OKGraph: Online Knowledge Graph Probing for Open-vocabulary Recognition

Supplementary Material

Junhui Yin[1,7†] Zhizhen Cai[2†] Puze Wang[2†] Guanzhou Ke[3] Jianhua Yang[4*]
Man Zhang[2] Qiang Zhang[6] Shengfeng He[7]

[1] University of Science and Technology Beijing [2] Beijing University of Posts and Telecommunications
[3] Beijing Jiaotong University [4] Institute of Automation, Chinese Academy of Sciences
[6] China Electric Power Research Institute Co., Ltd. (CEPRI) [7] Singapore Management University

This supplementary material provides additional technical details and experimental analyses that could not be fully included in the main paper due to space constraints. The contents are organized as follows:

- **S1**: Global Scene Graph Construction
- **S2**: Graph Matching
- **S3**: Results with Different Backbones
- **S4**: Hyperparameter Sensitivity Analysis
- **S5**: Inference Efficiency Analysis

S1. Global Scene Graph Construction

Graph Structure. To improve inference efficiency and reduce computational overhead, we pre-construct a global scene knowledge graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ offline, avoiding the need to build graphs for each test sample. The node set $\mathcal{V}_s$ contains all semantic categories in the dataset, where each node represents a concept category. The edge set $\mathcal{E}_s$ consists of *text-labeled relations* that encode visual attributes and semantic associations. We distinguish two types of relations:

- **Category-specific attribute relations** attach discriminative visual attributes to a category node c (e.g., "thick white fur" for polar bear). These relations are implemented as *self-loop (node-attached) edges* $e_c^{\text{cs}} = (c \rightarrow c, a)$.
- **Category-shared attribute relations** connect semantically related categories that share common visual properties (e.g., "four legs" for both polar bear and dog). Such relations are instantiated as cross-category edges $e_{c_i,c_j}^{\text{inter}} = (c_i \leftrightarrow c_j, a), i \neq j$.

To enhance efficiency, we encode attributes as *text-labeled relations* rather than modeling them as standalone nodes. Compared with treating attributes as explicit nodes, this edge-labeled design is more compact and directly compatible with CLIP text embeddings, enabling efficient aggregation of attribute relations with their associated category nodes. Moreover, to avoid dense pairwise connections, we do not connect all categories that share the same attribute. Instead, shared-attribute relations are only established within a semantic neighborhood, which preserves graph sparsity while maintaining meaningful semantic structure.

Attribute Extraction and Edge Instantiation. For each category c, we prompt a large language model (LLM) to output 4–8 concise visual attributes in a fixed structured format. For example, the LLM may describe a "chair" as "a piece of furniture typically with four legs and a backrest." From these descriptions, we extract key attributes (e.g., "four legs", "backrest") and use them as edge relations to connect categories that share the same attributes.

The entire graph construction process is fully automatic and does not require human intervention. Category-level descriptions are first generated and then re-processed by the LLM to extract concise attributes, which naturally form relations across category nodes.

To ensure attribute quality, we adopt a structured extraction pipeline. We normalize attributes by lowercasing and removing stopwords and punctuation, and merge near-duplicate attributes using CLIP text-embedding similarity. We then compute the corpus frequency $\text{freq}(a)$ of each attribute across all classes. If $\text{freq}(a) = 1$, the attribute a is treated as a discriminative attribute. Otherwise, it is treated as a shared attribute.

To further improve discriminativeness, we prompt the LLM with multiple class names simultaneously, so that attributes are generated in a contrastive manner. This encourages the model to produce category-distinguishing descriptions rather than generic ones. Potential noisy attributes (e.g., "no wings") are mitigated by retaining only the top-3

*Corresponding author.
†These authors contributed equally.

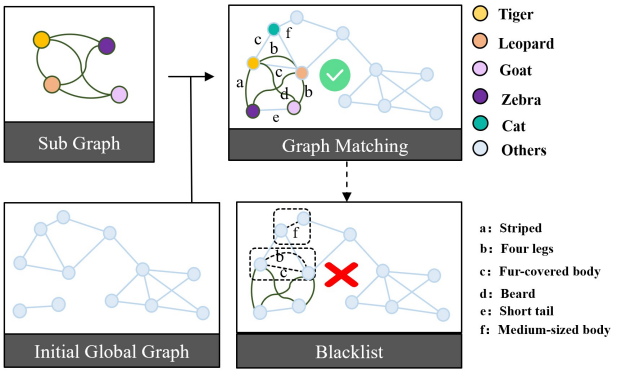

Figure 1. **Graph evolution.** Illustration of edge creation, addition, and pruning during knowledge graph refinement.

most discriminative attributes during inference.

Although each category is initially described by 4–8 attributes, only the top-3 discriminative attributes are used during inference. As a result, the performance is largely insensitive to the exact number of generated attributes. We verify this on the CounterAnimal Hard dataset, where using 4, 6, or 8 attributes all yields the same accuracy of 79.7, demonstrating robustness to attribute number as long as the most informative attributes are preserved.

To avoid dense all-pairs connections, shared-attribute relations are only instantiated within a restricted candidate neighborhood $\mathcal{N}_M(c)$ defined by the top-M CLIP semantic neighbors of class c. Two classes are connected by attribute a only if they both contain a and one lies in the other's neighborhood. This design keeps the graph sparse and computationally efficient while preserving semantically meaningful interactions.

Graph Evolution. We further provide qualitative examples to illustrate the evolution of the knowledge graph, including edge creation, addition, and deactivation.For instance, categories such as "tiger" and "leopard" are connected through the shared attribute "fur-covered body". As the graph evolves, attributes generated by the LLM (e.g., "stripes", "four legs") are added to the global graph to enrich semantic knowledge. Discriminative attributes (e.g., "stripes" for tiger) are retained, while overly generic attributes (e.g., "four legs" and "fur-covered body") are suppressed through a blacklist mechanism. This dynamic process allows the global graph to continuously refine its structure by preserving informative knowledge and removing redundant or noisy relations.

Graph and Edge Embedding. In our OKGraph, edges represent textual attributes associated with category nodes, and all nodes and edges are embedded into a unified vector space using the CLIP text encoder. Specifically, given a category c and an attribute a, we construct an attribute-conditioned textual description using a fixed prompt template:

```
Prompt: a photo of a {classname}
which has {attribute}.
```

The resulting text $e_{c,a}$ is then embedded via the CLIP text encoder to obtain the edge representation:

$$\text{Embed}(e_{c,a}) = \text{CLIP}\big(\texttt{Prompt}\big). \tag{1}$$

Embedding nodes and edges within a unified semantic space allows relational information to be naturally incorporated into the knowledge structure, enabling effective integration of graph-enhanced textual context during inference.

S2. Graph Matching

In this section, we describe the proposed Graph Matching and Knowledge Fusion mechanism (Algorithm 1). The objective of this module is to evaluate the consistency between the global scene graph $\mathcal{G}_s$ and the goal-oriented subgraph $\mathcal{G}_g$, and to determine whether $\mathcal{G}_g$ contains complementary knowledge that should be integrated into the global memory. The overall procedure consists of three phases: (1) Topological Structure Matching, (2) Semantic Node Matching, and (3) Fusion Decision.

Topological Structure Matching. We adopt a lightweight approximation of the Graph Edit Distance (GED) to assess the structural consistency between the scene graph $\mathcal{G}_s$ and the goal subgraph $\mathcal{G}_g$.Specifically, we first compute the number of nodes and edges in both graphs. The *raw structural distance* is defined as:

$$D_{\text{raw}} = \big||\mathcal{V}_g| - |\mathcal{V}_s|\big| + \big||\mathcal{E}_g| - |\mathcal{E}_s|\big|. \tag{2}$$

Here, $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. $|\mathcal{V}_s|$ and $|\mathcal{E}_s|$ represent the total number of nodes and edges in $\mathcal{G}_s$, while $|\mathcal{V}_g|$ and $|\mathcal{E}_g|$ correspond to those in $\mathcal{G}_g$.

To make this metric comparable across graphs of different scales, we normalize it using the maximum graph complexity, resulting in:

$$D_{\text{norm}} = \frac{D_{\text{raw}}}{\max(|\mathcal{V}_g| + |\mathcal{E}_g|,\ |\mathcal{V}_s| + |\mathcal{E}_s|,\ 1)}. \tag{3}$$

We then define the *topological similarity* as its complement:

$$S_T = 1 - D_{\text{norm}}, \tag{4}$$

where $S_T \in [0, 1]$. Lower values of S_T indicate that $\mathcal{G}_g$ and $\mathcal{G}_s$ share dissimilar node–edge structures, reflecting stronger topological complement.

Semantic Node Matching. Semantic node matching aims to evaluate whether the corresponding category nodes carry

Method	Caltech	Pets	Cars	Flowers	DTD	UCF101	EuroSAT	Food101	SUN397	Aircraft	Easy	Hard	**Avg**
CLIP [4]	81.5	81.0	56.1	60.2	38.8	53.5	23.7	71.5	57.0	15.7	84.3	66.1	57.5
CLIP + OKGraph	87.5$_{\uparrow 6.0}$	83.9$_{\uparrow 2.9}$	56.9$_{\uparrow 0.8}$	66.3$_{\uparrow 6.1}$	43.4$_{\uparrow 4.6}$	58.8$_{\uparrow 5.3}$	22.6$_{\uparrow -1.1}$	75.1$_{\uparrow 3.6}$	60.2$_{\uparrow 3.2}$	16.7$_{\uparrow 1.0}$	85.4$_{\uparrow 1.1}$	68.7$_{\uparrow 2.6}$	60.5$_{\uparrow 3.0}$
TPT [5]	86.5	84.5	58.5	62.7	41.6	58.7	26.0	74.9	60.8	16.9	86.4	68.0	60.5
TPT + OKGraph	87.6$_{\uparrow 1.1}$	85.4$_{\uparrow 0.9}$	58.8$_{\uparrow 0.3}$	65.7$_{\uparrow 3.0}$	43.0$_{\uparrow 1.4}$	60.6$_{\uparrow 1.9}$	25.3$_{\uparrow -0.7}$	76.8$_{\uparrow 1.9}$	63.2$_{\uparrow 2.4}$	17.8$_{\uparrow 0.9}$	86.7$_{\uparrow 0.3}$	68.9$_{\uparrow 0.9}$	61.7$_{\uparrow 1.2}$
C-TPT [6]	86.9	83.8	56.5	65.2	42.2	59.7	24.2	74.7	61.0	16.8	85.5	66.8	60.3
C-TPT + OKGraph	87.3$_{\uparrow 0.4}$	84.0$_{\uparrow 0.2}$	57.2$_{\uparrow 0.7}$	67.1$_{\uparrow 1.9}$	44.5$_{\uparrow 2.3}$	60.7$_{\uparrow 1.0}$	24.4$_{\uparrow 0.2}$	76.5$_{\uparrow 1.8}$	63.1$_{\uparrow 2.1}$	17.8$_{\uparrow 1.0}$	85.8$_{\uparrow 0.3}$	68.8$_{\uparrow 2.0}$	61.4$_{\uparrow 1.1}$

Table 1. **Zero-shot Classification Results.** We re-implement CLIP, TPT, C-TPT, and TCA using the backbone of CLIP RN50, and build our OKGraph upon them. The top-1 accuracy (%) on the test set is reported, and **Avg** denotes average results across all datasets.

Method	Caltech	Pets	Cars	Flowers	DTD	UCF101	EuroSAT	Food101	SUN397	Aircraft	**Avg**
CLIP (a photo of a {})	81.5	81.0	56.1	60.2	38.8	53.5	23.7	71.5	57.0	15.7	53.9
prompt tuning methods											
CoOp [7]	87.5	85.9	55.6	68.1	44.4	61.9	50.6	74.3	60.3	9.6	59.8
PLOT [1]	89.8	87.5	56.6	71.7	46.6	64.5	54.1	77.7	62.5	17.9	62.9
ProGrad [8]	88.7	88.4	58.4	73.2	46.1	65.6	56.3	76.0	60.5	18.8	63.2
automatic prompt optimization methods											
PN [2]	89.1	88.1	56.2	67.2	44.8	60.2	49.0	78.3	61.0	18.1	61.2
ProAPO [3]	89.9	88.7	57.9	74.8	49.8	64.6	52.0	81.6	63.5	20.7	64.4
ProAPO + OKGraph	90.0$_{\uparrow 0.1}$	88.9$_{\uparrow 0.2}$	58.1$_{\uparrow 0.2}$	75.2$_{\uparrow 0.4}$	50.4$_{\uparrow 0.6}$	64.9$_{\uparrow 0.3}$	52.3$_{\uparrow 0.3}$	81.8$_{\uparrow 0.2}$	63.9$_{\uparrow 0.4}$	21.2$_{\uparrow 0.5}$	64.7$_{\uparrow 0.3}$

Table 2. **Few-shot Classification Results.** We re-implement ProAPO using the CLIP ViT-B/32 backbone as our baseline and build OKGraph upon it. We compare our method with both hand-engineered and description-based approaches.

complementary semantic information. To this end, we embed all node descriptions using a pre-trained encoder Φ, mapping discrete category labels into a shared semantic space. Let $\mathbf{E}_s$ and $\mathbf{E}_g$ denote the embedding matrices of the scene graph and goal subgraph, respectively. For each category c appearing in both graphs, we compute the cosine similarity:

$$sim_c = \frac{\mathbf{E}_s^{(c)} \cdot (\mathbf{E}_g^{(c)})^T}{\|\mathbf{E}_s^{(c)}\| \, \|\mathbf{E}_g^{(c)}\|}. \tag{5}$$

We then aggregate these similarities using a pooling function $\mathcal{M}$ (mean pooling in our implementation), yielding the *semantic similarity score*:

$$S_N = \mathcal{M}(sim_c),$$

which reflects how well the semantic content of $\mathcal{G}_g$ aligns with that of $\mathcal{G}_s$.

Fusion Decision. After obtaining both topological similarity S_T and semantic similarity S_N, we determine whether the goal subgraph introduces complementary information not yet captured in $\mathcal{G}_s$. We define a fusion score:

$$S = 1 - S_T - S_N, \tag{6}$$

where larger values imply lower structural and semantic alignment, suggesting that $\mathcal{G}_g$ provides complementary information.

Based on a sensitivity threshold δ_s, the system decides whether to integrate the goal subgraph:

$$\text{Decision} = \begin{cases} \text{Fuse} : \mathcal{G}_s \leftarrow \mathcal{G}_s \oplus \mathcal{G}_g, & \text{if } S > \delta_s, \\ \text{Discard}, & \text{otherwise.} \end{cases} \tag{7}$$

Here, the graph composition operator $\oplus$ merges non-duplicated nodes and edges from $\mathcal{G}_g$ into $\mathcal{G}_s$, expanding the global knowledge base while avoiding redundant information. The full procedure is summarized in Algorithm 1.

S3. Results With Different Backbones

To study the impact of different CLIP backbones, we further integrate OKGraph into three representative CLIP-based methods, i.e., CLIP, TPT, and C-TPT, using the CLIP RN50 backbone. All experimental settings follow those of the ViT-B/16 configuration, except that the encoder is replaced with RN50. We evaluate on three tasks: zero-shot generalization, few-shot adaptation, and natural distribution shifts, and report top-1 accuracy on each dataset and average (Avg) results across all datasets.

Zero-shot Generalization. As shown in Table 1, OKGraph improves zero-shot classification performance by an average of 3.0% over the CLIP baseline. Specifically, with a convolutional backbone, the proposed dynamic semantic reasoning effectively enhances model's fine-grained recognition capacity. Moreover, when integrated with existing test-time adaptation methods (i.e., TPT [5] and C-TPT [6]), OKGraph also yields clear improvements, demonstrating that it can be seamlessly incorporated as a plug-and-play semantic enhancement module.

Few-shot Adaptation. Following [3], we conduct few-shot adaptation experiments and report the results in Table 2. Using ProAPO with CLIP RN50 as the baseline and integrating OKGraph into this framework, our method achieves

Algorithm 1 Graph Matching and Fusion

Initialize matching scores $\mathcal{S}_T \leftarrow 0, \mathcal{S}_N \leftarrow 0$
// Phase 1: Topology Matching
Calculate cardinality:
  $|\mathcal{V}_s| \leftarrow \text{len}(\mathcal{G}_s), |\mathcal{E}s| \leftarrow \sum \text{edges}(\mathcal{V}_s)$
  $|\mathcal{V}_g| \leftarrow \text{len}(\mathcal{G}_g), |\mathcal{E}g| \leftarrow \sum \text{edges}(\mathcal{V}_g)$
Compute Raw Graph Edit Distance:
  $D_{\text{raw}} \leftarrow \big||\mathcal{V}g| - |\mathcal{V}s|\big| + \big||\mathcal{E}g| - |\mathcal{E}s|\big|$
Compute Normalized Distance:
  $D_{\text{norm}} \leftarrow \frac{D_{\text{raw}}}{\max(|\mathcal{V}g|+|\mathcal{E}g|,\ |\mathcal{V}s|+|\mathcal{E}s|,\ 1)}$
Compute Topology Similarity:
  $\mathcal{S}_T \leftarrow 1 - D_{\text{norm}}$
// Phase 2: Node Matching
Extract semantic embeddings:
  $\mathbf{E}_s \leftarrow \text{Embed}(\mathcal{V}_s)$
  $\mathbf{E}_g \leftarrow \text{Embed}(\mathcal{V}_g)$
Compute Semantic Similarity:
  for each common class c **do**
    $sim_c \leftarrow \mathbf{E}_s^{(c)} \cdot (\mathbf{E}_g^{(c)})^T$
  end for
  $\mathcal{S}_N \leftarrow \mathcal{M}(sim_c)$
// Phase 3: Fusion Decision
Calculate fusion decision score S:
  $S \leftarrow 1 - \mathcal{S}_T - \mathcal{S}_N$
if $S > \delta_s$ **then** ▷ High novelty
  Execute Graph Fusion:
  $\mathcal{G}a \leftarrow \mathcal{G}s \oplus \mathcal{G}_g$
  Update $\mathcal{G}s \leftarrow \mathcal{G}a$
else
  Discard $\mathcal{G}_g$ ▷ Redundant
end if
return $\mathcal{G}_s$

Method	IN-A	IN-V2	IN-R	IN-S	**Avg**
CLIP [4]	21.8	51.4	56.2	33.4	40.7
CLIP + OKGraph	23.2$_{\uparrow 1.4}$	53.6$_{\uparrow 2.2}$	57.3$_{\uparrow 1.1}$	34.4 $_{\uparrow 1.0}$	42.1$_{\uparrow 1.4}$
TPT [5]	23.2	52.9	58.5	35.1	42.4
TPT + OKGraph	25.0$_{\uparrow 1.8}$	55.0$_{\uparrow 2.1}$	59.1$_{\uparrow 0.6}$	35.7$_{\uparrow 0.6}$	43.7$_{\uparrow 1.3}$
C-TPT [6]	22.5	53.2	57.3	35.1	42.0
C-TPT + OKGraph	23.6$_{\uparrow 1.1}$	54.4$_{\uparrow 1.2}$	58.0$_{\uparrow 0.7}$	35.3$_{\uparrow 0.2}$	42.8$_{\uparrow 0.8}$

Table 3. **Natural Distribution Shifts Results On RN50**. We re-implement CLIP, TPT, and C-TPT using the backbone of CLIP RN50, and build our OKGraph upon them. The top-1 accuracy (%) on the test set is reported, and **Avg** denotes average results across all datasets.

consistent performance gains across datasets. These results demonstrate that the textual and graph-based enhancements provided by OKGraph remain effective even when the backbone is further adapted with a limited number of labeled examples.

Natural Distribution Shifts (ImageNet-A/V2/R/S). On natural OOD benchmarks such as ImageNet-A, ImageNet-V2, ImageNet-R, and ImageNet-S (Table 3), OKGraph also brings consistent gains on top of CLIP RN50 and its TPT variants. Despite the limited capacity of the convolutional backbone, the structured semantic reasoning introduced by OKGraph improves robustness under distribution shifts, highlighting its advantage in challenging real-world scenarios.

Backbone Generalization. The RN50 experiments demonstrate that OKGraph consistently delivers performance gains even when applied to a smaller convolutional backbone. Together with the ViT-B/16 results presented in the main paper, these findings indicate that our method is not restricted to any particular architectural family. OKGraph can be reliably integrated into both transformer-based and convolutional CLIP variants while maintaining stable improvements across different backbones. This conclusion is further supported by additional experiments on other architectures, including CLIP (ViT-L/14), MetaCLIP2 (ViT-L/14), and SigLIP (siglip-so400m-patch14-38) on ImageNet-A. The results show that OKGraph is not specific to a particular CLIP backbone and can generalize more broadly to different vision-language model families.

Robustness of Prompt-based Settings. We also note that prompt-based methods are known to be sensitive to random initialization, and noticeable performance variations across different seeds have been reported in prior work. For example, TPT reports 60.81%, while DiffTPT and C-TPT report 59.57% and 59.90%, respectively. Against this background, the consistent improvements brought by OKGraph further highlight its robustness and stability under different experimental settings.

Comparison with Transduction. OKGraph is complementary to transductive methods such as TransCLIP rather than being an alternative limited to the same setting. Specifically, transductive methods require pre-optimization over the entire test set, whereas OKGraph is training-free and supports online semantic graph construction for real-time inference. Moreover, transductive methods mainly exploit relationships among visual image features, while OKGraph focuses on semantic relationships between categories. Because of these differences, the two directions can be naturally combined: one may first perform transductive optimization and then enhance online testing with OKGraph. In fact, integrating OKGraph into TransCLIP improves zero-shot performance on DTD from 49.5% to 50.7%, demonstrating that OKGraph remains effective even when built on top of transductive adaptation.

S4. Hyperparameter Sensitivity Analysis

To further assess the robustness of OKGraph under different hyperparameter settings, we perform sensitivity analysis on

No. of Attributes	4	6	8
Ours	79.7	79.7	79.7

Table 4. **Effect of the number of fine-grained attributes.** Performance remains stable across different attribute counts.

the following three additional hyperparameters:

1. the number of fine-grained attributes per category (4–8 during construction).
2. Threshold (δ_s) of S which is the difference of $\mathcal{G}_g$ and $\mathcal{G}_s$ in graph matching, where $S = 1 - S_T - S_N > \delta_s$.
3. Blacklist update frequency in the blacklist-based forgetting mechanism.
4. Activation frequency threshold (δ_f) in blacklist-based forgetting mechanism.

We adopt TPT as the baseline, integrate OKGraph into it, and conduct experiments on three representative datasets: EuroSAT, DTD, and Pets. We next analyze the effect of each hyperparameter in detail.

Number of fine-grained Attributes. In OKGraph, each category is initially described by 4–8 attributes, while only the top-3 most discriminative attributes are used during inference.

We evaluate the performance under different numbers of generated attributes on the CounterAnimal Hard dataset. As shown in Table 4, the performance remains identical when using 4, 6, or 8 attributes, demonstrating that OKGraph is largely insensitive to the attribute count as long as the most informative attributes are preserved.

The hyperparameter δ_s. This hyperparameter controls whether the target subgraph $\mathcal{G}_g$ provides complementary information relative to the global scene graph $\mathcal{G}_s$. We compute the complementarity score as

$$S = 1 - S_T - S_N, \tag{8}$$

and only when $S > \delta_s$ do we permit the fusion of edges from $\mathcal{G}_g$ into $\mathcal{G}_s$. If the difference is too small (*i.e.*, the fusion is redundant), the update is rejected. We evaluate δ_s over the range $[0.1, 0.3, 0.5, 0.7, 0.9]$, with results shown in Figure 2. As shown in Figure 2, model performance remains largely stable, demonstrating the strong robustness of OKGraph with respect to the hyperparameter δ_s. When δ_s is set too small (e.g., $\delta_s = 0.1$), fusion is triggered too frequently, causing the global graph to absorb noisy or redundant edges and slightly degrading accuracy. Conversely, when δ_s is set too large (e.g., $\delta_s = 0.9$), subgraphs rarely qualify for integration, and OKGraph rarely exploit test-time semantic cues. Balancing these effects, we set $\delta_s = 0.3$ in all experiments to ensure both stable performance and effective adaptive knowledge fusion.

Pruning Hyperparameters: Blacklist Update Frequency and Activation Threshold. The pruning mechanism in OKGraph is governed by two coupled hyperparameters: (i) the *blacklist update frequency*, which determines how often pruning is triggered, and (ii) the *activation threshold* δ_f, which specifies how many times an edge must be activated within a pruning cycle to avoid being removed. Together, these two parameters control the balance between removing long-term inactive edges and retaining rarely used but semantically valuable ones.

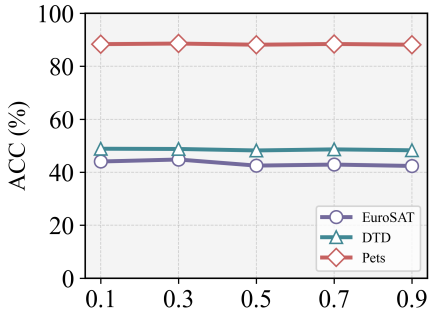

Figure 2. **Effect of Complementarity Threshold δ_s.**

We evaluate the blacklist update frequency over $200, 220, 240, 260, 280$ and the activation threshold δ_f over $40, 60, 80, 100, 120$, with the corresponding results presented in Figure 3 and Figure 4. Across all tested configurations, OKGraph maintains stable performance, demonstrating strong robustness to these pruning hyperparameters. When pruning is triggered too frequently or the retention threshold is set too high, the model prematurely removes edges that should be preserved, causing the global graph to oscillate between forgetting and reconstruction and ultimately degrading performance. Conversely, when pruning occurs too infrequently or the threshold is too low, noisy edges accumulate over time, introducing semantic clutter that interferes with test-time reasoning.

To balance these effects, we set the blacklist update frequency to 240 and the activation threshold to $\delta_f = 80$ in all experiments, achieving stable performance while effectively suppressing noise.

S5. Computational Efficiency Analysis

We evaluate the computational efficiency of our OKGraph on the DTD and Imagenet-A dataset under two baselines: CLIP ViT-B/16 and TPT. When integrated with CLIP ViT-B/16, OKGraph increases inference time but yields a substantial +5.9% and +2.0% improvement in top-1 accuracy. This result indicates that OKGraph enhances traditional zero-shot models by effectively leveraging domain-specific information extracted from tessst samples. A similar performance trend is observed when applying OKGraph to the TPT baseline. The primary computational cost arises from

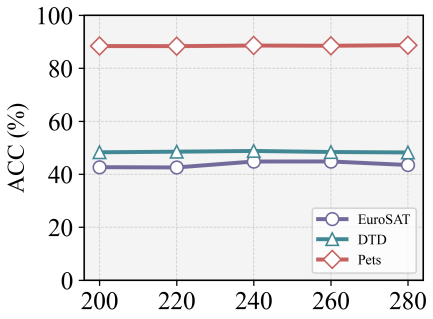

Figure 3. **Effect of Blacklist Update Frequency.**

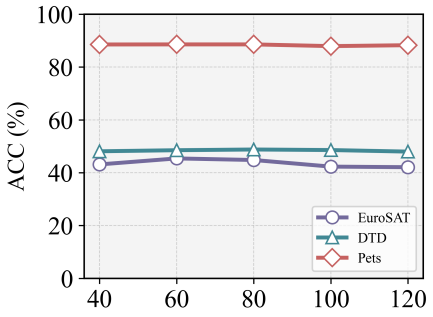

Figure 4. **Effect of Activation Threshold δ_f.**

Table 5. Comparisons of our OKGraph with CLIP and TPT in terms of efficiency (*Testing Time*) and effectiveness (*Accuracy*). The final column shows the accuracy gain relative to baselines.

Method	Dataset	Testing Time	Accuracy	Gain
CLIP ViT-B/16	DTD	2min 12s	44.6	–
	ImageNet-A	5min 37s	47.9	–
CLIP+OKGraph (Ours)	DTD	9min 36s	50.5	+5.9
	ImageNet-A	28min 36s	49.9	+2.0
TPT	DTD	17min 03s	47.0	–
	ImageNet-A	49min 01s	73.5	–
TPT+OKGraph (Ours)	DTD	39min 53s	48.8	+1.8
	ImageNet-A	62min 14s	74.2	+0.7

API calls to large language models, which is inherent to all methods that rely on LLM-based textual reasoning. Overall, although OKGraph introduces additional inference time, it consistently delivers clear accuracy gains across both baselines by dynamically exploiting domain information during test-time reasoning.

References

[1] Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. Plot: Prompt learning with optimal transport for vision-language models. *arXiv preprint arXiv:2210.01253*, 2022. 3

[2] Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12687–12697, 2024. 3

[3] Xiangyan Qu, Gaopeng Gou, Jiamin Zhuang, Jing Yu, Kun Song, Qihao Wang, Yili Li, and Gang Xiong. Proapo: Progressively automatic prompt optimization for visual classification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 25145–25155, 2025. 3

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 3, 4

[5] 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. *Advances in Neural Information Processing Systems*, 35:14274–14289, 2022. 3, 4

[6] Hee Suk Yoon, Eunseop Yoon, Joshua Tian Jin Tee, Mark Hasegawa-Johnson, Yingzhen Li, and Chang D Yoo. C-tpt: Calibrated test-time prompt tuning for vision-language models via text feature dispersion. *arXiv preprint arXiv:2403.14119*, 2024. 3, 4

[7] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022. 3

[8] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2605–2615, 2023. 3