Compositional Caching for Training-free Open-vocabulary Attribute Detection

Supplementary Material

In the following, we provide additional technical and implementation details, and further analyses. Specifically:

- Sec. 6 describes how we implemented and tested the baselines presented in Sec. 4 (*Main*).
- Sec. 7 presents detailed results with different backbone VLMs.
- Sec. 8 provides a detailed description of the box-free setting, including how we adapted zero-shot and cache-based models to it, as well as extended results.
- Sec. 9 introduces additional ablation studies to those presented in Sec. 4.2 (*Main*), and provides extended results for the ablations in *Main*.
- Sec. 10 offers additional qualitative results.
- Sec. 11 lists all the prompt templates that ComCa uses.
- Sec. 12 provides details on the hardware employed to conduct experiments.
- Sec. 13 comments on the limitations of ComCa, and provides possible future research directions to address them and improve attribute-detection models' capabilities.

6. Baselines

To conduct a detailed evaluation of ComCa, we implement several related works and test them in the open-vocabulary attribute detection task. They are presented in Sec. 4 of the main paper (*Main*), and this section provides additional details. This section also presents Tab. 6, an extended version of Tab. 1 (*Main*) providing (i) detailed results on all parts of the distributions of OVAD and VAW, (ii) details on the configuration used for each cache, and (iii) additional results with more configurations.

Image-based. It is a simple baseline that disregards attribute-object compositions, thus making it possible to understand their effects. In addition, it uses no textual input, as it focuses on image-to-image retrieval. As a result, this strategy does not suffer from potential noise in prompts and/or in the text-to-image retrieval process. The cache is constructed on a per-input basis, *i.e.*, specific to each input image. The rationale is that, given an image, its closest ones should bear the same semantic. Given an input image x, it retrieves the K shots x_c using no other prior knowledge, *i.e.*, it leverages no LLM and no web-scale database to estimate attribute-object compatibility. As a result, this method has two key characteristics: (i) the retrieved images are the most similar to the input x; (ii) it provides no labels, as image-to-image retrieving is associated to no textual prompt. Consequently, image-based works only with our soft labeling mechanism, as it provides no one-hot labels.

TIP-Adapter + IAP. This approach leverages TIP-Adapter [18] and IAP [9, 10] (indirect attribute prediction), using TIP-Adapter to predict the object's category, and IAP to recognize the attributes, conditioned on the predicted category. IAP was introduced to take into account the effect of the class (*i.e.*, object) on the attribute distribution, *i.e.*, to condition the probability of observing an attribute given a class. As a result, IAP exploits knowledge on the object category, thus treating attributes differently for different objects. IAP is defined as:

$$p(a_m|x) = \sum_{i=1}^{C} p(a_m|y_i)p(y_i|x), \tag{1}$$

where C is the number of classes and $p(y_i|x)$ is the probability of input image x of belonging to class y_i. We use TIP-Adapter to obtain this classification probability (*i.e.*, $p(y_i|x)$, with the cache built only for objects), while $p(a_m|y_i)$ is the probability distribution of attribute a_m of being observed together with y_i. This is estimated on CC12M, and it is the same that we use to construct our cache.

TIP-Adapter. We directly apply TIP-Adapter [18], as originally defined, to directly detect attributes. However, since it was originally designed for object classification, we adapt it to our task by constructing its cache on attributes, rather than objects. We populate its cache in various ways: sampling K shots per attribute ($|A| \times K$) and sampling K shots per attribute-object pair ($|O| \times |A| \times K$). These two alternatives cover both possible ways to adapt TIP-Adapter to our task. We note that we are unable to run the all attribute-object pairs ($|O| \times |A| \times K$) setting on VAW when $K > 1$, as memory requirements increase dramatically. We estimate some hundred gigabytes of RAM would be necessary to handle all the $2,260$ categories times 620 attributes times 16 shots per pair.

SuS-X. Similarly, we implement SuS-X [14], a successor of TIP-Adapter that leverages a more advanced inference formulation based on the KL-divergence, while computing the scores for an input x. We construct its cache by sampling K images for each attribute, thus filling it with $|A| \times K$ samples, similarly to the original implementation.

6.1. Full results

In Tab. 6, we propose and compare with seven baselines on OVAD and VAW. We test image-based by constructing a cache with (i) K samples and (ii) $|A| \times K$ samples, with $K = 16$ in both cases. Notably, the smaller cache performs better, scoring 16.8 mAP vs 9.4 mAP on OVAD and 52.9 mAP vs 31.5 mAP on VAW. We hypothesize this is due to the inaccuracies in the retrieval process, which considers solely

	Method	Configuration		OVAD [2]				VAW [12]			
		Type	Size	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
Zero shot	CLIP [5] RN50			11.8	41.0	11.7	1.4	35.3	37.8	35.1	26.2
	CLIP [5] ViT-B/32			17.0	44.3	18.4	5.5	50.0	51.0	50.9	43.2
	CLIP [5] ViT-L/14			18.3	44.4	20.5	6.4	51.0	51.4	52.5	45.0
Cache-based	Image-based	$\mathcal{I}$	K	16.8	45.8	19.2	3.5	52.9	53.7	53.8	46.4
		$\mathcal{I}$	$\lvert A\rvert \times K$	9.4	36.8	8.2	1.3	31.5	35.7	29.7	21.2
	TIP-Adapter [18] + IAP [10]	$\mathcal{O}$	$\lvert O\rvert \times K$	15.1	43.3	17.8	1.7	27.7	32.4	25.5	16.6
	TIP-Adapter [18]	$\mathcal{A}$	$\lvert A\rvert \times K$	16.7	44.4	19.7	3.1	57.5	57.5	59.4	51.2
		$\mathcal{AO}$	$\lvert O\rvert \times \lvert A\rvert$	9.8	38.3	8.9	0.9	53.4	53.8	54.8	47.7
		$\mathcal{AO}$	$\lvert O\rvert \times \lvert A\rvert \times K$	10.6	41.2	9.6	1.0	-	-	-	-
	SuS-X [14]	$\mathcal{A}$	$\lvert A\rvert \times K$	20.2	48.9	24.6	4.5	30.2	33.8	28.8	20.3
	ComCa	$\mathcal{AO}$	$\lvert A\rvert \times K$	**27.4**	**54.3**	**34.6**	**9.0**	**58.1**	**58.2**	**59.9**	**51.7**
Training based	OVAD [2]	Train size: 110K		21.4	48.0	26.9	5.2	-	-	-	-
	OvarNet [4]	Train size: 190k		28.6	58.6	35.5	9.5	68.5	-	-	-
	ArtVLM [19]	Train size: N/A		-	-	-	-	71.9	75.0	72.1	59.4
	LOWA [7]	Train size: 1.33M		18.7	58.0	20.4	2.6	42.6	46.4	41.0	32.9

Table 6. **Comparison with state of the art.** Extended version of Tab. 1 (*Main*), reporting (i) details on *head*, *medium*, and *tail* on OVAD and VAW, and (ii) details on the configuration used. All cache-based models are tested with CLIP ViT-B/32 [5] as backbone. Green indicates ComCa. For ArtVLM [19], we report its best scores. **Bold** indicates best among training-free methods. $\mathcal{I}$ refers to images, $\mathcal{O}$ to objects, $\mathcal{A}$ to attributes, and $\mathcal{AO}$ to attribute-objects.

the input image x, with no regard to the target attributes, *e.g.*, the object may dominate the retrieval content, producing a cache less informative w.r.t. attribute labels. The smaller cache, despite its higher similarity to the input, might contain less noisy samples, thus allowing our soft labeling scheme to produce better labels.

Next, we test IAP, using TIP-Adapter as our object classifier. The cache contains $|O| \times K$ elements, and we set $K = 16$. Its performance is lower than the image-based baseline's, as it achieves 15.1 mAP on OVAD (*i.e.*, -1.7 mAP w.r.t. image-based) and 27.7 mAP on VAW (*i.e.*, -25.2 mAP w.r.t. image-based). This is a consequence of the constrained attribute predictions, inherited from the object ones. As $p(a_m|y_i)$ depends only on the category, different objects within the same category will have the exact same attribute probability. While the input image x influences the probability $p(y_i|x)$ that it belongs to category y_i, it has no direct influence on $p(a_m|y_i)$. Therefore, if two objects with completely different visual attributes have the same class probabilities, this baseline will assign them the same attribute probabilities.

When including attributes in TIP-Adapter, we consider three distinct settings, covering various ways to extend its cache design to our task. We find that the smaller cache (*i.e.*, $|A| \times K$), having the same size as ComCa's, is the best-performing one. However, on OVAD it underperforms its backbone when used in the zero-shot setting (16.7 vs 17.0 mAP), while it surpasses it by +7.5 mAP on VAW.

Lastly, we compare with SuS-X, which outperforms all other baselines on OVAD, achieving 20.2 mAP, *i.e.*, +3.5 w.r.t. TIP-Adapter when constructing the cache in the same way. However, on VAW it struggles, scoring only 30.2 mAP, *i.e.*-27.3 mAP w.r.t. TIP-Adapter.

ComCa outperforms all these baselines on both benchmarks, with a gap of up to +18.0 mAP compared to image-based on OVAD and up to +27.9 mAP w.r.t. SuS-X on VAW. Specifically, on OVAD ComCa surpasses TIP-Adapter by +15.0 mAP on average over all three configurations of TIP, and SuS-X by +7.2 mAP. On VAW, ComCa has a gain over image-based of +26.6 mAP and on TIP-Adapter + IAP of +30.4 mAP. Compared to SuS-X, ComCa's gap is +27.9 mAP, and obtains an average increase of +2.65 mAP w.r.t. TIP-Adapter, always surpassing it.

7. Backbones

ComCa is a backbone-agnostic approach, therefore it supports virtually any vision-language model that can match text and images, similar to CLIP. Tab. 7 is the extended version of Tab. 2, reporting detailed results on all parts of the distribution of OVAD and VAW. Results are consistent with those of Tab. 2, with ComCa always improving the model it is applied to, across all metrics.

8. Box-free setting

As explained in Sec. 4.1, we primarily focus on attribute recognition, thus adopting the box-given setting as our competitors [2, 4]. In Tab. 3, we extend ComCa to the box-free

Method		OVAD [2]					VAW [12]				
Backbone	ComCa	mAP	Head	Medium	Tail	Δ_{mAP}	mAP	Head	Medium	Tail	Δ_{mAP}
CLIP RN50 [5]		11.8	41.0	11.7	1.4		35.3	37.8	35.1	26.2	
	✓	19.2	48.6	21.7	5.6	+7.4	51.1	51.8	52.4	44.2	+15.8
CLIP ViT-B/32 [5]		17.0	44.3	18.4	5.5		50.0	51.0	50.9	43.2	
	✓	27.4	54.3	34.6	9.0	+10.4	58.1	58.2	59.9	51.7	+8.1
CLIP ViT-L/14 [5]		18.3	44.4	20.5	6.4		51.0	51.4	52.5	45.0	
	✓	24.8	53.0	30.6	7.6	+6.5	61.0	60.5	63.5	55.6	+10.0
SigLIP ViT-B/16 [17]		13.7	41.5	15.2	2.0		52.5	52.1	54.6	47.8	
	✓	22.1	49.5	27.2	6.1	+8.4	59.8	59.0	62.4	54.7	+7.3
CoCa ViT-B/32 [15]		15.1	43.7	16.7	2.9		49.6	50.7	50.6	42.6	
	✓	24.4	50.7	31.7	6.1	+9.3	51.4	52.1	52.6	44.7	+1.8
CoCa ViT-L/14 [15]		14.4	42.8	15.4	3.0		49.8	50.7	50.9	42.9	
	✓	25.9	51.7	33.7	7.1	+11.5	54.5	54.7	56.2	48.3	+4.7
BLIP [11]		15.9	44.5	18.3	2.8		51.0	51.3	52.0	46.6	
	✓	22.7	51.3	27.3	7.0	+6.8	56.0	55.8	58.1	50.6	+5.0
X-VLM [16]		17.4	46.0	20.0	4.1		54.8	56.4	55.7	46.2	
	✓	28.4	54.8	37.7	7.5	+11.0	57.7	59.4	59.4	46.9	+2.9

Table 7. **Box-given results with different backbones.** Extended version of Tab. 2 (*Main*). Green indicates ComCa applied on top of the backbone. Δ indicates improvements w.r.t. the corresponding baseline.

setting by introducing an object detector. In Tab. 8, we show the complete version of Tab. 3, reporting the results on the *head*, *medium*, and *tail* distributions of OVAD. In addition, Tab. 8 reports results of all zero-shot and cache-based models with all five detectors of the YOLOv11 family [8], *i.e.*, from the smallest "N" to the largest "X". Specifically, we evaluate CLIP RN50 [5], CLIP ViT-B/32 [5], and CLIP ViT-L/14 [5] as zero-shot models. For cache-based models, we report results for "image-based" (see Sec. 4 and Sec. 6), IAP [6, 10], TIP-Adapter [18], SuS-X [14], and ComCa using CLIP ViT-B/32 as backbone.

Implementation details. We introduce an object detector $f_d : \mathcal{X} \rightarrow \textbf{box}$, where $\textbf{box} \in \mathbb{R}^{N\times 4}$ is the set of N predicted bounding boxes. We crop the input image on each predicted bounding box, obtaining N crops, and we input each cropped image to the model, as we do in the box-given setting.

We leverage the YOLOv11 family of object detectors, testing all five models, which have an increasing number of parameters: from 2.6M for "N", up to 56.9M for "X".

Results. In Tab. 8, we evaluate ComCa in the box-free setting with five different detectors and two different backbones. Zero-shot models generally underperform their corresponding box-given counterpart, with larger models (*e.g.*, CLIP ViT-L/14) scoring better performance than smaller models (*e.g.*, CLIP RN50). Interestingly, CLIP ViT-L/14 is the best-performing zero-shot model with all detectors when considering mAP, as performance ranges from 13.0 with YOLOv11N to 13.4 with YOLOv11X. However, it is always surpassed by other models on the *head* part of the distribution. For instance, CLIP RN50 outperforms it by +0.6 on *head* when using YOLOv11N.

When considering cache-based models, ComCa is the best-performing approach on all metrics, regardless of the backbone of choice. ComCa outperforms SuS-X by +6.0 mAP on average, across all object detectors. When compared with the other competitors, the gap further increases: on average, +11.8 mAP w.r.t. image-based, +11.3 mAP w.r.t. to TIP-Adapter + IAP, and +11.1 mAP w.r.t. TIP-Adapter.

When introducing ComCa, we observe a constant increase in performance as we increase the detector's size.

9. Additional ablations

In this section, we provide the extended versions of the ablation experiments presented in Sec. 4.2 (from Secs. 9.1 to 9.3), and we perform additional ablation studies to further evaluate the performance of ComCa. Extended results on cache construction are provided in Sec. 9.1, complementing the main findings in Tab. 4. Sec. 9.2 shows detailed results on all parts of the distribution of OVAD and VAW for the ablation on soft labels. In Sec. 9.3, we present results using varying numbers of samples per attribute, specifically more and less than 16, the value used in the main paper. Sec. 9.4 presents detailed results for the ablation on Eq. (7). Next, Sec. 9.5 shows how different LLMs influence performance when they provide *generative scores* to estimate attribute-object distri-

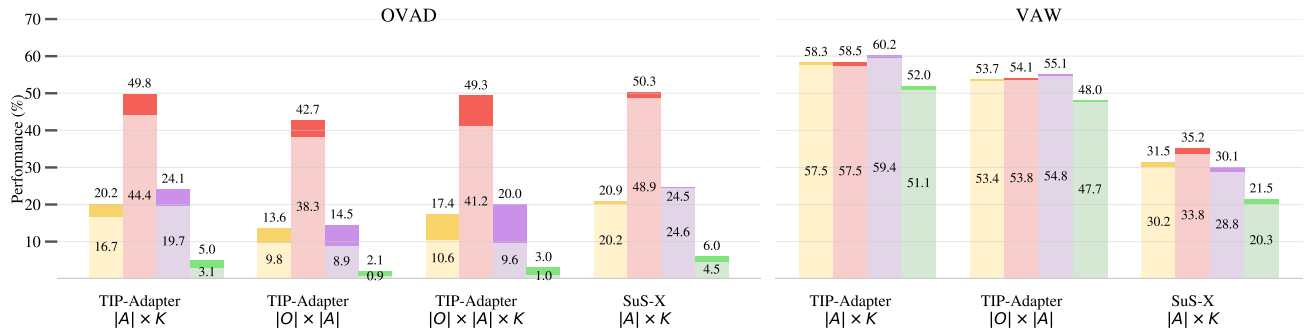

Figure 7. **Soft labels on cache-based baselines.** We apply our soft labeling mechanism to TIP-Adapter [18] and SuS-X [14]. Yellow represents mean average precision (mAP). Red represents performance on *head*, purple on *medium*, and green on *tail*. Brighter versions of these colors represent performance with soft labels. Number of shots K is always 16 if not specified otherwise.

	YOLOv11	Method	OVAD [2] mAP	Head	Medium	Tail
Zero-shot	N	CLIP RN50 [5]	12.0	39.2	12.5	1.8
		CLIP ViT-B/32 [5]	12.4	38.6	12.8	2.7
		CLIP ViT-L/14 [5]	13.0	38.6	12.9	2.7
	S	CLIP RN50 [5]	12.3	39.8	12.6	2.1
		CLIP ViT-B/32 [5]	12.4	39.3	12.8	2.4
		CLIP ViT-L/14 [5]	13.2	39.3	14.3	2.5
	M	CLIP RN50 [5]	12.3	40.2	12.7	1.9
		CLIP ViT-B/32 [5]	12.5	39.6	12.5	2.7
		CLIP ViT-L/14 [5]	13.4	39.7	14.3	2.9
	L	CLIP RN50 [5]	12.3	40.6	12.6	1.9
		CLIP ViT-B/32 [5]	12.4	39.9	12.5	2.5
		CLIP ViT-L/14 [5]	13.4	39.9	14.4	2.7
	X	CLIP RN50 [5]	12.3	40.7	12.7	1.8
		CLIP ViT-B/32 [5]	12.4	39.9	12.6	2.4
		CLIP ViT-L/14 [5]	13.4	39.9	14.5	2.6
Cache-based	N	Image-based	9.4	35.1	8.8	1.1
		TIP-Adapter [18] + IAP [10]	9.9	36.6	9.4	1.0
		TIP-Adapter [18]	10.1	35.6	9.9	1.3
		SuS-X [14]	14.9	41.4	16.7	3.4
		ComCa	20.2	46.3	24.8	5.4
	S	Image-based	9.4	35.3	8.8	1.0
		TIP-Adapter [18] + IAP [10]	9.9	36.7	9.2	1.0
		TIP-Adapter [18]	10.1	35.9	9.8	1.2
		SuS-X [14]	15.2	42.5	17.0	3.2
		ComCa	21.1	47.7	26.0	5.6
	M	Image-based	9.4	35.4	8.7	1.0
		TIP-Adapter [18] + IAP [10]	9.8	36.9	9.2	1.0
		TIP-Adapter [18]	10.1	36.0	9.8	1.2
		SuS-X [14]	15.3	43.0	17.2	3.2
		ComCa	21.5	48.4	26.4	6.0
	L	Image-based	9.4	35.6	8.7	1.0
		TIP-Adapter [18] + IAP [10]	9.9	37.0	9.2	1.0
		TIP-Adapter [18]	10.1	36.1	9.8	1.2
		SuS-X [14]	15.3	43.2	17.2	3.2
		ComCa	21.6	48.7	26.5	6.0
	X	Image-based	9.4	35.5	8.7	1.0
		TIP-Adapter [18] + IAP [10]	9.9	37.1	9.2	1.0
		TIP-Adapter [18]	10.1	36.2	9.7	1.2
		SuS-X [14]	15.4	43.5	17.2	3.1
		ComCa	21.7	49.0	26.6	6.1

Table 8. **Results in the box-free setting.** Extended version of Tab. 3 (*Main*). Green indicates ComCa. Experiments for cache-based models are conducted using CLIP ViT-B/32 [5] as backbone.

butions, as described in Sec. 3.2. Additionally, Sec. 9.6 replicates the ablation study conducted in TIP-Adapter [18] on the hyperparameter λ, thus studying if its effect is similar on ComCa's cache. In Sec. 9.7, we compare image retrieval (our default) with image generation. Lastly, Sec. 9.8 and Sec. 9.9 explore two key components of our algorithm: the hyperparameter α, which blends soft and hard labels, and the normalization of final predictions. Importantly, we highlight that α is set on VAW's validation set, while $\lambda = 1.17$ is the default value, taken from TIP-Adapter.

9.1. Cache construction

In Tab. 9, we report the complete results of the experiments presented in Tab. 4 (*Main*). We provide details on *head*, *medium*, and *tail* parts of the distributions of the OVAD and VAW benchmarks. We note that *Brute force* cannot be run on VAW due to the size of the target space: the cache would be too large, and it cannot be practically tested. Results are consistent with those of Tab. 4, showing the importance of attribute-object compatibility to build the cache, and with databases and LLMs providing complementary information.

We highlight that the cache construction process is fairly lightweight: image retrieval and soft scoring (Sec. 9.2) take ≈ 0.8 seconds per attribute and LLM-based scoring (Sec. 9.5) takes up to 10 seconds per attribute. Both of these operations occur *before* inference and take much less time compared to training-based methods (*e.g.*, 7 hours for ArtVLM [19]). During inference, using the cache adds only two matrix multiplications and one scaling operation (Eq. (10)), resulting in a negligible $+0.3\%$ time overhead.

9.2. Soft labels

Tab. 10 shows the complete results of the ablation on soft labels, presented in Tab. 5 (*Main*). The table includes scores on all parts of the distributions of OVAD and VAW. Results are consistent with those of Tab. 5, showing that soft labels always increase performance by a significant margin.

Configuration	OVAD [2]				VAW [12]			
	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
Zero-shot [5]	17.0	44.3	18.4	5.5	50.0	51.0	50.9	43.2
Random	16.7	44.4	19.7	3.1	**57.5**	**57.5**	**59.4**	**51.2**
Brute force	10.6	41.2	9.6	1.0	-	-	-	-
DB scores	18.6	46.8	22.5	3.7	54.1	53.7	55.8	50.1
LLM scores	21.1	51.5	25.0	5.6	55.4	56.0	57.0	47.9
DB + LLM scores	**26.4**	**52.5**	**33.7**	**8.3**	55.2	55.5	56.8	48.9

Table 9. **Ablation on cache construction.** Extended version of Tab. 4 (*Main*). Green indicates our configuration. **Bold** indicates the best for each column. Results are without *soft labels*.

Configuration	OVAD [2]				VAW [12]			
	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
Zero-shot [5]	17.0	44.3	18.4	5.5	50.0	51.0	50.9	43.2
One-hot	26.4	52.5	33.7	8.3	55.2	55.5	56.8	48.9
Soft	26.7	53.6	33.8	8.5	53.2	54.3	53.9	46.6
- Sharpening [1]	21.5	49.4	26.0	6.0	27.7	32.4	25.5	16.6
- Softmax	24.5	52.2	30.3	7.6	**58.5**	**58.7**	**60.3**	**52.4**
- Eq. (9)	**27.4**	**54.3**	**34.6**	**9.0**	58.1	58.2	59.9	51.7

Table 10. **Ablation on soft labels.** Extended version of Tab. 5 (*Main*). Green indicates our default configuration. **Bold** indicates the best for each column.

Configuration	OVAD [2]				VAW [12]			
	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
Baseline zero-shot [5]	17.0	44.3	18.4	5.5	50.0	51.0	50.9	43.2
$K = 1$	21.8	49.2	26.7	6.1	**58.4**	**58.6**	60.2	52.2
$K = 2$	22.8	50.7	27.8	6.8	**58.4**	**58.6**	**60.3**	51.8
$K = 4$	24.3	51.9	29.9	7.7	**58.4**	58.5	**60.3**	**52.1**
$K = 8$	26.0	**54.6**	32.2	8.2	**58.4**	**58.6**	60.2	51.9
$K = 16$	**27.4**	54.3	**34.6**	**9.0**	58.1	58.2	59.9	51.7
$K = 24$	27.1	**54.6**	**34.6**	8.2	57.8	58.1	59.4	51.6

Table 11. **Ablation on the number of shots.** Extended version of Fig. 5 (*Main*). Green indicates our default configuration. **Bold** indicates the best for each column.

Configuration	OVAD [2]				VAW [12]			
	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
None	24.0	52.5	29.6	7.1	57.9	57.8	59.9	52.5
Sum	24.8	52.5	31.8	6.4	**58.3**	**58.4**	**60.1**	**52.0**
Multiplication – Eq. (7)	**27.4**	**54.3**	**34.6**	**9.0**	58.1	58.2	59.9	51.7

Table 12. **Ablation on Eq. (7).** Results of CoMCa on OVAD using no CC12M prior (LLM only), CC12M combined with the LLM scores using sum and multiplication, as in Eq. (7) (our default configuration). **Bold** indicates the best results.

Prior		OVAD [2]			
LLM	CC12M	mAP	Head	Medium	Tail
Gemma 7b		21.0	52.2	24.8	5.3
	✓	22.8	54.7	27.0	6.3
LLaMa 2 7b		21.4	50.4	25.9	5.6
	✓	23.2	52.4	28.2	6.8
LLaMa 3 8b		22.8	53.8	27.3	6.2
	✓	24.1	54.8	29.0	7.1
Mistral		23.9	52.8	29.4	7.0
	✓	24.6	53.2	30.5	7.2
GPT 3.5 Turbo		24.0	52.5	29.6	7.1
	✓	**27.4**	**54.3**	**34.6**	9.0
GPT 4o-mini		23.6	50.9	28.5	7.9
	✓	25.4	53.0	30.3	**9.7**

Table 13. **Ablation on LLM backbones.** Comparison of different LLMs to obtain the attribute-object compatibility score. ComCa is tested by using LLM scores only, and combining them with the prior extracted from CC12M [3]. Green indicates our default configuration. **Bold** indicates the best performance.

λ	OVAD [2]			
	mAP	Head	Medium	Tail
CLIP only	17.0	44.3	18.4	5.5
0	20.6	48.7	25.1	5.3
0.5	25.7	53.0	32.0	8.2
1.0	27.1	54.1	34.1	**9.0**
1.17 (TIP default)	**27.4**	**54.3**	34.6	**9.0**
2.0	**27.4**	**54.3**	**34.9**	8.7
3.0	27.2	54.1	34.8	8.4
4.0	27.1	53.9	34.7	8.2
Cache only	25.9	52.8	33.6	6.9

Table 14. **Ablation on hyperparameter λ.** We replicate TIP-Adapter's [18] ablation on λ to determine if ComCa's cache behaves similarly. Green indicates the default configuration for ComCa, the same used in TIP-Adapter. **Bold** indicates the best performance.

In addition, we evaluate and demonstrate the effectiveness of our soft-labeling mechanism by applying it to the TIP-Adapter and SuS-X baselines in Fig. 7. We evaluate the same four settings as in Tab. 6. Due to the impossibility of running TIP-Adapter in the $|O| \times |A| \times K$ setting, with $K = 16$, as explained in Sec. 6, we omit the plot in the figure.

We note that our soft labels *always* improve mAP performance, with an average of +3.7 on OVAD and +0.8 on VAW. Notably, it often leads to large performance increases on the *medium* part of the distribution (up to +10.4), and it provides significant boosts on the *tail* (up to +2.0).

9.3. Number of shots

We evaluate our model's performance with different numbers of samples per attribute in the cache. Tab. 11 provides an extended version of the results shown in Fig. 5, illustrating that more samples generally improve performance. On OVAD [2], performance increases steadily from $K = 1$ to $K = 16$, with a slight loss at $K = 24$. In contrast, VAW [12] is less sensitive to the number of samples. We hypothesize that this is due to our soft labeling mechanism, which may yield diminishing returns as the number of attributes increases (VAW has 620 attributes). Results are consistent with those shown in Fig. 5 and confirm the ablation on the number of shots shown in TIP-Adapter [18], where a modest cache of 16 elements contains all the necessary information.

9.4. Prior combination

In Tab. 12, we assess changes in performance when changing the interaction between the two priors – database and LLM. We study three case, *None*, *Sum*, and *Multiplication*, conducting experiments on both OVAD and VAW. In *None*, we discard the database prior, *i.e.*, the compatibility scores extracted from CC12M. In *Sum*, we sum the scores produced by the LLM to the scores extracted from the web-scale database, while in *Multiplication*, we follow our definition as in Eq. (7) (refer to *Main*).

None is the worst strategy, because it excludes the information from the web-scale database. This is supported by the results presented in Fig. 6 in Sec. 4.1 in the *Main* paper. When introducing the retrieval scores with *Sum*, we observe an increase in performance: +0.8 mAP on OVAD and +0.4 mAP on VAW. *Multiplication* largely outperforms *Sum* on OVAD, surpassing it by +2.6 mAP (+3.4 mAP w.r.t. *None*), while it performs slightly worse on VAW, scoring 58.1 mAP (-0.2 w.r.t. *Sum*). However, it is superior to *None*, with a gap of +0.2 mAP.

9.5. Different LLMs

All experiments in the main paper, along with ablations in both the *Main* and *Supp. Mat.*, use GPT 3.5 Turbo as the LLM for generative scores (Eq. (6)). In Tab. 13, we replace GPT 3.5 Turbo both with GPT 4o-mini and with publicly available LLMs that can be run locally. For computational efficiency, we select models in the 7-8 billion parameters range: Gemma 7b, LLaMa 2 7b, LLaMa 3 8b, and Mistral. We evaluate each LLM's performance using its scores alone and in combination with retrieval scores (Eq. (5)), as described in Eq. (7). GPT 4o-mini underperforms GPT 3.5 Turbo, which surpasses it by +2.0 mAP. However, GPT 4o-mini outperforms all four open-source small models, with an average gap of +1.7 mAP. Although open source models are surpassed by the GPT family of closed models, they represent a valid open-source and cost-effective alternative.

Configuration	OVAD [2]			
	mAP	Head	Medium	Tail
Retrieval (CC12M)	27.4	54.3	34.6	9.0
Generation (Stable Diffusion XL)	**28.0**	**57.2**	**34.9**	**9.1**
Δ	+0.6	+2.9	+0.3	+0.1

Table 15. **Ablation on cache construction.** Results on OVAD when constructing the cache with retrieval or generation. Green indicates our default configuration, which uses retrieval to populate the cache. **Bold** indicates the best performance. Δ indicates the difference between generation and retrieval.

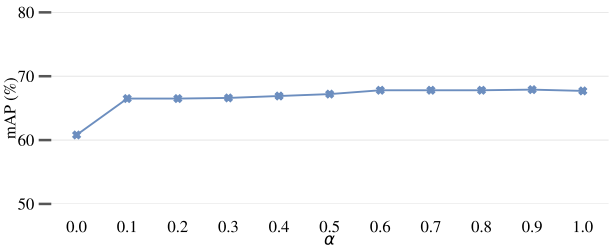

Figure 8. **Ablation on hyperparameter α.** The blue line is mAP on the validation set of VAW. The higher α, the larger the contribution of soft labels.

In all cases, combining both sources yields better results, with mAP gains of $+1.8$ mAP for Gemma 7b, $+1.8$ mAP for LLaMa 2 7b, $+1.3$ mAP for LLaMa 3 8b, $+0.7$ mAP for Mistral, and +1.8 mAP for GPT 4o-mini, thus reinforcing the findings presented in Fig. 6 (*Main*).

9.6. Combining CLIP and cache scores

By default, we use the same value of $\lambda = 1.17$ as in TIP-Adapter [18], but we analyze the effect of λ in Eq. (11), where it is used to linearly combine vanilla CLIP scores with cache scores. We evaluate its impact on ComCa in Tab. 14. The first row, *CLIP only*, reports the results for the vanilla CLIP model, as seen in our baseline in Tabs. 1 and 2 (*CLIP ViT-B/32*). In the subsequent rows, increasing λ from 0 to 4 shows performance gains up to $\lambda = 2.0$, with a slight decline decline afterward. Finally, *Cache only* represents using only cache scores, which, while lower than the combined approach, still outperforms *CLIP only* by $+8.9$ mAP.

9.7. Image retrieval vs generation

Our approach involves populating the cache with images from CC12M [3], selected based on cosine similarity to a text query describing an object and an attribute. Alternatively, one could use a generative model, such as Stable Diffusion, to create samples from the same query. To compare these methods, we conducted an ablation study contrasting web-scale image-to-text retrieval with text-to-image generation using diffusion models. Results, shown in Tab. 15, reveal no significant performance difference, with a 0.6 mAP gap on OVAD favoring generation. Although retrieval is slightly less effective, it is far more computationally efficient.

9.8. Alpha blending

We evaluate the effect of the α parameter, which controls the blending of one-hot and soft labels, as outlined at the end of Sec. 3.2. We set α to 0.6 on the validation set of VAW, thus assigning a weight of 0.6 to soft labels and $1 - 0.6 = 0.4$ to hard labels.

To understand the effect of this parameter, we perform experiments on the validation set of VAW [12]. As shown in Fig. 8, incorporating soft labels improves performance. Starting at $\alpha = 0$, which uses only one-hot labels, we see a rapid increase in performance, peaking at 67.8 mAP when $\alpha = 0.6$. Beyond this point, performance remains constant, slightly decreasing to 67.7 mAP with $\alpha = 1.0$.

We hypothesize this behavior is due to the nature of one-hot and soft labels. One-hot labels focus solely on the attribute in the prompt, ignoring all other attributes in the image. Relying only on hard labels reduces the task to a multiclass problem, where only one attribute is considered positive and the rest negative. Conversely, using only soft labels yields results similar to CLIP baselines, as both approaches would use the same scoring mechanism. The best performance is achieved by balancing the contributions of both hard and soft labels.

9.9. Cache scores normalization

In Tab. 16, we analyze the effect of scaling the scores from cache-based predictions by comparing three strategies: no normalization (`no norm`, as in [18]), `min-max` normalization (subtracting the minimum score and dividing by the maximum), and our approach of scaling by the maximum value (`max`) used in Eq. (11). Additionally, we apply softmax within η_A to ensure numerical stability and produce a proper probability distribution. The results, shown at the bottom of Tab. 16, indicate that no normalization yields poor performance (8.7 mAP on OVAD, 40.3 mAP on VAW), while `min-max` improves results (25.9 mAP on OVAD, 56.6 mAP on VAW). Our method achieves the best performance overall, with 27.4 mAP on VAW and 58.1 mAP on VAW.

9.10. Seen vs unseen bias

We note that by omitting the training stage (i) ComCa is not exposed to training biases and (ii) it is impractical to

Configuration	OVAD [2]				VAW [12]			
	mAP	Head	Medium	Tail	mAP	Head	Medium	Tail
Baseline zero-shot [5]	17.0	44.3	18.4	5.5	50.0	51.0	50.9	43.2
`no norm`	8.7	36.0	7.4	0.6	40.3	43.1	39.6	31.7
`min-max norm`	25.9	53.4	32.1	8.67	56.6	56.9	58.5	49.7
`max norm, softmax`	27.4	54.3	34.6	9.0	58.1	58.2	59.9	51.7

Table 16. **Ablation on scores normalization**. Results with different cache score normalization strategies. Green indicates the default configuration for ComCa.

identify the seen/unseen split, as everything is technically unseen. We use VAW's unseen split from [4] to directly compare with OvarNet. ComCa shows more consistent results (58.4 mAP on seen, 56.9 on unseen) than OvarNet (69.8 mAP on seen, 56.4 on unseen). As potential biases may come from the cache, we check the performance on the object-attribute compositions there stored, considering as "seen" those present in the cache. ComCa scores 56.5 mAP (seen) and 58.2 (unseen) on VAW, and 44.8 mAP (seen) and 27.9 (unseen) on OVAD. Note that the difference in performance between the two sets in OVAD follows the zero-shot performance of the base model (*e.g.*, 37.8 and 18.8 mAP) rather than cache-specific biases.

10. Additional qualitative results

In Fig. 9, we provide additional qualitative results to those presented in Fig. 3 in *Main*. We compare OVAD and a vanilla zero-shot CLIP ViT-B/32 with ComCa, based on the same architecture, similar to Fig. 3. Firstly, we notice that OVAD often recognizes opposite attributes, in particular struggling with color. For instance, in Fig. 9a it detects "green", "white", "red", and "yellow" for the tennis racket and "green", "yellow", and "red" for the apple. In addition, it struggles with materials, for example by predicting "paper, cardboard" for the mobile phone (Fig. 9a) and "leather" for the skateboard (Fig. 9c). Similarly, we observe that also vanilla CLIP is prone to predicting contradictory attributes, such as "cooked, baked, warmed" and "raw, fresh" for the cake, or "multicolored" and "single-colored" for the kite. Moreover, CLIP seems to disregard the part of the object an attribute is bound to: for instance, the apple is correctly predicted as "green", but also "hair color" is predicted as "green". Similarly, the desktop monitor in Fig. 9b is predicted as "white" three times: for "color" (correct), "hair color" (incorrect), and "clothes color" (incorrect). On the other hand, ComCa has a finer understanding of attributes and detects them more effectively, as demonstrated quantitatively in all our experiments. Notably, it predicts "two-colored" for the mobile phone in the leftmost picture in Fig. 9a: although it is wrong according to the ground truth, we argue that indeed it could be considered two-colored (white and pink). Similarly, it predicts "full, whole" for the skateboard in Fig. 9c: although that attribute is not annotated for that specific object (thus we mark it in yellow), we argue that it would be a correct prediction.

11. Details on prompts

We utilize five types of prompts: (i) for retrieval, to construct queries for web-scale datasets; (ii) for image generation; (iii) for soft labeling; (iv) for inference, to define prompts for the attributes of interest; and (v) to obtain attribute-object scores via LLMs.

Retrieval. For retrieval, we construct prompts using the template `A photo of {noun} that is {attribute}`, following OvarNet [4]. For example, when retrieving images for a `red car`, the prompt will be `A photo of car that is red`.

Image generation. To generate images we employ the following template: `a {attr} {noun} on a clear background, hyperrealistic, highly detailed, sharp focus, 4k` for attributes of type "is", as defined in OVAD [2], and the following template: `a {noun} with {attr} {obj} on a clear background, hyperrealistic, highly detailed, sharp focus, 4k` for attributes of type "has".

Soft labeling. To compute soft labels, ComCa first computes the similarity between the target attributes and the images in the cache. Following [13], we encapsulate attributes into templates to improve performance and reduce noise. We leverage the same templates of OVAD [2]. For "has" type attributes, we use the following prompts:

- `a {attr} {obj} {noun}`
- `a {noun} has {attr} {obj}`

For "is" type attributes, we use the following prompts:

- `a {attr} {noun}`
- `a {noun} is {attr}`

OVAD	color: white pattern: lettered material: paper, cardboard	color: green color: white color: red color: yellow	color: green color: yellow color: red state: open
CLIP	smooth, sleek clean, neat asphalt, cement, clay, concrete, stucco	single, one striped, lined clothes pattern: tiled clothes pattern: dotted, speckled, spotted	color: green hair color: green raw, fresh face expression: disgust, frowning
ComCa	vertical, upright, standing two-colored single, one, individual, sole	horizontal single, one, individual, sole smooth, sleek two-colored	full, whole single-colored single, one raw, fresh

(a) Comparison of performance on a mobile phone, a tennis racket, and an apple.


OVAD	color: white color: red material: glass state: closed	color: white clothes color: white state: wet state: dry	state: closed state: open color: white color: gray
CLIP	cooked, baked, warmed raw, fresh smooth, sleek	opaque single-colored young, baby color: white	opaque hair color: white clothes color: white color: white
ComCa	vertical, upright, standing two-colored single, one, individual, sole	full, whole vertical, upright, standing single, one, individual, sole two-colored	state: on single, one, individual, sole vertical, upright, standing single-colored

(b) Comparison of performance on a cake, a frisbee flying disc, and a PC monitor.



OVAD	state: open sitting, sit state: on	multicolored, colorful color quantity: two-colored big, large, giant, huge	group, bunch, collection male, man, guy, boy material: leather
CLIP	sitting, sit dotted, speckled, spotted	multicolored, colorful single-colored textile, cloth, fabric, denim	dotted, speckled, spotted adult, old, aged
ComCa	state: on single, one, individual, sole	horizontal clean, neat smooth, sleek	full, whole single, one

(c) Comparison of performance on a laptop, a kite, and a skateboard.

Figure 9. **Additional qualitative results.** Top positive predictions of OVAD, CLIP and ComCa on sample images from OVAD. Green are correct predictions, red are wrong ones.

As VAW [12] does not provide *is/has* type annotation, we use "is" type templates. Similarly to OVAD [2], we utilize the word "object" as the general noun to bind attributes to while computing these soft cache scores.

Inference. Following OvarNet [4], we construct prompts for inference by using the following template: *A photo of something that is* {*attribute*} . For example, for the attribute "clean" the query produced at inference time will be `A photo of something that is clean`.

Object-attribute scoring via LLM. When utilising LLMs to generate object-attribute compatibility scores we carefully instruct the model to ensure it provides sensible results. Specifically, we utilize the following template:

> Let's play a role game. You will play the role of a researcher who is both a statistician and linguist. I will interpret a silly student who has many questions regarding language and statistics of language.
>
> In particular, I will ask you to tell me which classes, or categories if you prefer, match or bind well with the attribute I will provide you. More precisely, you will have to tell me if each class/category that I will give you matches well the given attribute. You should also tell me how well they match on a scale 0 (the class cannot have the attribute) to 10 (the class can have the attribute and it is semantically fine to associate the attribute to the class).
>
> Your response should list all the {count_categories} classes, and provide for each one of them the score on the scale explained above. The output format should be 'class: score'. No explanation at all, just plain output.
>
> Additional rules:
> - do not provide any outputs but the list of chosen categories
> - the output must be in the form of "x. category: score", where 'x' is the index of the category
> - the output must be in the form of a list
> - make sure you provide a score for each category. There are {count_categories} categories, so the output list must have {count_categories} elements.
>
> There are {count_categories} classes (categories). The list of classes, or categories, is the following: {categories}
>
> The attribute is: {attribute}.

Note that this template has different parameters:

`{count_categories}` How many categories the prompt will contain.

`{categories}` The list of categories to evaluate. More precisely, they are passed as a numbered list, with one category per line.

`{attribute}` The attribute to which categories have to be bound for evaluation.

We empirically find that the LLM may struggle when handling hundreds or thousands of categories, as in the case of VAW [12]. Therefore, we find it useful to split the categories in multiple sets, construct independent prompts, and ask the model to respond to batches of categories separately, thus ensuring its context window is not filled.

12. Resources used

Our training-free method does not require extensive computational resources and is designed to be computationally efficient. We run all our experiments on a NVIDIA RTX 2080Ti GPU with 12GB of VRAM. We use up to 4 NVIDIA RTX A6000 GPUs with 48GB of VRAM for the experiments that involve image generation, and are therefore more resource-demanding. Specifically, we use this configuration to generate images for the ablation in Tab. 15, which uses Stable Diffusion XL to generate images. Evaluation time on the NVIDIA RTX 2080Ti GPU takes approximately 4 minutes on OVAD and 10 minutes on VAW.

13. Limitations

We presented the first approach for open-vocabulary training-free attribute detection. While effective, our method can be improved in several directions. First, ComCa relies on an external database to generate the cache, so if this database does not contain a category, we cannot sample/bind it to an attribute. In some applications, using a general-purpose web-scale dataset might not suffice. Second, ComCa relies on a large language model (LLM). If the LLM generates inaccurate or incorrect information, known as hallucinations, it could adversely affect the overall performance of the method. Finally, our experimental evaluation demonstrates that ComCa is competitive with training-based approaches and more effective when we aim to generalize to other datasets and domains (see Fig. 4 in *Main*). However, training-based approaches may still be preferable if resources in terms of large scale datasets and computing infrastructure are available.

References

[1] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *ICCV*, 2021. 5

[2] Maria A Bravo, Sudhanshu Mittal, Simon Ging, and Thomas Brox. Open-vocabulary attribute detection. In *CVPR*, 2023. 2, 3, 4, 5, 6, 7, 8, 10

[3] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*, 2021. 6, 7

[4] Keyan Chen, Xiaolong Jiang, Yao Hu, Xu Tang, Yan Gao, Jianqi Chen, and Weidi Xie. Ovarnet: Towards open-vocabulary object attribute recognition. In *CVPR*, 2023. 2, 8, 10

[5] 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*, 2023. 2, 3, 4, 5, 8

[6] Ali Farhadi, Ian Endres, and Derek Hoiem. Attribute-centric recognition for cross-category generalization. In *CVPR*, 2010. 3

[7] Xiaoyuan Guo, Kezhen Chen, Jinmeng Rao, Yawen Zhang, Baochen Sun, and Jie Yang. Lowa: Localize objects in the wild with attributes. In *NeurIPS-WS*, 2023. 2

[8] Glenn Jocher and Jing Qiu. Ultralytics yolo11. 2024. 3

[9] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 1

[10] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE TPAMI*, 36(3):453–465, 2013. 1, 2, 3, 4

[11] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. 3

[12] Khoi Pham, Kushal Kafle, Zhe Lin, Zhihong Ding, Scott Cohen, Quan Tran, and Abhinav Shrivastava. Learning to predict visual attributes in the wild. In *CVPR*, 2021. 2, 3, 5, 6, 7, 8, 10

[13] 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 *ICML*, 2021. 8

[14] Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. Sus-x: Training-free name-only transfer of vision-language models. In *ICCV*, 2023. 1, 2, 3, 4

[15] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *TMLR*, 2022. 3

[16] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-grained vision language pre-training: Aligning texts with visual concepts. *ICML*, 2022. 3

[17] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 3

[18] 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*, 2022. 1, 2, 3, 4, 6, 7

[19] William Y Zhu, Keren Ye, Junjie Ke, Jiahui Yu, Leonidas Guibas, Peyman Milanfar, and Feng Yang. Artvlm: Attribute recognition through vision-based prefix language modeling. *ECCV*, 2024. 2, 4