Rounded or Streamlined Head? Bridging Concept Bottleneck Models and Attribute-Described Object Parts

Supplementary Material

Dataset	number of classes	number of Images	
		Train	Dev
Food-101	101	70,700	30,300
FLOWER-102	102	5726	2,463
CUB-200-2011	200	5,994	5,794
CIFAR-100	100	50,000	10,000
ImageNet	1,000	1,281,167	50,000

Table 8. Detailed statistics of the classification datasets.

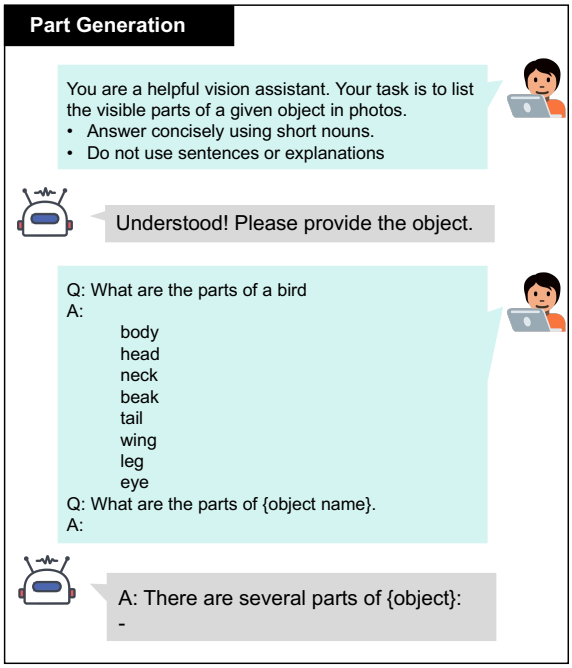

Figure 6. In-Context Learning Prompt used for generating parts.

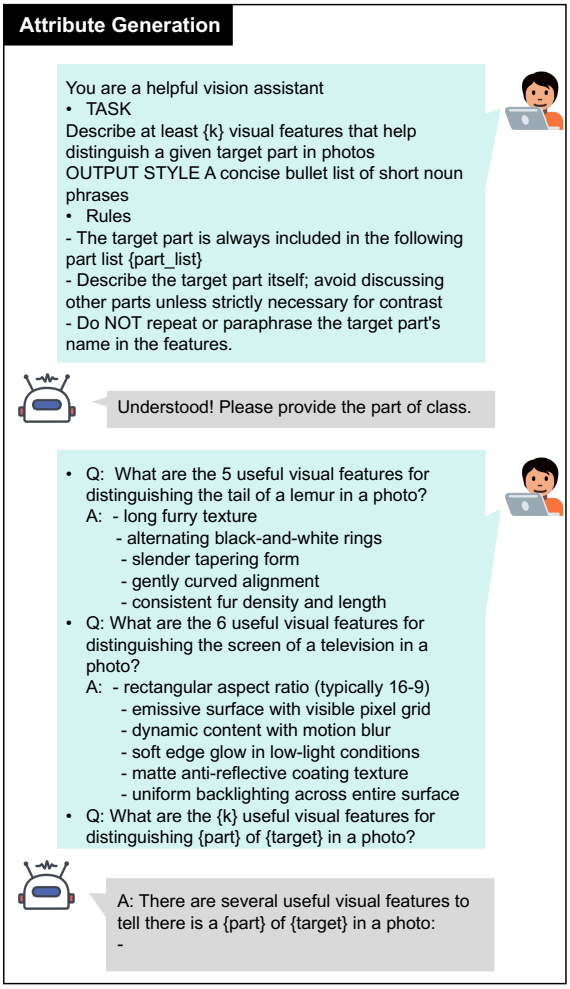

Figure 7. In-Context Learning Prompt used for generating attributes.

Superclass	Object Categories
Quadruped	tiger, giant panda, leopard, gazelle, **ice bear**, **impala**, **golden retriever**
Snake	green mamba, **Indian cobra**
Reptile	green lizard, Komodo dragon, tree frog, **box turtle**, **American alligator**
Boat	yawl, pirate, **schooner**
Fish	barracouta, goldfish, killer whale, **tench**
Bird	albatross, goose, **bald eagle**
Car	garbage truck, minibus, ambulance, **jeep**, **school bus**
Bicycle	mountain bike, moped, **motor scooter**
Biped	gorilla, orangutan, **chimpanzee**
Bottle	beer bottle, water bottle, **wine bottle**
Aeroplane	warplane, **airliner**

Table 9. Superclass and Object Categories

A. Resources

A.1. Classification dataset

Table 8 provides detailed statistics for all datasets. Following the standard practice, we adopt the official train/dev splits for each dataset.

A.2. Construction of Concept Set.

Previous work [4, 58, 61] defines concepts as attributes of specific objects and prompts large language models (LLMs) to generate object-related concepts. However, this approach may lead to incomplete coverage and often results in visu-

ally irrelevant or noisy concepts. This issue is particularly prominent in the generative paradigm of Labo [61]. To obtain more fine-grained visual concepts, we redefine the concept in a 'part-attribute' format. The basic steps are as follows:

- *Part generation:* For each object, we use an In-Context Learning (ICL) prompt to improve the LLM's ability to generate appropriate parts, as shown in Figure 6. Specifically, we provide several ICL examples and then prompt the LLM to output the set of parts associated with the given object.
- *Attribute generation:* Using the generated parts and the object name, we prompt the LLM to produce fine-grained part–attribute descriptions, as illustrated in Figure 7. In this step, human-defined concepts can also be included as ICL exemplars to guide the attribute generation process and improve consistency.
- *Concept Deduplication:* After obtaining a set of part-attribute concepts, we perform concept deduplication using SemHash. This utilizes Model2Vec to efficiently generate embeddings for any text in any language. Duplicate concepts are filtered out if their similarity score exceeds 0.9.
- *Concept Filtering:* Given a concept and an image-caption dataset, let X represent the image-concept similarity as measured by a Vision-Language Model (VLM), and Y be a binary indicator for caption-concept correspondence based on a Large Language Model (LLM). We compute the Mutual Information (MI) between X and Y as:

$$MI(X, Y) = H(X) - H(X|Y)$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the conditional entropy of X given Y. Based on the desired size of the concept set, we select the top-K concepts with the highest mutual information. We test different numbers of categories for each dataset to determine the optimal size of the concept set.

A.3. Concept Grounding Dataset

We collect two publicly available part-segmentation datasets: PartCUB-70 [3] and PartImageNet [18]. We follow the process outlined in Section A.2 with minor modifications, such as not needing to generate object parts, as the part-segmentation datasets already provide them.

PartAttrImageNet. PartImageNet [18] contains 16k training images and 2.9k validation images, segmented into 158 object classes from ImageNet, which are organized into 11 super-categories. Following the protocol in [9] and as shown in Table 9, we select 40 classes out of the 158 to measure more generalized performance. We maintain the proportion of existing superclasses as much as possible. For each superclass, at least 50% of the categories are designated as seen categories, with the remaining categories as unseen. This results in 25 seen classes and 15 unseen classes in our PartImageNet evaluation dataset. Based on this, we construct the PartAttrImageNet concept grounding dataset following the process outlined in Section A.2.

PartAttrCUB. Based on PartCUB-70, we construct the PartAttrCUB concept grounding dataset. PartCUB-70 is a subset derived from the first 70 classes of the CUB-200 dataset. For this subset, pixel-wise annotation masks were manually produced for 11 parts: head, right eye, left eye, beak, neck, body, right wing, left wing, right leg, left leg, and tail. We consolidate these into 8 unified parts: head, eye, beak, neck, body, wing, leg, and tail. The construction process is as follows:

- **Step 1:** Prompting GPT-5 to generate k attributes for each part, as shown in Figure 7.
- **Step 2:** Perform concept deduplication and filtering as described in Section A.2.
- **Step 3:** Each concept is organized into a tree structure: object–part–attribute. For part-attribute combinations that cannot distinguish the object, we merge them into a single class.
- **Step 4:** Unlike PartAttrImageNet, we collect all 70 bird species and prompt GPT for each part. We then perform clustering based on visual features as part-object clustering, since it is not always possible to distinguish species based on a single part.

Our part–attribute concept set contains 8 parts (body, head, neck, beak, tail, wing, leg, eye), and each part includes multiple fine-grained visual groups. Each group is defined by:

- **id**: unique identifier of the visual group;
- **label**: human-readable cluster name;
- **categories**: the list of bird species belonging to this group;
- **concept**: a set of 5 textual, visual-groundable attributes.

Due to the large size of the full taxonomy, we include a small selection of representative examples below. The complete JSON file is provided in the supplementary materials and our project page.

Example Concept (Body)


```
{
  "id": 1,
  "label": "Medium gray back + white belly gulls",
  "categories": ["California Gull", "Herring Gull",
                 "Western Gull"],
  "concept": [
    "mid-gray mantle over clean white underparts",
    "bulky sea-gull mass",
    "smooth two-tone paneling"
  ]
}
```

Backbone	Concept Grounding on PartAttrImageNet									Concept Grounding on PartAttrCUB								
	Oracle-Obj			Oracle-All			Pred-All			Oracle-Obj			Oracle-All			Pred-All		
	Seen	Unseen	h-IoU	Seen	Unseen	h-IoU	Seen	Unseen	h-IoU	Seen	Unseen	h-IoU	Seen	Unseen	h-IoU	Seen	Unseen	h-IoU
CLIP	66.4	38.9	49.1	47.8	16.7	24.8	46.9	15.1	22.9	35.3	25.2	29.4	13.0	11.9	12.4	13.1	7.8	9.8
DINOv2TXT	72.3	38.6	50.3	55.8	20.5	30.0	56.4	21.8	31.5	39.1	43.7	41.3	10.0	15.2	12.1	6.0	4.9	5.4
DINOv3TXT	71.3	46.1	56.0	49.9	24.8	33.1	49.8	25.8	34.0	66.8	25.7	37.1	58.5	28.2	38.0	30.2	28.5	29.3
SC-CLIP	31.8	39.2	35.1	18.5	21.4	19.9	18.5	20.9	19.6	25.4	17.3	20.6	11.3	8.8	9.9	7.1	7.2	7.1
LLM-DINO	74.0	39.7	57.8	52.2	14.4	22.6	50.9	14.3	22.3	59.4	22.3	32.4	33.3	8.3	13.3	32.4	0.5	1.0
LLM-CLIP	60.3	44.8	51.4	46.2	30.8	37.0	44.1	29.9	35.7	48.7	26.0	33.9	28.0	22.0	24.7	19.5	22.9	20.8

Table 10. Comparison of zero-shot performance with different backbones on our concept segmentation datasets. mIoU (%) under Oracle-Obj, Oracle-All and Pred-All protocols. h-IoU is the harmonic mean of Seen/Unseen.

Backbone (m-IoU)	Part Grounding			Obj
	Oracle-Obj	Oracle-All	Pred-All	
CLIP	56.4	54.7	51.7	91.8
DINOv2TXT	67.6	62.2	60.3	93.8
DINOv3TXT	66.6	58.9	56.7	93.8
SC-CLIP	44.8	40.5	37.6	91.3
LLM-DINO	69.2	65.3	62.3	94.4
LLM-CLIP	50.5	48.3	46.5	83.1

Table 11. Comparison of zero-shot part-only performance with various backbones on PartAttrImageNet datasets. mIoU (%) under Oracle-Obj, Oracle-All and Pred-All protocols.

Backbone (m-IoU)	Part Grounding			Obj
	Oracle-Obj	Oracle-All	Pred-All	
CLIP	53.7	50.3	44.0	89.1
DINOv2TXT	55.3	50.5	41.2	85.1
DINOv3TXT	74.6	73.3	28.7	68.1
SC-CLIP	52.0	49.6	17.8	66.1
LLM-DINO	71.4	69.1	33.0	71.0
LLM-CLIP	62.5	48.3	46.5	83.1

Table 12. Comparison of zero-shot part-only performance with various backbones on PartAttrCUB datasets. mIoU (%) under Oracle-Obj, Oracle-All and Pred-All protocols.

Example Concept (Head)

```
{
  "id": 3,
  "label": "White → lightly streaked gull heads",
  "categories": ["Ivory Gull", "California Gull"],
  "concept": [
    "immaculate to near-clean white crowns",
    "seasonal fine brown penciling",
    "crisp white forehead plane",
    "high-key face with soft nape fade"
  ]
}
```

Quality control. involves two key tasks: ambiguity checks and manual spot audits on held-out samples.

- *Ambiguity checks:* For each generated part-attribute concept, we perform consistency checks by comparing the model's generated concepts to their corresponding parts in the image. We manually validate whether the concept aligns with the intended object part (e.g., head, body, tail), ensuring there are no ambiguous associations. Additionally, we use similarity-based tools like SemHash to automatically detect concept duplication or conflicts. Concepts with a similarity score above 0.9 are flagged for further review.
- *Manual spot audits:* A random selection of held-out samples is manually reviewed to ensure data quality. For each sample, the part-attribute concepts are checked for accuracy and consistency. Any discrepancies are flagged for correction. This manual review process helps assess the generalization of the model across unseen data and ensures that the part-attribute mappings are reliable.

Zero-shot Setting. We conduct the dataset evaluation as follows: Models are trained on a training dataset composed of seen classes. Segmentation performance are then assessed on a validation dataset containing both seen and unseen classes. Evaluations were conducted in both and Oracle-Obj, Oracle-All and Pred-All settings.

B. Additional Experiments

B.1. Grounding across backbones

A key advantage of our proposed framework is its versatility; it functions as a plug-and-play module that can be seamlessly integrated into various pre-trained architectures. To validate this universality, we evaluate our method on four representative Vision-Language Models (VLMs): CLIP [35], DINOv2.TXT [20, 33], DINOv3.TXT [44], and SC-CLIP [2].

As shown in Table 10, our method yields consistent improvements when applied to different backbones (denoted as LLM-DINO and LLM-CLIP). On the PartAttrImageNet dataset, LLM-DINO achieves a state-of-the-art h-IoU of 57.8% under the Oracle-Obj protocol, surpassing the strong DINOv3.TXT baseline. Similarly, LLM-CLIP significantly boosts the generalization capability of the standard CLIP model, raising the unseen class performance from 38.9% to 44.8%. Tables 11 and 12 further detail the performance on part-only grounding. Our approach demonstrates robust performance across both PartAttrImageNet and PartAttrCUB benchmarks, consistently enhancing part segmentation accuracy (e.g., LLM-DINO achieves 69.2% Oracle-Obj mIoU on PartAttrImageNet) while maintaining high

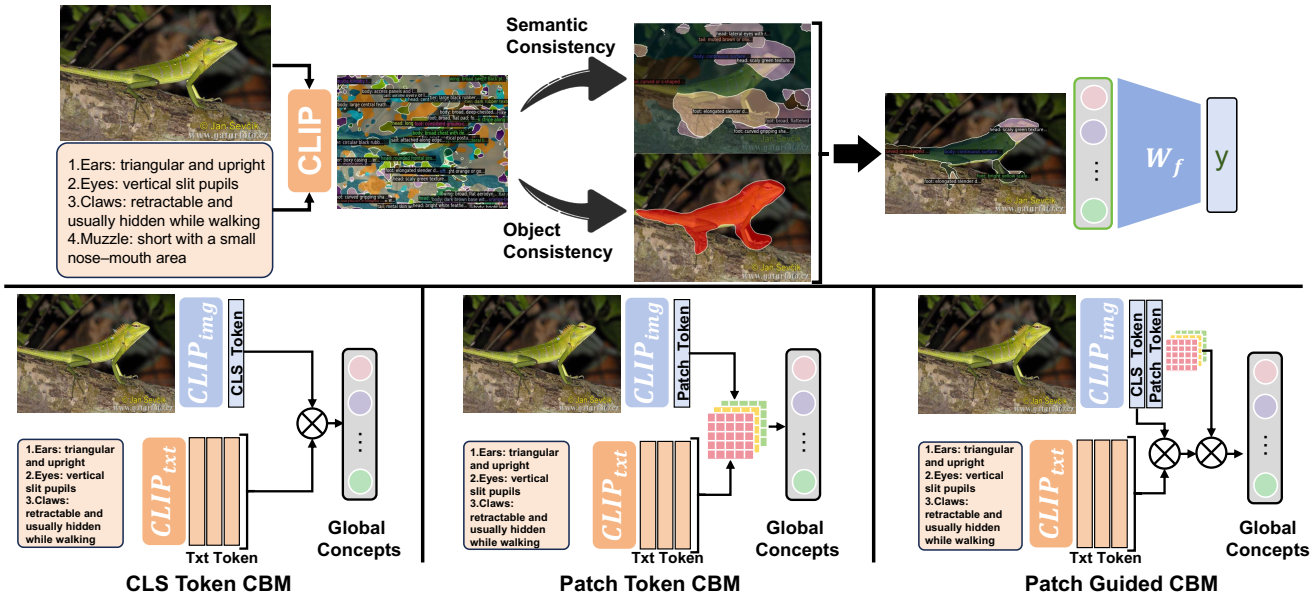

Figure 8. Overview of **OA-CBM**. *Row 1 (pipeline of our proposed methods):* We perform cost aggregation (CA) to stabilize the correspondence between visual and *class-agnostic* concept tokens, mitigating semantic inconsistency. We also perform hierarchical clustering (HC) to derive a *class-agnostic foreground object mask* by measuring similarity between visual and concept representations, thereby preserving object consistency by suppressing spurious activations from background regions. *Row 2 (three classification settings of CBM):* We present three classification paradigms: CLS Token, Patch Token, and Patch-Guided.

object-level detection rates. These results confirm that our method is backbone-agnostic and effective at refining concept grounding without compromising foundational object recognition capabilities.

Visual Granularity vs. Semantic Alignment. As observed in Table 10 and Table 11, DINO-based backbones (DINOv2.TXT, DINOv3.TXT) generally outperform CLIP-based models in part-level grounding tasks. For instance, on PartAttrImageNet (Table 11), DINOv2TXT achieves 67.6% Oracle-Obj mIoU compared to CLIP's 56.4%. This disparity stems from the pre-training objectives: CLIP focuses on global image-text alignment, often lacking the spatial granularity required for dense, pixel-level part segmentation. In contrast, DINO's self-supervised learning fosters robust local feature correspondence, making it inherently more suitable for fine-grained part grounding.

Impact on Generalization (Seen vs. Unseen). While standard backbones struggle with unseen concepts (e.g., CLIP drops from 66.4% on Seen to 38.9% on Unseen in Table 10), our method significantly narrows this gap. Specifically, LLM-CLIP improves the Unseen performance of the CLIP backbone by +5.9% (38.9% $\rightarrow$ 44.8%). This indicates that our LLM-driven module compensates for the lack of visual discriminability in CLIP by leveraging rich semantic knowledge to guide the grounding process, effectively "reasoning" about unseen concepts.

Performance on Fine-Grained Domains. On the PartAttrCUB dataset (Table 12), which requires distinguishing subtle visual traits, DINOv3TXT shows extreme overfitting (high Seen but low Unseen accuracy). However, LLM-DINO achieves a more balanced performance with a significantly higher harmonic mean (h-IoU) compared to the baselines. This suggests that combining strong visual features (DINO) with our logic-enhanced grounding (LLM) prevents the model from overfitting to specific visual patterns of seen classes, leading to more robust open-vocabulary generalization.

Backbone	Concept per class	Classification Accuracy (**Patch Token**)			
		CUB200	CIFAR100	FOOD101	FLOWER
CLIP	1	50.0	52.3	63.4	63.3
CLIP	5	65.3	82.8	82.8	83.5
CLIP	10	67.8	92.6	85.7	90.3
DINOv2TXT	1	61.6	67.3	57.0	78.3
DINOv2TXT	5	75.2	90.5	78.8	88.5
DINOv2TXT	10	76.9	94.6	82.5	96.2
SC-CLIP	1	41.4	41.0	58.3	50.0
SC-CLIP	5	54.1	72.3	60.5	65.5
SC-CLIP	10	58.0	85.9	82.1	72.1
LLM-DINO	1	22.3	37.3	34.5	21.4
LLM-DINO	5	42.9	73.1	60.5	48.0
LLM-DINO	10	48.4	83.2	66.7	55.4
LLM-CLIP	1	41.4	69.1	72.6	73.6
LLM-CLIP	5	58.2	91.7	85.4	89.5
LLM-CLIP	10	63.3	96.2	87.7	92.8

Table 13. Performance comparison of classification accuracy with different backbones.

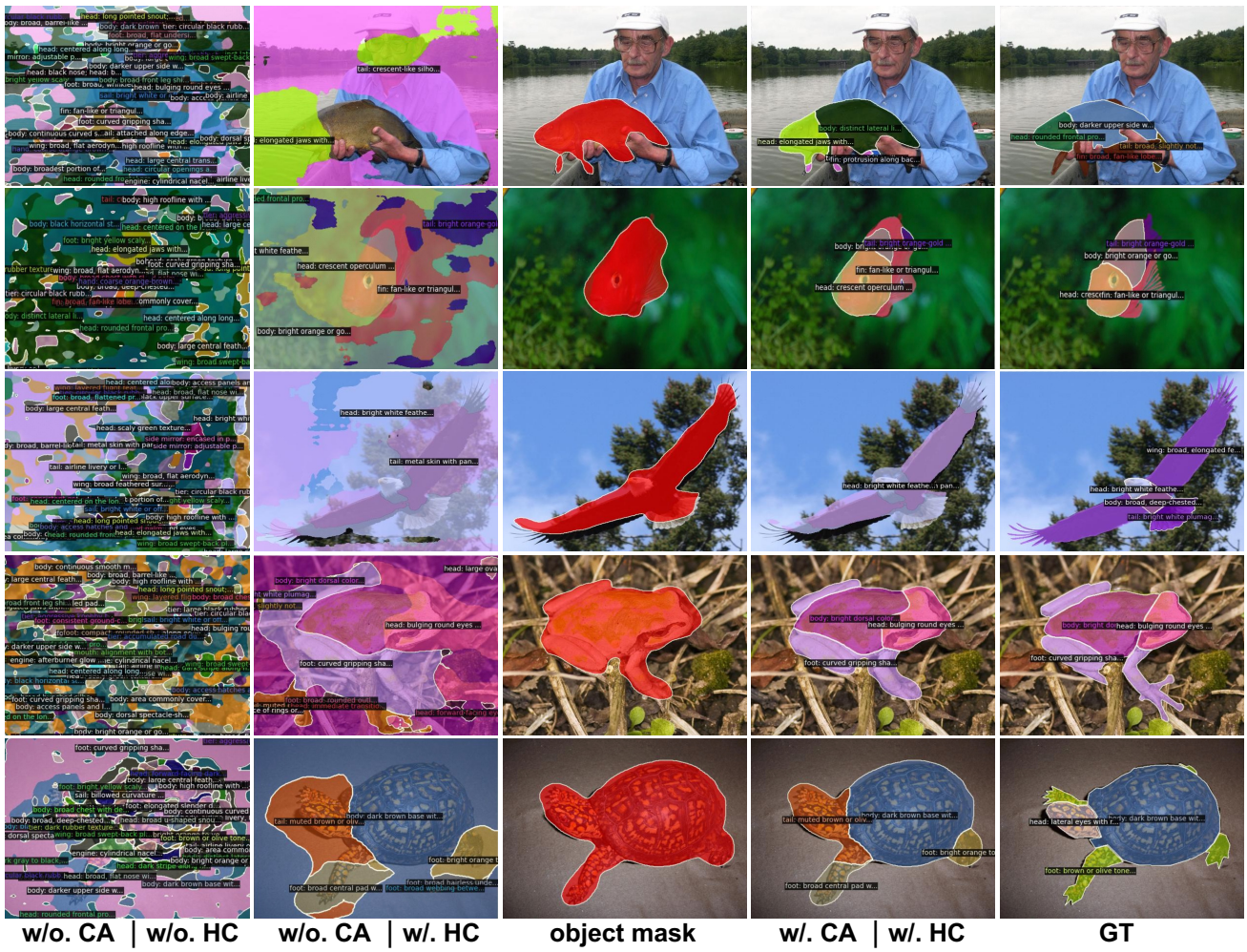

Figure 9. Visualization. The progressive improvement of OA-CBM through the Cost Aggregation (CA) and Hierarchical Clustering (HC) modules.

B.2. Classification across Backbones

We evaluate the classification performance across different backbones using the three settings outlined in Section 4.3 (visualized in Figure 8, Row 2). In this analysis, we specifically focus on the **Patch Token CBM** paradigm due to its inherent transparency and interpretability. The results are summarized in Table 13. Consistent with prior CBM literature, increasing the number of concepts per class (from 1 to 10) yields improvements in accuracy across all backbones and datasets. This confirms that a richer semantic description enhances the discriminative power of the bottleneck layer. Our **LLM-CLIP** variant demonstrates robust performance. Notably, on CIFAR100 with 10 concepts, LLM-CLIP achieves 96.2%, surpassing the standard CLIP backbone (92.6%) and DINOv2TXT (94.6%). This suggests that LLM-refined concepts align effectively with CLIP's global latent space for classification.

Following the NEC@k metric of VLG-CBM [45], which measures performance when only k concepts are activated, we count the average number of activated concepts per image in OA-CBM: approximately 20 on CUB-200, where OA-CBM achieves NEC-20 of 72.4%, and approximately 10 on CIFAR-100, where OA-CBM achieves NEC-10 of 77.8%. The higher number of activated concepts is likely due to the finer-grained part–attribute descriptions, which assign more concepts per object than prior methods.

C. Implementation Details.

Model architecture. We adopt a pre-trained ViT-L/14@336 image encoder [35] and a pre-trained Llama3 text encoder [54], both projected to a shared feature dimension of $d = 1280$. All images are resized to 336×336 during training and inference. Following CAT-Seg [8], our hierarchical clustering (HC) and cost aggregation (CA) mod-

ules are implemented in Detectron2.

Training stages. Our training consists of two stages:

- *Stage 1: Training CA and HC.* We train the CA and HC modules on our concept grounding datasets. The CA module learns concept-wise cost maps, while HC produces object masks used for object-consistent aggregation.
- *Stage 2: Training the classifier.* After Stage 1, we freeze CA and HC and using PyTorch Lightning for reproducibility. A linear classifier is then trained on downstream classification datasets using the aggregated concept activations. With CA and HC modules frozen, we extract image features once and reuse them across all experiments, significantly reducing training time. Since we only optimize a set of concept vectors and a linear layer, which is highly efficient.

Others. We use Adam [22] with $\beta_1 = 0.9$, $\beta_2 = 0.999$. For Stage 1, we use a learning rate of 2×10^{-4} with cosine decay and a batch size of 4, training for 40000 steps. For Stage 2, the linear classifier is trained for 500 epochs with a learning rate of 5×10^{-4} and batch size 512. We apply standard CLIP data augmentations (center resize–crop, color jitter) during Stage 1 and Stage 2. All experiments are conducted on a single NVIDIA H20 GPU (96GB). We fix random seeds across PyTorch, NumPy, and CUDA, and enable deterministic operations when possible.