **More ablation of group sampling strategy.** We ablate the ways for selecting salient groups. **Times** denotes the time that ProAPO runs on ImageNet with the default setting.

Process	**Build Library**	**Sample Strategy**	**Template Optim.**	**Description Optim.**
Times	60 min	3 min	1.6 min	10.4 min

Table 10. **Computation cost analysis in the ImageNet dataset.**

Dataset	**Module**	**TF**	**Number of training samples**					**UB**
	(RN50)		1	2	4	8	16	
Avg (11)	CoOp [47]	✗	59.6	62.3	**66.8**	**69.9**	**73.4**	-
	ProAPO	✓	**64.6**	**65.0**	65.4	65.8	66.1	67.2
IN-1K	CoOp [47]	✗	57.2	57.8	60.0	**61.6**	**63.0**	-
	ProAPO	✓	**61.5**	**61.6**	**61.5**	**61.6**	61.6	61.7
Caltech	CoOp [47]	✗	87.5	87.9	89.6	90.2	91.8	-
	ProAPO	✓	**90.3**	**90.4**	**90.6**	**90.7**	**91.0**	91.1
Cars	CoOp [47]	✗	55.6	58.3	**62.6**	**68.4**	**73.4**	-
	ProAPO	✓	**58.0**	**58.5**	58.8	58.9	59.1	60.8
DTD	CoOp [47]	✗	44.4	45.2	**53.5**	**60.0**	**63.6**	-
	ProAPO	✓	**52.3**	**52.7**	53.0	53.4	53.6	
ESAT	CoOp [47]	✗	50.6	**61.5**	**70.2**	**76.7**	**83.5**	-
	ProAPO	✓	**51.7**	53.5	55.6	57.4	58.3	62.2
FGVC	CoOp [47]	✗	9.6	18.7	**21.9**	**26.1**	**31.3**	-
	ProAPO	✓	**21.1**	**21.0**	21.2	21.2	21.3	21.5
FLO	CoOp [47]	✗	68.1	**77.5**	**86.2**	**91.2**	**94.5**	-
	ProAPO	✓	**75.1**	75.6	76.4	76.7	77.8	79.1
Food	CoOp [47]	✗	74.3	72.5	73.3	71.8	74.7	-
	ProAPO	✓	**81.8**	**82.0**	**82.1**	**82.2**	**82.3**	82.9
Pets	CoOp [47]	✗	85.9	82.6	86.7	85.3	87.0	-
	ProAPO	✓	**88.7**	**89.4**	**89.5**	**89.8**	**89.9**	91.0
SUN	CoOp [47]	✗	60.3	59.5	63.5	**65.5**	**69.3**	-
	ProAPO	✓	**63.7**	**63.8**	**63.8**	63.8	63.9	64.5
UCF	CoOp [47]	✗	61.9	64.1	67.0	**71.9**	**75.7**	-
	ProAPO	✓	**66.0**	**66.8**	**67.1**	68.1	68.9	71.4

Table 11. **Scaling up to more shots. Avg (11)** denotes average results across 11 datasets. **TF** denotes training-free approaches. **UB** denotes upper bound evaluated on the test set.

α	0	1e1	1e2	5e2	1e3	5e3	1e4	1e5
Avg (13)	62.3	63.4	64.4	64.9	**65.0**	64.8	63.7	63.1

Table 12. **Effect of α value in Eq.6 across 13 datasets.**

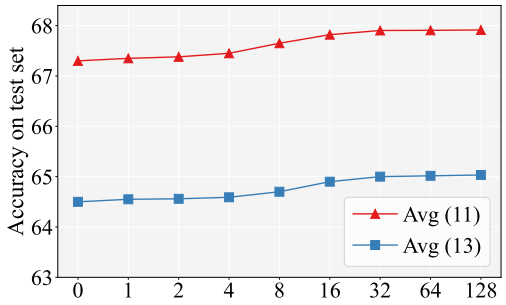


Figure 1. **Effect of sampled numbers T_{sample}.**

H.4. Effect of Quality of Prompt Library

In Fig. 2 and Fig. 3, we analyze two key factors affecting the prompt library: LLM-query prompts and generated descriptions. Our ProAPO improves prompt quality even under a small number of query prompts and descriptions, demonstrating its effectiveness in a limited prompt library.

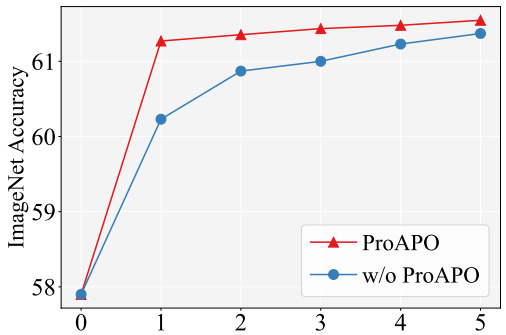


Figure 2. **Effect of Number of LLM-query Prompts.**

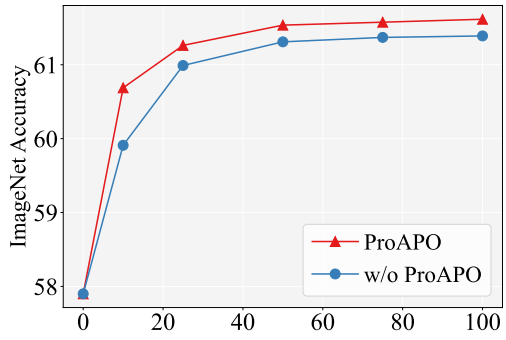


Figure 3. **Effect of Number of Generated Descriptions.**

I. More Qualitative Results

In Fig. 4, we show more examples of the changes in descriptions with our ProAPO, including images of animals, flowers, and textures. Similarly, we see that common descriptions are removed and discriminative ones are retained for fine-grained categories, which further verifies the effectiveness of our progressive optimization.

J. Detailed Results of More Benefits by Optimal Prompts

J.1. Transfer to Adapter-based Methods

In Fig. 5 and Fig. 6, we show the detailed results of popular training-free and training adapter-based methods [11, 36, 44, 48] with different prompt initialization, *i.e.*, SOTA method CuPL [28] and our ProAPO. Adapter-based methods with ProAPO (solid lines) consistently surpass those with CuPL (dotted lines). It reveals that high-quality prompts make adapters perform better. Even in low shots, training with ProAPO achieves notable performance gains, which further verifies its effectiveness.

J.2. Transfer to Different Backbones

In Fig. 7, we show detailed results of transferring prompts from source to target models in thirteen datasets. Our optimized prompts of ResNet50 and ViT-B/32 are reported. We see that ProAPO achieves stable performance gains compared to CuPL [28], which verifies that ProAPO transfers easily across different backbones.

K. Detailed Results of Performance Improvement Analysis

K.1. Analysis of the Effect of Single VS Ensemble Prompts

In Tab. 13, we show detailed results of the effect of single vs ensemble prompts. Compared to PN [20], we utilize prompt ensembling instead of a single prompt to optimize the template and description. We observe that ensemble templates have a higher upper bound than the single template. Similarly, our optimized templates achieve higher performance than PN [20], even better than the best single template, further verifying the effectiveness of our method.

K.2. Performance Improvement of Description Methods by ProAPO

In Tab. 14, we show detailed results of description methods [23, 28, 31, 39] with our ATO and ProAPO. We see a notable improvement in description methods by at least 2.7% average in thirteen datasets. It further verifies the effectiveness of our progressive optimization.

L. Detailed Results of Ablation Study

L.1. Ablation of Edit- and Evolution-based Operators

In Tab. 15, we show detailed ablation results of edit- and evolution-based operators. For edit-based operators, we observe that the model with add, delete, and replace operations achieves a higher result in row d). After introducing evolution-based operators, *i.e.*, crossover operator to combine advantages of high-scoring candidates, and mutation operator to avoid locally optimal solutions, we see an increase in performance in rows e)-g). It confirms that evolution-based operators make the model search the optimal prompt faster with limited iterations.

L.2. Ablation of Two Sampling Strategies

In Tab. 16, we show detailed ablation results of two sampling strategies. Without the prompt sampling, we see a slight decrease in times while results drop in row a). It verifies the effectiveness of the prompt sampling. Without the group sampling to select salient categories for optimization, we observe a notable increase in time costs (from 15 min to 300+ min, 20 times) yet similar results in row b) and the full model. It reveals that group sampling simultaneously improves performance and efficiency.

L.3. Ablation of Different Score Functions

In Tab. 17, we show detailed ablation results of score functions. Accuracy obtains the worst result as the score function due to the overfitting problem. Our full model with accuracy and entropy constraints as the score function achieves the SOTA result. The score function with only accuracy or entropy constraint achieves suboptimal results, suggesting a trade-off process between them.

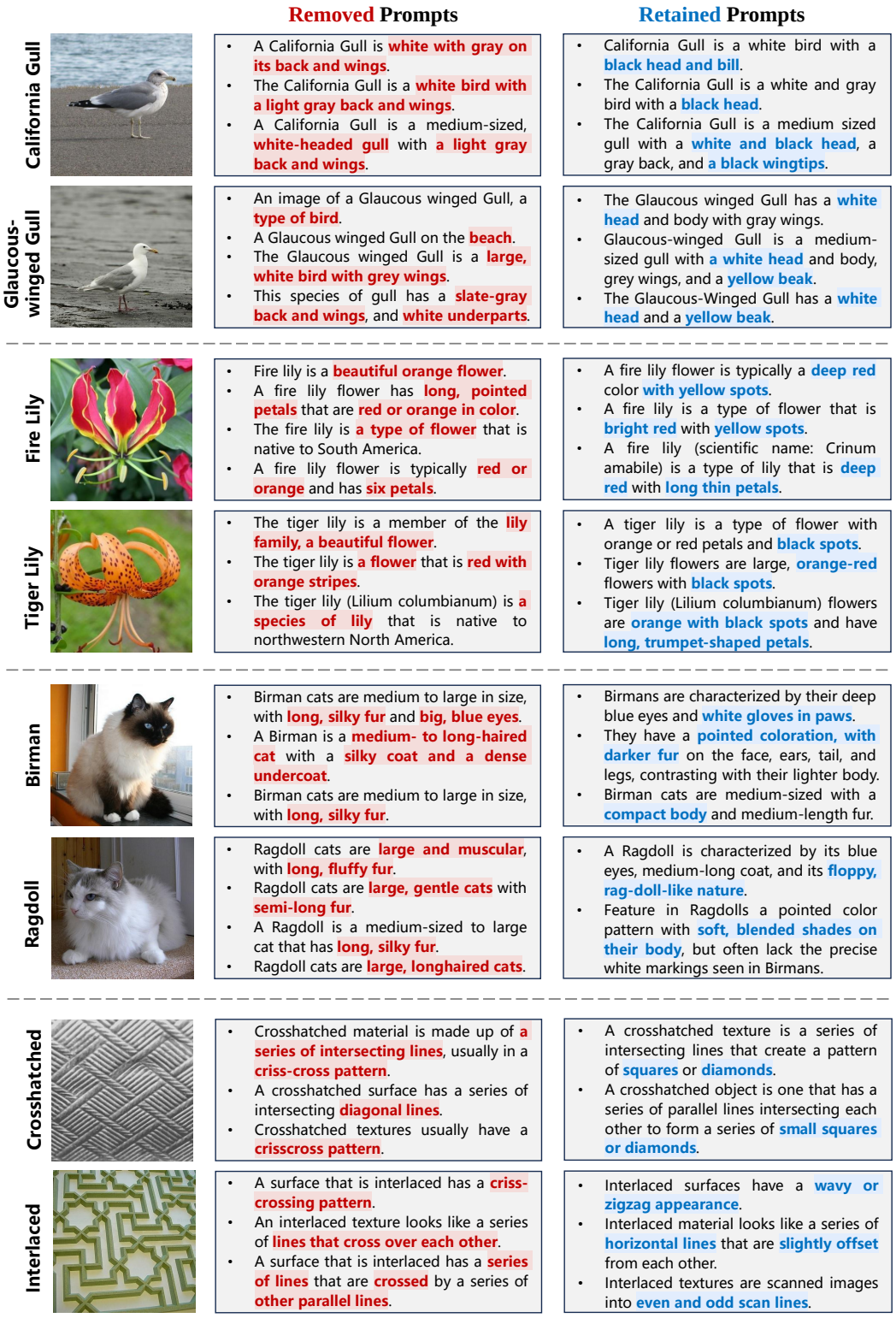
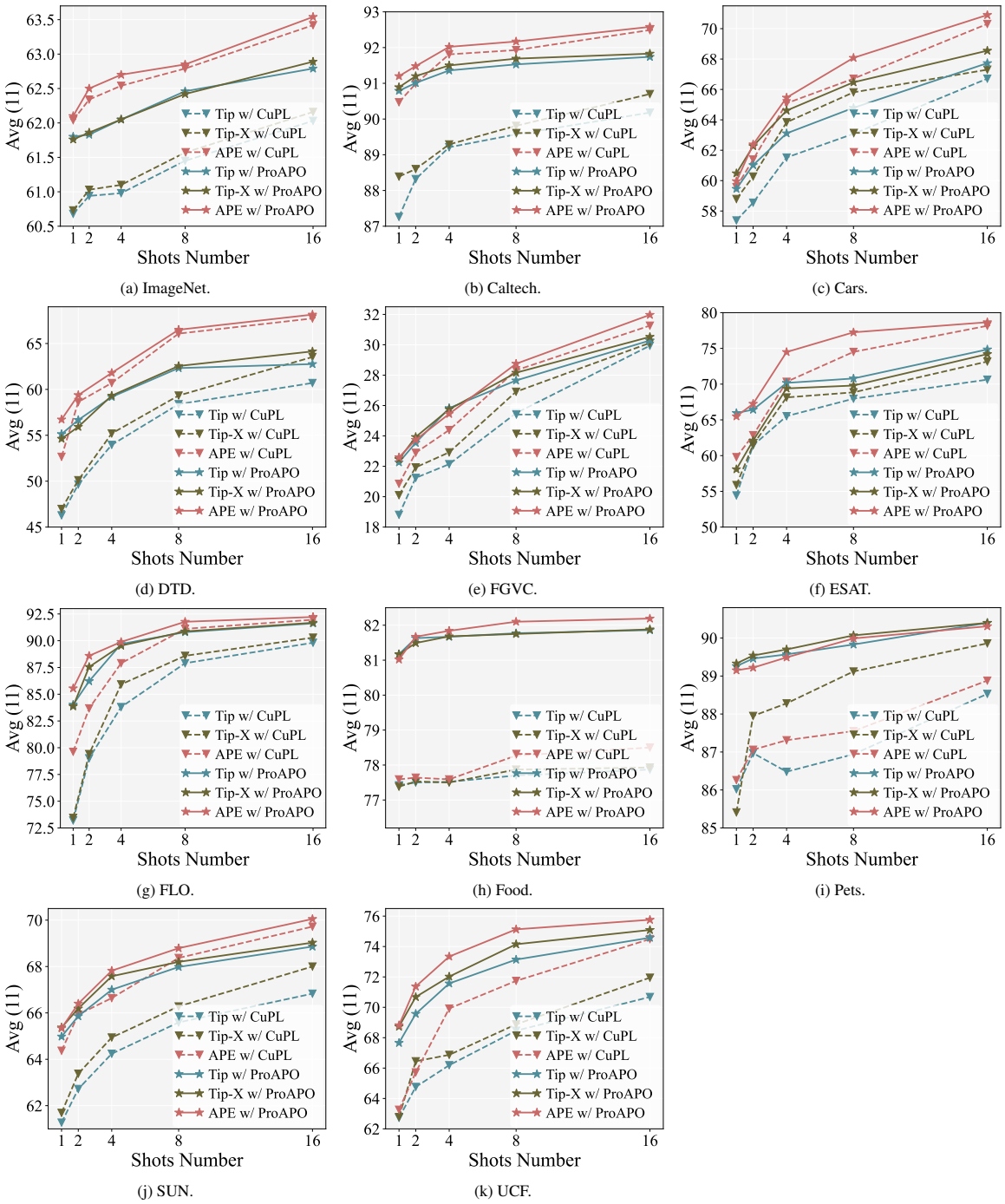


Figure 4. **Qualitative analysis of class-specific prompt optimization by ProAPO.** Shaded **red** and **blue** words denote common and discriminative descriptions in two confused categories.



(a) ImageNet. (b) Caltech. (c) Cars.

(d) DTD. (e) FGVC. (f) ESAT.

(g) FLO. (h) Food. (i) Pets.

(j) SUN. (k) UCF.

Figure 5. **Results of training-free adapter-based methods with different initial prompts.** Solid and dotted lines denote prompt initialization with ProAPO and CuPL, respectively. We see that our ProAPO consistently improves adapter-based methods.

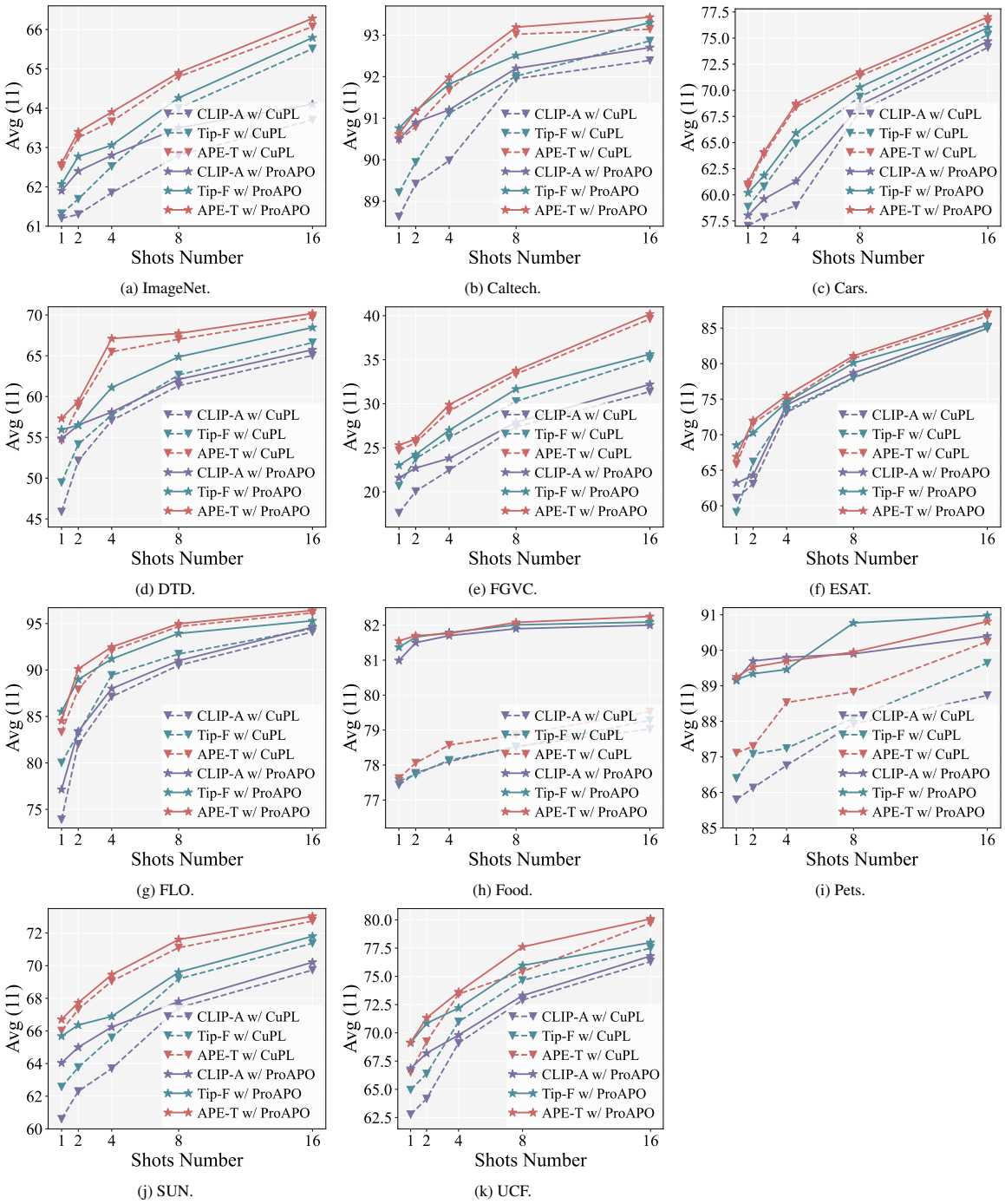


Figure 6. **Results of training adapter-based methods with different initial prompts.** Solid and dotted lines denote prompt initialization with ProAPO and CuPL, respectively. We see that our ProAPO consistently improves adapter-based methods.

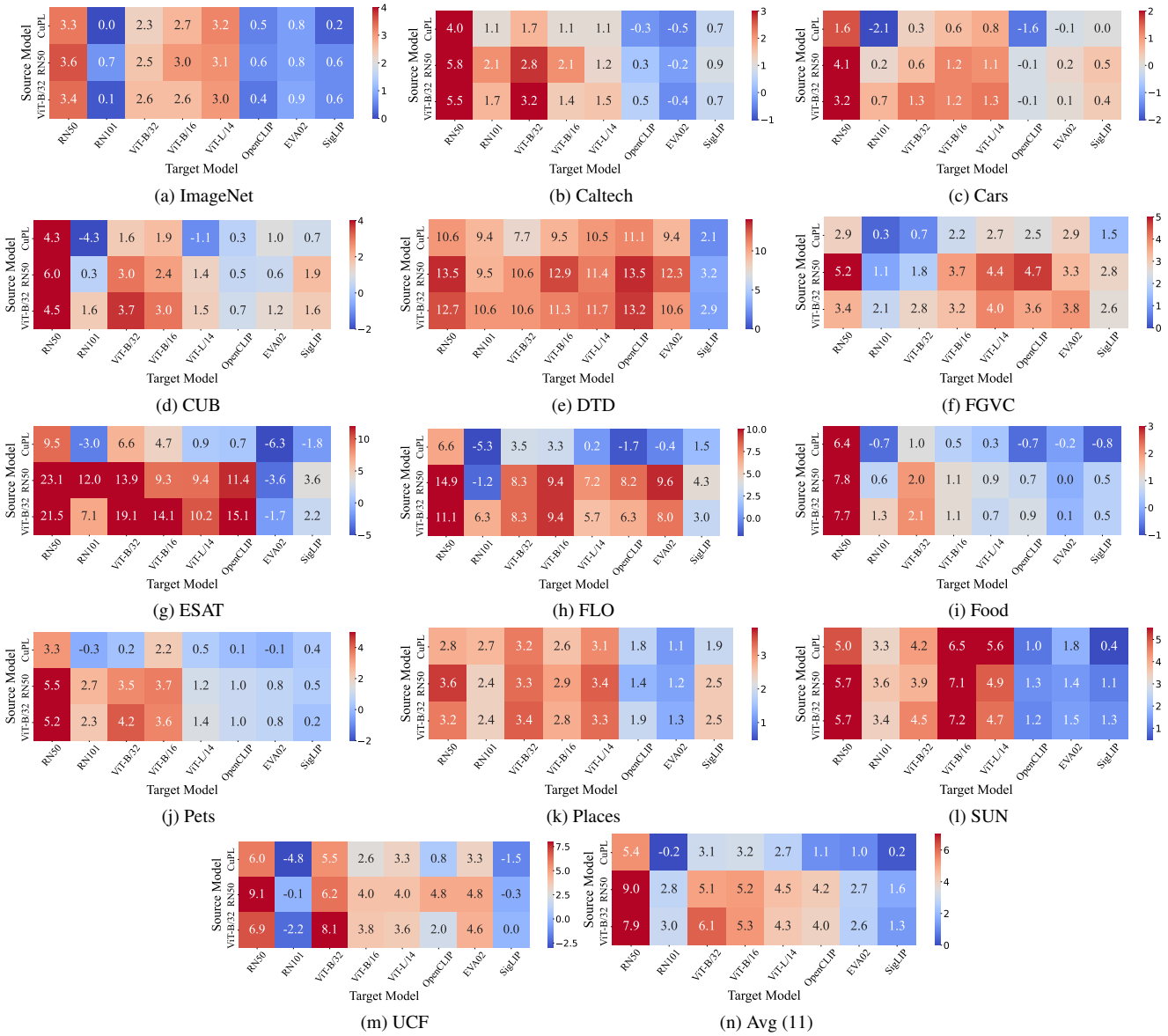


Figure 7. **Results of prompt transfer to different backbones.** The value denotes performance gains compared to vanilla VLMs. Our optimized prompts of ResNet50 and ViT-B/32 are reported. We see that we achieve stable performance gains compared to CuPL [28].

Module (ResNet50)	**IN-1K**	**Caltech**	**Cars**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**SUN**	**UCF**	**Avg (11)**
CLIP (a photo of a {})	57.9	84.5	53.9	38.8	28.6	15.9	60.2	74.0	83.2	58.0	56.9	55.6
Single Prompt												
PN [20]	59.6	89.1	56.2	44.8	49.0	18.1	67.2	78.3	88.1	61.0	60.2	61.1
Best Single*	60.2	89.2	57.9	45.0	46.0	18.3	68.1	81.8	88.3	61.5	62.6	61.7
Ensemble Prompt												
ATO (ours)	61.3	89.2	57.9	45.4	44.7	18.2	68.1	81.8	88.5	61.8	63.9	61.9
Best Ensemble*	**61.5**	**90.0**	**58.4**	**47.0**	**49.1**	**18.7**	**69.9**	**82.2**	**89.4**	**62.1**	**64.8**	**63.0**

Table 13. **Analysis of the effect of single vs ensemble prompts.** * denotes results evaluated in the test set. ATO is our automatic template optimization algorithm. We see that our optimized templates achieve higher results than PN [20], even better than the best single template.

Module (ViT-B/32)	**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**
Vanilla CLIP	62.1	91.2	60.4	51.7	42.9	43.9	20.2	66.0	83.2	86.8	39.9	62.1	60.9	61.8	59.3
DCLIP [23]	63.3	92.7	59.4	52.7	44.1	38.4	19.4	66.1	83.9	88.1	41.2	65.0	65.8	62.4	60.0
+ **ATO**	63.8	93.0	60.3	52.5	46.5	54.1	21.8	68.9	84.0	88.4	41.5	65.4	66.0	64.7	62.0
+ **ProAPO**	**64.1**	**93.2**	**60.6**	**53.6**	**48.2**	**59.4**	**22.6**	**71.5**	**84.2**	**88.7**	**42.7**	**66.0**	**68.0**	**66.0**	**63.3**
Δ	+ 0.8	+ 0.5	+ 1.2	+ 0.9	+ 4.1	+ 21.0	+ 3.2	+ 5.4	+ 0.3	+ 0.6	+ 1.5	+ 1.0	+ 2.2	+ 3.6	+ 3.3
CuPL-base [28]	64.0	92.3	60.1	54.3	47.2	42.4	21.7	68.7	84.3	88.8	42.0	**66.2**	66.7	63.8	61.4
+ **ATO**	64.2	93.3	60.9	54.8	47.8	53.1	22.2	70.4	84.9	89.2	42.3	65.5	67.4	65.3	62.8
+ **ProAPO**	**64.4**	**94.2**	**61.8**	**55.9**	**48.1**	**62.1**	**23.2**	**74.4**	**85.4**	**91.0**	**42.7**	65.6	**68.6**	**67.2**	**64.4**
Δ	+ 0.4	+ 1.9	+ 1.7	+ 1.6	+ 0.9	+ 19.7	+ 1.5	+ 5.7	+ 1.1	+ 2.2	+ 0.7	-0.6	+ 1.9	+ 3.4	+ 3.0
CuPL-full [28]	64.4	92.9	60.7	53.3	50.6	50.5	20.9	69.5	84.2	87.0	43.1	66.3	66.4	64.9	62.3
+ **ATO**	64.5	93.7	61.0	54.0	52.0	58.7	22.1	70.5	84.6	89.2	43.2	66.4	67.5	66.4	63.6
+ **ProAPO**	**64.7**	**94.4**	**61.7**	**55.4**	**53.5**	**63.0**	**23.0**	**74.3**	**85.3**	**91.0**	**43.3**	**66.6**	**69.0**	**67.9**	**65.0**
Δ	+ 0.3	+ 1.5	+ 1.0	+ 2.1	+ 2.9	+ 12.5	+ 2.1	+ 4.8	+ 1.1	+ 4.0	+ 0.2	+ 0.3	+ 2.6	+ 3.0	+ 2.7
GPT4Vis [39]	63.5	93.1	61.4	52.7	48.5	47.0	21.4	69.8	84.3	88.1	42.7	64.2	65.7	64.3	61.7
+ **ATO**	63.8	93.4	61.2	53.8	49.0	54.0	22.4	70.8	84.7	88.1	42.6	64.7	66.8	65.3	62.7
+ **ProAPO**	**64.4**	**93.7**	**61.8**	**55.4**	**49.3**	**62.6**	**23.9**	**73.8**	**85.4**	**90.7**	**42.8**	**65.5**	**68.2**	**67.2**	**64.4**
Δ	+ 0.9	+ 0.6	+ 0.4	+ 2.7	+ 0.8	+ 15.6	+ 2.5	+ 4.0	+ 1.1	+ 2.6	+ 0.1	+ 1.3	+ 2.5	+ 2.9	+ 2.7
AdaptCLIP [31]	63.3	92.7	59.7	53.6	47.4	51.3	20.8	67.2	84.2	87.6	41.9	66.1	66.5	64.2	61.7
+ **ATO**	63.9	93.2	60.4	54.2	47.9	55.5	**22.4**	69.1	84.7	88.8	42.3	66.3	67.6	65.4	62.8
+ **ProAPO**	**64.4**	**93.7**	**61.8**	**55.5**	**49.6**	**61.6**	23.3	**73.8**	**85.4**	**91.0**	**42.6**	**66.5**	**68.6**	**67.2**	**64.5**
Δ	+ 1.1	+ 1.0	+ 2.1	+ 1.9	+ 2.2	+ 10.3	+ 2.5	+ 6.6	+ 1.2	+ 3.4	+ 0.7	+ 0.4	+ 2.1	+ 3.0	+ 2.8

Table 14. **Performance improvement of description methods by our ProAPO.** Avg (11) and Avg (13) denote average results across 11 datasets (excluding CUB [37] and Places [45]) and all 13 datasets, respectively. Δ denotes performance gains compared to baseline.

Component						**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**
Add	Del	Rep	Cross	Mut																
Vanilla CLIP (ViT-B/32)						62.1	91.2	60.4	51.7	42.9	43.9	20.2	66.0	83.2	86.8	39.9	62.1	60.9	61.8	59.3
edit-based generation																				
a)	✓					63.8	93.6	60.0	54.6	51.8	59.0	21.8	74.0	82.2	86.7	43.0	65.7	66.8	66.0	63.3
b)	✓	✓				64.6	94.0	60.9	55.0	52.6	59.3	21.8	72.0	83.2	88.0	43.2	66.4	68.0	66.4	63.8
c)	✓		✓			64.4	94.0	61.0	55.2	52.3	59.7	22.4	71.9	84.0	87.7	43.2	66.4	67.8	66.5	63.8
d)	✓	✓	✓			64.6	93.6	60.8	54.4	53.1	60.1	22.2	**74.7**	82.4	87.2	**43.4**	66.5	68.6	66.7	64.0
evolution-based generation																				
e)	✓	✓	✓	✓		64.6	94.3	61.2	55.0	53.2	62.6	22.9	73.9	84.3	88.0	43.1	**66.8**	68.5	67.3	64.5
f)	✓	✓	✓		✓	**64.7**	94.3	61.4	55.1	52.9	61.4	22.6	74.0	83.6	87.7	**43.4**	66.7	68.3	67.1	64.3
g)	✓	✓	✓	✓	✓	**64.7**	**94.4**	**61.7**	**55.4**	**53.5**	**63.0**	**23.0**	74.3	**85.3**	**91.0**	43.3	66.6	**69.0**	**67.9**	**65.0**

Table 15. **Ablation of edit- and evolution-based operators.**

Module (ViT-B/32)	**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**	**Times**
a) w/o prompt sampling	64.4	93.8	**61.8**	55.4	51.8	60.0	23.2	74.0	85.1	90.7	43.0	66.0	69.3	67.3	64.5	12 min
b) w/o group sampling	**64.8**	**94.5**	61.7	**55.5**	**53.6**	**63.5**	23.2	75.3	**85.4**	90.8	**43.3**	**66.7**	**69.8**	**68.1**	**65.2**	**306 min**
c) w/o sampling strategies	64.5	93.4	57.4	54.8	**53.6**	63.2	**23.4**	**76.8**	83.8	86.9	**43.3**	66.1	69.7	67.2	64.4	302 min
ProAPO (full model)	64.7	94.4	61.7	55.4	53.5	63.0	23.0	74.3	85.3	**91.0**	**43.3**	66.6	69.0	67.9	65.0	15 min

Table 16. **Ablation of two sampling strategies.**

Module (ViT-B/32)	**IN-1K**	**Caltech**	**Cars**	**CUB**	**DTD**	**ESAT**	**FGVC**	**FLO**	**Food**	**Pets**	**Places**	**SUN**	**UCF**	**Avg (11)**	**Avg (13)**
a) w/ only accuracy	64.0	93.0	60.8	54.2	49.1	55.5	20.4	68.3	84.8	88.4	41.9	64.6	65.1	64.9	62.3
b) w/ only entropy constrain	64.3	93.4	61.6	54.8	49.3	56.7	22.3	69.9	85.2	89.1	42.4	65.1	66.7	65.8	63.1
ProAPO (full model)	**64.7**	**94.4**	**61.7**	**55.4**	**53.5**	**63.0**	**23.0**	**74.3**	**85.3**	**91.0**	**43.3**	**66.6**	**69.0**	**67.9**	**65.0**

Table 17. **Ablation of different score functions.**

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, and Pauline Luc et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 4

[2] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *ECCV*, pages 446–461, 2014. 2, 4, 7

[3] Tom B. Brown, Benjamin Mann, and Nick Ryder et al. Language models are few-shot learners. In *NeurIPS*, 2020. 3

[4] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *CVPR*, pages 2818–2829, 2023. 4, 5

[5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014. 2, 4, 5, 7

[6] Yufeng Cui, Lichen Zhao, Feng Liang, Yangguang Li, and Jing Shao. Democratizing contrastive language-image pre-training: A CLIP benchmark of data, model, and supervision. *CoRR*, abs/2203.05796, 2022. 1, 4

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2, 4, 7

[8] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A visual representation for neon genesis. *Image Vis. Comput.*, 149:105171, 2024. 4, 5

[9] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW*, page 178, 2004. 2, 4, 7

[10] Chun-Mei Feng, Kai Yu, Yong Liu, Salman Khan, and Wangmeng Zuo. Diverse data augmentation with diffusions for effective test-time prompt tuning. In *ICCV*, pages 2704–2714, 2023. 6

[11] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *IJCV*, 132(2):581–595, 2024. 9

[12] Songhao Han, Le Zhuo, Yue Liao, and Si Liu. Llms as visual explainers: Advancing image classification with evolving visual descriptions. *CoRR*, abs/2311.11904, 2023. 4, 6

[13] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 12(7):2217–2226, 2019. 2, 4, 5, 7

[14] Chao Jia, Yinfei Yang, and Ye Xia et al. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, pages 4904–4916, 2021. 4

[15] Muhammad Uzair Khattak, Hanoona Abdul Rasheed, Muhammad Maaz, Salman H. Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *CVPR*, pages 19113–19122. IEEE, 2023. 6

[16] Muhammad Uzair Khattak, Syed Talal Wasim, Muzammal Naseer, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Self-regulating prompts: Foundational model adaptation without forgetting. In *ICCV*, pages 15144–15154, 2023. 6

[17] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV*, pages 554–561, 2013. 2, 4, 7

[18] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pages 12888–12900, 2022. 4

[19] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Su-

pervision exists everywhere: A data efficient contrastive language-image pre-training paradigm. In *ICLR*, 2022. 4

[20] Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models. In *CVPR*, pages 12687–12697, 2024. 1, 7, 9, 13

[21] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. 2, 4, 7

[22] Mayug Maniparambil, Chris Vorster, Derek Molloy, Noel Murphy, Kevin McGuinness, and Noel E. O'Connor. Enhancing CLIP with GPT-4: harnessing visual descriptions as prompts. In *ICCV*, pages 262–271, 2023. 1

[23] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. In *ICLR*, 2023. 2, 3, 5, 9, 14

[24] Norman Mu, Alexander Kirillov, David A. Wagner, and Saining Xie. SLIP: self-supervision meets language-image pre-training. In *ECCV*, pages 529–544, 2022. 4

[25] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729, 2008. 2, 4, 5, 7

[26] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. 3

[27] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012. 2, 4, 7

[28] Sarah M. Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *ICCV*, pages 15645–15655, 2023. 2, 3, 4, 5, 6, 9, 13, 14

[29] Alec Radford, Jong Wook Kim, and Chris Hallacy et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 1, 3, 4, 5, 6

[30] Karsten Roth, Jae-Myung Kim, A. Sophia Koepke, Oriol Vinyals, Cordelia Schmid, and Zeynep Akata. Waffling around for performance: Visual classification with random words and broad concepts. In *ICCV*, pages 15700–15711, 2023. 1

[31] Oindrila Saha, Grant Van Horn, and Subhransu Maji. Improved zero-shot classification by adapting vlms with text descriptions. In *CVPR*, pages 17542–17552, 2024. 2, 3, 5, 9, 14

[32] Jameel Abdul Samadh, Hanan Gani, Noor Hussein, Muhammad Uzair Khattak, Muzammal Naseer, Fahad Shahbaz Khan, and Salman H. Khan. Align your prompts: Test-time prompting with distribution alignment for zero-shot generalization. In *NeurIPS*, 2023. 6

[33] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *NeurIPS*, 2022. 6

[34] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 2, 4, 5, 7

[35] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: improved training techniques for CLIP at scale. *CoRR*, abs/2303.15389, 2023. 4

[36] Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. Sus-x: Training-free name-only transfer of vision-language models. In *ICCV*, pages 2725–2736, 2023. 9

[37] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *california institute of technology*, 2011. 2, 4, 5, 7, 14

[38] Dongsheng Wang, Miaoge Li, Xinyang Liu, Mingsheng Xu, Bo Chen, and Hanwang Zhang. Tuning multi-mode token-level prompt alignment across modalities. In *NeurIPS*, 2023. 6

[39] Wenhao Wu, Huanjin Yao, Mengxi Zhang, Yuxin Song, Wanli Ouyang, and Jingdong Wang. Gpt4vis: What can GPT-4 do for zero-shot visual recognition? *CoRR*, abs/2311.15732, 2023. 2, 3, 5, 9, 14

[40] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 2, 4, 7

[41] Lewei Yao, Runhui Huang, and Lu Hou et al. FILIP: fine-grained interactive language-image pre-training. In *ICLR*. OpenReview.net, 2022. 1, 4

[42] Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. Unified vision and language prompt learning. *CoRR*, abs/2210.07225, 2022. 6

[43] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, pages 11941–11952, 2023. 4, 5

[44] 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. 9

[45] Bolei Zhou, Àgata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 40(6):1452–1464, 2018. 2, 4, 5, 7, 14

[46] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16795–16804, 2022. 6

[47] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 3, 4, 7, 8

[48] Xiangyang Zhu, Renrui Zhang, Bowei He, Aojun Zhou, Dong Wang, Bin Zhao, and Peng Gao. Not all features matter: Enhancing few-shot CLIP with adaptive prior refinement. In *ICCV*, pages 2605–2615, 2023. 4, 6, 9

[49] Yuhan Zhu, Yuyang Ji, Zhiyu Zhao, Gangshan Wu, and Limin Wang. AWT: transferring vision-language models via augmentation, weighting, and transportation. In *NeurIPS*, 2024. 4, 6

[50] Yuhan Zhu, Guozhen Zhang, Chen Xu, Haocheng Shen, Xiaoxin Chen, Gangshan Wu, and Limin Wang. Efficient test-time prompt tuning for vision-language models. *CoRR*, abs/2408.05775, 2024. 6