CLASSIFY BY DESCRIPTIONS: A REALITY CHECK

Anonymous authors
Paper under double-blind review

ABSTRACT

Classify by Descriptions (CbD) has been shown to both improve zero-shot image classification and provide natural language explanations of predictions. Instead of relying on costly per-task human expert training data to achieve this, CbD infers per-class descriptors from large language models (LLMs) and then uses a vision-language model (VLM) to recognize their appearance in a query image. In this work, we critically analyze CbD and characterize its behavior when recognition becomes more fine-grained and target categories might be underrepresented in LLM and VLM pre-training data. Existing evaluation setups are unable to support this analysis with either limited coverage of realistic problem domains or a single granularity of classification. To address this gap, we introduce the classification benchmark *TaxoBench*, a benchmark spanning 7 realistic classification domains and multiple levels of classification granularity. Using *TaxoBench*, we find that CbD performance drops substantially as categories become more fine-grained and less familiar to the underlying VLM. We further identify two key sources of failure: LLM-generated descriptions often fail to capture the visual distinctions required by the task, and VLMs struggle to reliably recognize the resulting open-vocabulary concepts. We evaluate existing and new mitigation strategies but find only modest improvements, emphasizing the need for further research in more robust pre-training of foundational models and CbD as a whole.

1 INTRODUCTION

Consider an expert teaching a layperson how to recognize the endangered butternut tree. The expert might tell us that it has "a light ash-gray bark with flat-topped ridges, sticky football-shaped nuts, and leaf scars resembling a monkey's face with a fuzzy 'mustache' ". This intermediate layperson's description makes the recognition of a previously unseen tree easier. Evidence from psychology also suggests that such intermediate descriptions are central to how humans do subordinate-level or *fine-grained* categorization: a common task in many scientific domains (Biederman et al., 1999).

This idea of categorization based on high-level descriptions instead of large training datasets has been explored with *zero-shot learning* (Lampert et al., 2014; Xian et al., 2020). Early work used hand-crafted descriptions for each category and focused on training special-purpose models for recognizing these descriptions in particular domains (Lampert et al., 2009; Xian et al., 2017). However, with recent advances in computer vision and NLP, we can now use general-purpose LLMs (OpenAI, 2025) to generate descriptions for any class in any domain and use general-purpose VLMs (Radford et al., 2021) to determine if these descriptions are present in the image. If the LLM knows the category and can describe it in simple words, and if the VLM can recognize the simple descriptions in images, then we can get accurate recognition without training data. This framework is called "Classify by Descriptions" (CbD) (Menon & Vondrick, 2022; Pratt et al., 2023) (see Figure 1 (top)).

Zero-shot learning via CbD is attractive not just because it allows for recognizing new concepts without any training data. It is also *interpretable* because the intermediate descriptions are human-understandable. This allows human operators to audit the system by, e.g., editing bad intermediate descriptions or recognizing when a poorly trained VLM is unable to recognize the description. This is in contrast to more black-box approaches to zero-shot learning, where class names are directly embedded into an uninterpretable embedding space and compared to similarly uninterpretable image embeddings. The interpretability of CbD and its ability to work without training data makes it particularly attractive for critical domains where trust and/or lack of annotations is a concern.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

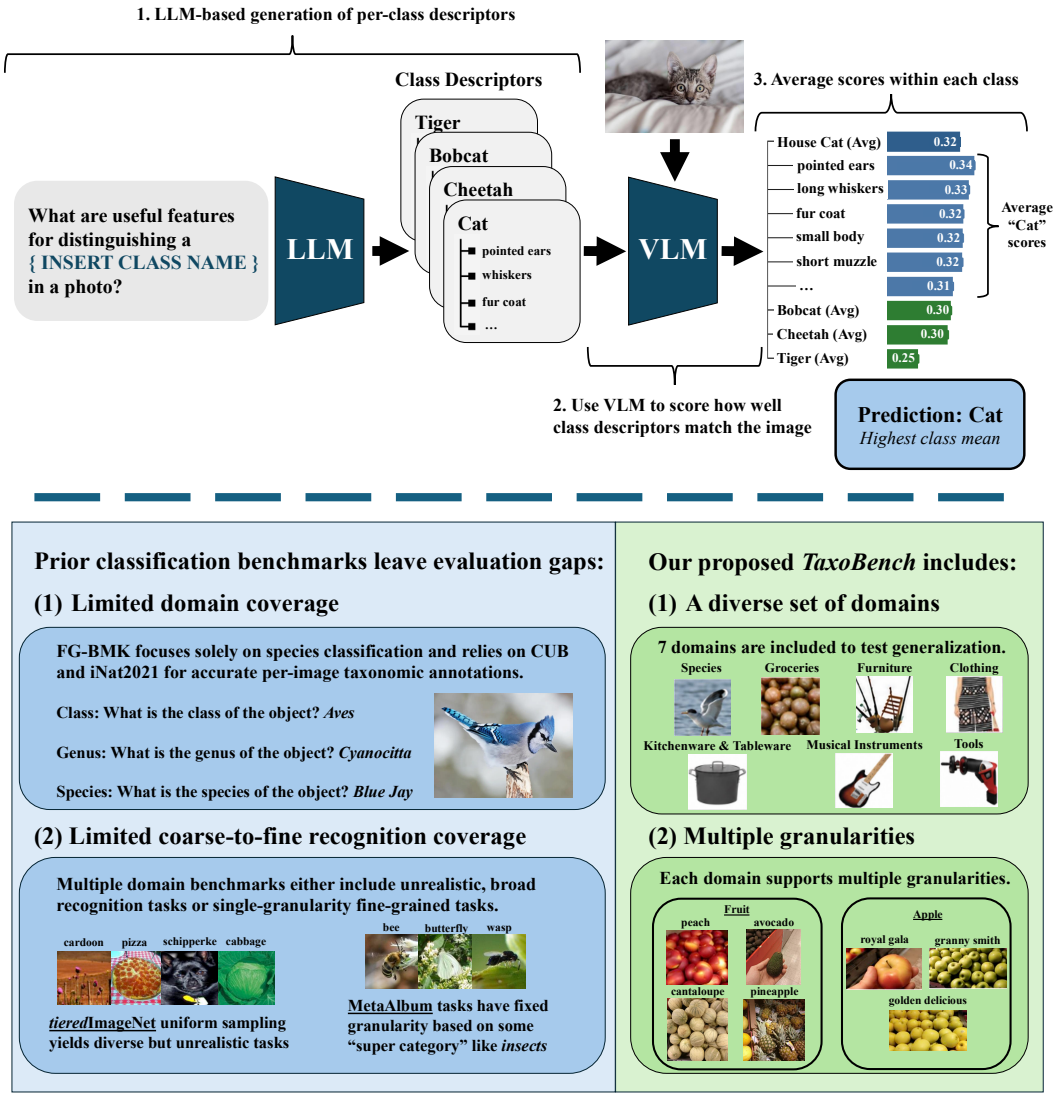

Figure 1: (Top) Standard Classify by Descriptions (CbD) pipeline; (Bottom) our benchmark includes diverse domains and degrees of fine-grain recognition.

Despite this promise, CbD does not work well in practice unfortunately. Relying on zero-shot performance of existing foundation models introduces new potential points of failure. The LLM-generated descriptors might not outline the optimal visual differences between classes, or the VLM might be unreliable on different visual concepts. Although prior work has explored some of these issues, including how descriptors are generated (Yang et al., 2023; Chiquier et al., 2024; Sehgal et al., 2025) and recognized (Roth et al., 2023; Kang et al., 2025; Saha et al., 2024), progress is unclear. The core challenge is that current benchmarks for this task are unreliable: they either feature coarse-grained categories less relevant to real-world applications of CbD (Krause et al., 2013), or narrow classification problems that existing foundation models have likely overfit to (Wah et al., 2011).

To mitigate this challenge and provide clear diagnostic evaluation, we introduce *TaxoBench*, a benchmark designed to evaluate zero- and few-shot methods for CbD and other interpretable zero-shot models on a wide range of realistic classification tasks and variable classification difficulty (see bottom of Figure 1). Unlike prior classification datasets, *TaxoBench* provides broad coverage of realistic classification domains, common and uncommon category names, and both coarse and fine-grained recognition problems. We use our benchmark to posit and answer three important questions:

How does CbD compare to black-box embedding-based zero-shot learning? We find that using interpretable descriptions, as in CbD barely outperforms simply using the embedding of the class name. Moreover, we find that CbD often "cheats" by using the class name mentioned in the description. Removing the class name craters performance by ∼50% and ∼64% across the coarse and fine-grained settings, respectively. This is surprising, as interpretable descriptions should be easier to recognize than uninterpretable class names, especially for difficult tasks such as species recognition, where performance drops by ∼7%, ∼15%, and ∼26% across the three levels of granularities.

Are the intermediate descriptions poor? The poor performance of CbD may stem from poor intermediate descriptions or poor recognition of these descriptions. Upon evaluating the former, we find the performance to be remarkably sensitive to prompting strategies. We find that including the entry-level category (Rosch et al., 1976) in the description improves accuracy by ∼22% on the most fine-grained settings. That said, even with "oracles" using self-supervision or few-shot training to produce better descriptions (Sehgal et al., 2025; Chiquier et al., 2024), accuracy still lags far behind even few-shot approaches. Thus, the interpretability of CbD appears to come at a significant cost.

Is the VLM unable to recognize the descriptions? Evaluating the alternative hypothesis that recognition is the bottleneck, we compare using CLIP in CbD with more recent CLIP-like models and multimodal LLMs (mLLMs). We find that mLLMs offer a significant improvement on the most fine-grained settings with GPT-4.1 improving over CLIP (15.46% versus 33.28%). We find that training an mLLM-style architecture on our training data provides a modest improvement compared to the standard VLM architectures without incurring the high cost of mLLM inference.

In sum, we present a new benchmark, *TaxoBench*, for evaluating interpretable zero-shot models such as CbD. Our benchmark avoids overfitting issues in past benchmarks and includes both coarse and fine-grained classification problems. Our evaluations reveal that current approaches struggle both with *producing good descriptions* and *with recognizing these descriptions*, therefore limiting classify-by-description in novel domains. We hope that our benchmark serves as an informative evaluation for the community to work on this problem.

2 PRELIMINARIES

In this section, we establish the relevant background and evaluation procedures to set up our subsequent analysis of CbD (Menon & Vondrick, 2022; Pratt et al., 2023).

Classify by Descriptions. We denote $\mathcal{T}$ as the set of classification tasks, $\mathcal{C}_t = \{c_1, \ldots, c_{K_t}\}$ as task t's candidate classes, $\mathcal{E}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$ as task t's test set where x_i is an image and $y_i \in \mathcal{C}_t$ ($|\mathcal{C}_t| = K_t$). CbD first generates descriptors for each class c with an LLM, $D(c) = \{d_{c,1}, \ldots, d_{c,M_c}\}$. A VLM ϕ then scores per-class descriptors $d \in D(c)$ on an image x as $\phi(x, d)$, which is averaged into class scores $s_{\text{desc}}(x, c) = \frac{1}{|D(c)|} \sum_{d \in D(c)} \phi(x, d)$ (see Figure 1 (top)). Model class predictions are then $\hat{c}_{\text{desc}}(x) = \arg\max_{c \in \mathcal{C}_t} s_{\text{desc}}(x, c)$.

Class name inclusion. Prior CbD methods commonly incorporate the class name into every descriptor. We denote this class-conditioned descriptor by $d_{c,j} = q(c, d_{c,j})$ where q is a prompt template. For our analysis, we assume the standard prompt $q(c, d_{c,j}) =$ "{ c }, { $d_{c,j}$ }" where the descriptor *"slender, elongated body"* becomes *"common thorntail dragonfly, slender, elongated body."* The resulting class scores are $s^*_{\text{desc}}(x, c) = \frac{1}{|D(c)|} \sum_{d \in D(c)} \phi(x, q(c, d))$.

CbD without class names tests whether the visual descriptions alone are sufficient for recognition, while *CbD with class names* exploits any class-level pre-training in the VLM (Baron et al., 2024). Importantly, neither variant of CbD uses labeled images or concept annotations from the target task.

Follow-up CbD works. Prior analyses attribute CbD performance to zero-shot recognition of class names rather than transfer across shared visual characteristics (Baron et al., 2024; Roth et al., 2023). Proposed improvements either refine the zero-shot, LLM-generated descriptors (Yang et al., 2023; Chiquier et al., 2024; Sehgal et al., 2025) or modify pre-training data or architecture for open-vocabulary descriptor recognition (Saha et al., 2024; Kang et al., 2025). We implement representative prior methods and intuitive extensions to investigate the benefits of them over standard CbD.

Evaluation metrics. We use macro accuracy, the average class-wise accuracy, $\text{MA}_t = \frac{1}{K_t} \sum_{i=1}^{K_t} \frac{TP_i}{N_i}$ where TP_i and N_i are the correctly classified and total examples of class i. Since

Table 1: **Comparison of Classification Benchmarks.** *TaxoBench* is the only dataset with zero/few-shot classification, grounded real-world tasks, and within/across-domain coarse-to-fine recognition.

Benchmark	Domains	Zero-Shot Evaluation	Few-Shot Evaluation	Real-world Tasks	Within-domain Coarse-to-Fine	Across Domains Coarse-to-Fine
Meta-Dataset	10	✗	✓	✗	✗	✗
Meta-Album	10	✗	✓	✓	✗	✗
CoOp evaluation suite	11	✓	✓	✗	✗	✗
ELEVATER	20	✓	✓	✗	✗	✗
*tiered*ImageNet	34	✗	✓	✗	✓	✓
FG-BMK	1*	✓	✗	✓	✓	✗
TaxoBench (ours)	**7**	✓	✓	✓	✓	✓

FG-BMK has 1* domain because, while it does include multiple domains, only biological species recognition is supported for hierarchical zero-shot classification.

MA_t is biased across tasks with different class counts, we also use normalized macro accuracy (Guyon et al.) K_t $\mathrm{AdjMA}_t = \frac{\mathrm{MA}_t - 1/K_t}{1 - 1/K_t}$. This allows performance comparison across tasks.

3 TaxoBench: A New Classification Benchmark

TaxoBench aims to evaluate the practicality of CbD in real-world recognition by changing the data sources and task sampling. We next describe them and compare *TaxoBench* to prior datasets.

TaxoBench's Data. *TaxoBench* draws from TreeOfLife-200M (Gu et al., 2025), DeepFashion (Liu et al., 2016), GroceryStoreDataset (Klasson et al., 2019), and ImageNet (Deng et al., 2009). These datasets were strategic choices and span a diverse set of domains: "species," "clothing," "groceries," "furniture," "kitchen & tableware," "musical instruments," and "tools." More information about each dataset and how it supplies labeled images for each of these domains is provided in Section B.

Coarse-to-Fine Task Sampling. Analysing CbD requires asking, "How does CbD handle coarse-to-fine-grained recognition within a practical domain?" and "In which domains is CbD is more effective?". Existing datasets have limited support for realistic coarse-to-fine recognition tasks within and across domains (see Figure 1, bottom).

To address this, we leveraged the above datasets. We sample the TreeOfLife-200M dataset according to its taxonomic hierarchy of species. We construct tasks at the *class*, *order*, and *family* levels to vary recognition granularity. Sampling and diversity criteria are provided in Section B.1.2. For the six human-made item domains, we use two granularities: *intermediate* and *fine*. DeepFashion's item and attribute-level annotations and GroceryStoreDataset's semantic hierarchy naturally provide these granularities. For "furniture," "kitchen & tableware," "musical instruments," and "tools", we derive them from ImageNet's 82,115 'synsets.' Details are provided in Section B.

Comparison with existing datasets. We compare *TaxoBench* against recent classification benchmarks, including Meta-Dataset (Triantafillou et al., 2020), Meta-Album (Ullah et al., 2022), CoOp 11 dataset evaluation set (Zhou et al., 2022), ELEVATER (Li et al., 2022), *tiered*ImageNet (Sun et al., 2019), and FG-BMK (Yu et al., 2026). Table 1 shows our benchmark is the only benchmark that supports within and across domains coarse-to-fine recognition on practical classification tasks.

4 Classify by Descriptions vs Black Box Approaches

We begin our analysis by comparing CbD with black-box methods that produce class predictions without a complementary human-understandable explanation. These include foundational VLMs like CLIP (Radford et al., 2021) and BioCLIP 2 (Gu et al., 2025) matching images directly to the class names, as well as nearest-centroid few-shot classifiers constructed from 1-8 image prototypes.

Results are shown in Table 2, with black-box models in the top half and CbD variants in the bottom half. Looking at the overall accuracy (last two columns), we find that generally CbD methods

Table 2: **CbD versus Black Box Methods.** Normalized macro accuracies are reported for black box zero and few-shot classification as well as CbD with and without class names prepended to class descriptors. The best results across domains, classification granularities, and inference methods are boldfaced. See Section C for experimental design details.

Method	Class names	Model	No. of shots	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
				Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
Black-box Zero-shot classification	Yes	CLIP		10.08	8.86	7.96	77.42	39.10	59.29	34.27	79.53	58.25	81.29	58.38	76.38	36.09	58.03	24.85	63.15	36.99
		OpenCLIP		6.25	6.08	5.66	**86.46**	**53.98**	**63.17**	**44.29**	**89.26**	**73.50**	**88.60**	**63.14**	**82.70**	**45.83**	**68.09**	**35.19**	**69.22**	**45.94**
		BioCLIP 2		**95.20**	**94.15**	**93.93**	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Few-shot classification	No	CLIP	1	41.66	32.45	27.01	35.84	13.64	65.57	37.89	52.33	32.17	43.55	25.38	47.25	18.00	47.42	22.07	47.66	25.17
			2	53.01	43.10	36.65	47.09	18.71	77.66	49.24	65.53	44.53	62.19	33.99	61.35	28.30	63.52	29.73	61.48	34.45
			4	63.10	52.88	46.31	62.16	27.26	84.76	58.81	72.81	54.87	72.10	44.02	73.30	37.66	73.29	38.65	71.65	43.94
			8	**69.51**	**60.60**	**54.14**	**69.71**	**34.88**	**88.55**	**67.18**	**79.30**	**61.83**	**77.69**	**51.61**	**80.72**	**45.35**	**78.40**	**45.80**	**77.70**	**51.54**
Interpretable CbD	Yes	CLIP		19.35	15.24	12.08	79.45	39.16	**65.61**	38.65	81.30	61.68	81.60	59.77	**87.23**	50.73	58.45	26.32	67.57	41.20
		OpenCLIP		17.68	13.71	10.66	**87.37**	**54.27**	64.51	**42.46**	**89.81**	**76.09**	**87.14**	**68.37**	88.89	**54.68**	**70.95**	**38.99**	**72.34**	**49.36**
		BioCLIP 2		**53.60**	**50.81**	**48.31**	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	No	CLIP		**18.05**	**12.88**	**8.95**	35.55	15.44	20.72	11.79	46.53	19.58	46.02	**23.39**	39.44	**15.99**	36.89	10.75	34.74	15.13
		OpenCLIP		15.85	12.60	8.75	**49.68**	**21.07**	**26.84**	**14.03**	**56.36**	**30.81**	**53.89**	22.56	**42.60**	14.35	**40.80**	**15.78**	**40.86**	**18.19**
		BioCLIP 2		11.05	10.05	6.46	–	–	–	–	–	–	–	–	–	–	–	–	–	–

barely outperform their black-box embedding-based zero-shot counterparts. In fact, simple few-shot learning with 8 training examples easily outperforms CbD.

More worryingly, CbD seems to rely fatally on the *name of the class* in the description. Removing the class name from the descriptions craters performance, with CbD using CLIP VLM achieving roughly *half* the accuracy of the black-box class embedding based approach (34.74% vs 67.57% on the easy setting). If CbD relies to such an extent on the class names, it implies that the classification is not actually interpretable. In other words, it is relying on recognition of the class rather than its understanding of the human-interpretable description.

Specialized domains vs. everyday concepts. One possible reason for why CbD is so reliant on class names is because the class names are *familiar*, i.e., seen during pre-training. If this is the case, we would expect that on domains that are unseen during pre-training, the natural language descriptions should provide an accuracy boost: if the model has never seen a "white walnut tree" before, a description in terms of recognizable concepts would be necessary.

Our benchmark allows us to evaluate this hypothesis by comparing species recognition (unseen domain) with everyday item recognition. Indeed, we find that on species recognition, CbD provides larger relative gains. Even with class names removed, CbD improves over standard zero-shot classification across the different degrees of taxonomic closeness.

As further corroboration of our hypothesis, we find that when we replace CLIP, which likely hasn't seen these species during pre-training, with BioCLIP, which has, once again the performance of CbD without class names craters. This confirms that *zero-shot performance relies on the overlap of new classes with pre-training data*.

Unfortunately, even when descriptions help for classes unseen during training, we find that accuracy significantly lags few-shot learning methods. Thus, CbD can help over black-box zero-shot learning, but continues to be a poor alternative when even a modicum of training data is available.

Coarse vs. fine-grained categorization. One may conjecture that descriptions are especially useful for fine-grained categorization, where the subtle visual differences may be difficult to capture from training data alone. Indeed, attributes and intermediate descriptions are central to human perception of fine-grained categories. Our benchmark allows us to evaluate whether CbD is useful in fine-grained settings relative to coarser classification problems.

Unfortunately, we find that accuracy drops for all models as one moves from coarse to fine-grained classification with CbD dropping from 72.34% to 49.36% when going from *easy* to *hard*.

CbD, in general, barely outperforms black-box zero and few-shot methods, even in settings where descriptions of the class ought to provide an advantage (fine-grained or "unseen" domains). **This points to fundamental issues with the production and recognition of zero-shot descriptions of candidate classes.**

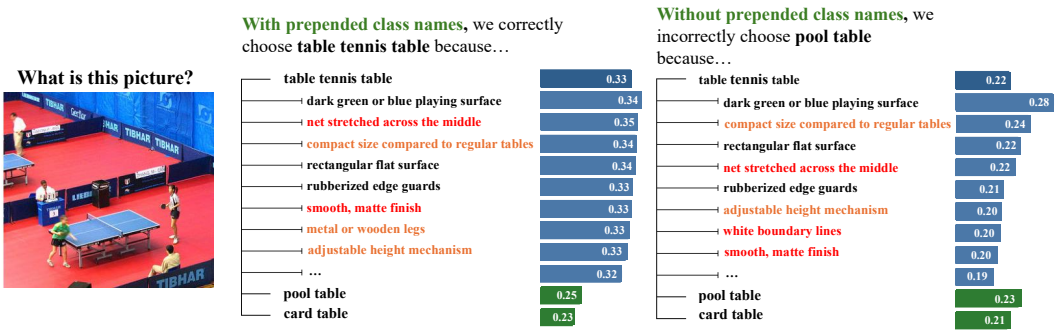

Figure 2: CbD provides explanations for its decision, yet inaccurate, spurious, or non-visual descriptors; unreliable recognition; and prepending class names onto descriptors limit their practicality.

Table 3: **Search Algorithms for CbD.** Normalized macro accuracies are reported for both self-supervised and few-shot descriptor evolutionary search. See Section D.1 for details.

Method	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
Zero-shot																	
CbD	12.53	9.53	9.65	21.49	14.09	19.07	11.72	68.35	22.31	39.29	21.27	37.55	18.62	30.22	10.07	32.64	15.39
Self-sup.																	
ESCHER	11.46	9.94	6.70	45.37	12.35	12.32	5.50	52.62	26.31	54.12	13.82	25.03	18.04	13.04	8.71	30.57	13.06
Few-shot																	
LLM-Mutate	**21.73**	**14.56**	**14.28**	**54.50**	**22.79**	**32.30**	**32.19**	**71.65**	**34.25**	**48.94**	**25.61**	**39.74**	**29.84**	**38.52**	**22.64**	**43.91**	**25.94**

In the subsequent sections, we focus on *CbD without the class name shortcut* and why it fails.

5 The Language Bottleneck

CbD has two major components (see Figure 2) and the problem could be in either or both of these components: 1) the LLM descriptor generation and 2) the open-vocabulary recognition. In this section, we focus on the former and ask: "Can zero-shot descriptors be improved, and exactly how much visual information can help?" Then in section 6, we look at the latter by exploring limitations of open-vocabulary recognition. To ensure fair comparison of the language bottleneck design, we use the CLIP model to score descriptors in this section. Thus, differences in performance can be attributed to how the concept descriptions are being generated.

5.1 Search for the best descriptors

One might argue that the low performance of CbD is the descriptions are zero-shot and have not been optimized or learnt on a training set. To investigate whether this is true, we relax the zero-shot assumption and provide CbD with a training set of images. We consider two possibilities: one, using an *unlabeled* training set and improving the descriptors via a self-supervised evolutionary search (ESCHER) (Sehgal et al., 2025)) and two, using *a few labeled examples* to hill-climb on accuracy (LLM-Mutate) (Chiquier et al., 2024) .

Counterintuitively, we find more often than not ESCHER's self-supervised optimization of the language descriptions *reduces* accuracy. In fact, ESCHER only improved performance on the *intermediate* "clothing" and "kitchenware & tableware" tasks and the *fine* "furniture" task. ESCHER relies on continually adding descriptors to classes with the most correlated predictions on a static unlabeled dataset. Unfortunately, without any feedback mechanism, the descriptor list for each class just becomes progressively longer and noisier.

We obtain significantly better results with the supervised LLM-Mutate approach. However, accuracies still lag behind few-shot approaches using the same training data (25.94% vs 51.54% on the *hard* split). This suggests that even with optimization, the best descriptions we find are insufficient.

Table 4: **Prompting Strategies for CbD.** Normalized macro accuracies are reported for different prompting strategies. See Section D.2 for details.

Method	No. of shots	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
		Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
CbD		17.70	12.96	8.80	32.69	15.48	20.67	11.30	47.79	19.67	42.50	22.70	36.27	15.75	33.54	11.16	31.97	14.79
full sentences		21.41	13.68	10.81	**65.26**	11.70	**39.47**	**17.44**	**76.97**	**48.02**	**73.08**	35.20	**76.14**	**22.75**	36.08	**17.44**	**55.72**	**23.72**
few-shot in context	4	25.45	19.53	15.79	45.82	16.00	29.20	16.08	59.71	36.11	67.15	32.80	57.27	15.27	44.62	16.02	46.02	21.13
	8	**28.36**	**20.26**	**16.05**	46.76	**16.63**	32.08	17.41	60.23	36.86	66.68	**36.06**	56.80	16.52	48.54	16.34	47.33	22.14

5.2 Prompting Strategies

The aforementioned search algorithms are one way to improve upon the standard zero-shot class descriptors. Another, less computationally expensive, option is to explore different prompting strategies to elicit better zero-shot production of descriptors. CbD, ESCHER and LLM-Mutate all rely on a manually designed prompt, so descriptions may be sensitive to prompt wording. Traditionally, CbD just asks for the visual features that might distinguish the provided class in a photo ("What are useful features for distinguishing a { CLASS } in a photo?"; see Figure 1). We conjectured that perhaps these descriptions are not detailed or expressive enough, and replaced the prompt with "What are useful full sentence features for distinguishing a { CLASS } in a photo?". Another possibility is to use labeled few-shot examples as part of the prompt instead of for optimization.

Table 4 evaluates these prompting strategies. Surprisingly, we see a significant boost from both prompting strategies. Using few-shot examples in context in the prompt outperforms LLM-Mutate, suggesting in part that the evolutionary search in LLM-Mutate may not return the optimal solution. Full-sentence descriptions provide an even larger overall gain, but these come mainly from everyday item classification. Moreover, full sentence descriptions come at the cost of lower interpretability: the descriptions are sometimes vague or conflate multiple visual features (e.g., "It typically includes armrests and a high backrest for support and protection."). Nevertheless, these prompting strategies provide a computationally inexpensive and simple yet effective way of improving CbD.

5.3 Entry Level Categorization

Descriptors prepended with class names hurt interpretability but yield the best performance (Baron et al., 2024). Can we retain the benefits of class names without sacrificing interpretability? We look to cognitive science for inspiration, where prior work found that humans tend to first categorize objects at an intermediate, entry-level. When seeing a golden retriever, people tend to first categorize it as a dog instead of a superordinate category like "animal" or a more fine-grained one like "golden retriever". Further entry-level categories provide context for interpreting fine-grained visual descriptors. For example, dark spots on a banana can indicate ripeness, whereas dark spots on a leopard are part of its natural coat pattern.

To leverage this insight, we prepend descriptors with an entry-level categorization common to all classes being classified (e.g., prepending "a bird" to generated descriptors when classifying between bird species). WaffleCLIP (Roth et al., 2023) similarly discussed entry-level categorization, but combined these categories with class names. Our analysis instead asks whether coarse categories can replace class names to improve performance while preserving interpretability.

In Table 5, we see that simply providing this coarse categorization improves zero-shot performance over using only zero-shot generated descriptors (38.95% versus 31.97% and 18.09% versus 14.79% on the *easy* and *hard* tasks respectively). These results mimic many of the trends seen in Table 4, suggesting that some performance gains in the last section might come from entry-level category information leaked into the longer descriptor sentences.

The language bottleneck in CbD can improve significantly either with more verbose descriptors or few-shot examples. The former reduces interpretability. The latter relies on relaxing the zero-shot constraint. **Entry-level categorization provides a happy medium that preserves interpretability without relying on labeled examples while improving performance.**

Table 5: **Entry-level Categorization.** Normalized macro accuracies are reported to see the improvement associated with entry-level categorization. See Section D.3 for details.

Method	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
Nothing	17.70	12.96	8.80	25.33	14.14	20.67	11.30	47.79	19.67	42.50	**22.70**	36.27	15.75	33.54	**11.16**	31.97	14.79
Entry-level Categories	**21.50**	**14.95**	**10.15**	**46.57**	**19.70**	**25.57**	**14.63**	**53.31**	**29.47**	**51.86**	21.42	**38.35**	**21.05**	**36.54**	10.56	**38.95**	**18.09**

Table 6: **Attribute-Focused VLMs.** Normalized macro accuracies comparing CLIP, DCSM. BLIP-2, EVA02-CLIP, and SigLIP-2 for no class name CbD. See Section E.1 for details.

Model	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
CLIP	17.70	12.96	8.80	25.33	14.14	20.67	11.30	47.79	19.67	42.50	22.70	36.27	15.75	**33.54**	11.16	31.97	14.79
DCSM	10.98	6.50	4.08	23.63	9.42	12.09	4.63	37.09	11.26	17.37	12.18	35.20	10.15	26.95	7.09	24.27	8.37
BLIP-2	15.74	12.61	9.09	46.77	13.44	15.46	**19.09**	58.84	33.12	47.06	28.08	32.75	13.72	26.42	10.62	35.30	18.48
EVA02-CLIP	18.04	**14.51**	9.38	**56.49**	**23.03**	20.97	13.67	**60.31**	**33.52**	**51.39**	26.28	**40.94**	**22.90**	29.97	**17.55**	**39.73**	**20.90**
SigLIP-2	**18.88**	14.32	**9.74**	50.34	21.86	**25.68**	12.05	57.94	29.98	50.26	**28.11**	34.98	12.55	32.39	16.97	38.64	18.75

6 Open-Vocabulary Descriptor Recognition

Recall in Figure 2, how both descriptors and recognition interplay in CbD. In particular, downstream performance relies on accurate descriptors and a capable recognition model. This raises the concern whether existing VLMs can, in fact, recognize open-vocabulary descriptors. Prior works (Yuksekgonul et al., 2023; Roth et al., 2023; Kang et al., 2025) have explored the limitations of CLIP-style models. Here, we explore what can be done to better recognize these descriptors without needing to augment them with entry-level category or class references. To isolate the impact of different VLM models, we keep standard CbD prompting for descriptors and do not prepend class names.

6.1 Attribute-focused CLIP models

The first foundational VLMs, such as CLIP, trained on image-caption pre-training data to align global image and text descriptors. However, this might not be optimal for our attribute-level categorization problem. In this section, we explore the performance of recent architectures and pre-training models for better open-vocabulary recognition. In particular, we explore DCSM (Kang et al., 2025), BLIP-2 (Li et al., 2023), EVA02-CLIP (Fang et al., 2024), and SigLIP-2 (Tschannen et al., 2025) to see how different pre-training strategies and architectural choices affect downstream performance.

We see in Table 6 that the various VLMs differ in accuracy. With the exception of DCSM, all other variants improved over the base CLIP model. Nevertheless, the performance gains were modest, with the largest improvement occurring between CLIP and EVA02-CLIP (31.97% to 39.73% and 14.79% to 20.90% on the *easy* and *hard* tasks).

6.2 Multimodal Large Language models

An alternative to using similarity between image and text embeddings is to use a multimodal large language model (mLLM), which combines a vision encoder with a language model capable of following instructions and generating free-form answers. Unlike CLIP-style models, mLLMs need not compress the entire image and descriptor into a single similarity score. They can attend to the image conditioned on a question, decompose a descriptor into its constituent properties, and express uncertainty when the relevant visual evidence is absent or ambiguous.

This raises a natural question: do mLLMs eliminate the need for improved descriptor recognition altogether? We investigate this question by replacing the typical CLIP-based VLM with a multimodal LLM, such as Qwen-3-VL and GPT-4.1 in the CbD inference scheme. With this change, we found substantial accuracy improvements over standard CLIP ((31.97% to 52.57% and 14.79% to 33.28% on the *easy* and *hard* tasks)). This suggests that CbD and its variants should likely switch to multimodal language models as the underlying recognition engine.

Table 7: **Multimodal LLMs.** Normalized macro accuracies for CLIP, an open-source and a proprietary mLLM with the no class name CbD inference scheme. See Section E.2 for details.

Model	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools		Overall	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Easy	Hard
CLIP	19.88	13.09	9.04	21.49	12.19	19.07	11.05	**73.29**	21.43	39.39	21.27	18.90	18.62	32.62	10.72	32.09	14.90
Qwen-3-VL-32B	29.64	23.05	16.11	**56.47**	**38.81**	**40.12**	24.13	54.07	29.58	74.84	44.13	51.03	37.69	38.52	19.78	49.24	30.03
GPT-4.1	**33.93**	**24.81**	**18.45**	29.72	21.39	9.95	**25.37**	60.71	**31.21**	**81.18**	**48.69**	**79.06**	**51.61**	**73.44**	**36.25**	**52.57**	**33.28**

Table 8: **Dedicated Data and Model for CbD.** Normalized macro accuracy at the order level. See Section E.3 for details.

	CLIP	AdaptCLIPZS	AdaptCLIPZS^{FT}	DCSM	DCSM^{FT}	Ours
Order accuracy	12.96	13.60	13.88	6.88	9.60	**15.95**

However, many of these tasks, especially the *hard* settings, have substantial room for performance gains when compared with even CLIP-based zero-shot image classification (63.15% and 36.99% on *easy* and *hard* setting). Thus, *TaxoBench* stands as not only a benchmark for CbD but also the progress of foundational mLLMs for open-vocabulary recognition. Moreover, we found that smaller, open-source mLLM performance is highly dependent on the prompting used (see Section E.2).

6.3 Dedicated model architecture and training

Motivated by the promising mLLM results and lackluster attribute-centric VLM results in the preceding section, we explored how to replicate mLLM performance without the associated expensive forward passes and prompt sensitivity. We leveraged this with a VLM architecture that cross-attends across image patches to produce image–text similarity scores (see Section E.3). We compare our trained VLM against prior works, AdaptCLIPZS (Saha et al., 2024) and DCSM (Kang et al., 2025) on the long-tail species domain to disentangle the confounding factors of task-specific training data and the model architecture. Table 8 shows that our method outperforms prior works. Nevertheless, performance gains are modest, necessitating continued research into better embedding-style VLMs.

The recognition bottleneck in CbD is only reliably improved with mLLMs with some more recent CLIP-style VLMs showing modest improvements. Nevertheless, reliance on mLLMs means significant compute costs and sensitivity to both prompting and inference schemes. **The realized performance gains are modest and highlight the need for continued research in the space of efficient, open-vocabulary recognition.**

7 Conclusion

In this work, we investigate the limitations of CbD. Existing classification benchmarks either lack coverage or controlled fine-grained granularity between classes. To remedy this, we present *TaxoBench*, the first benchmark that involves several problem spaces and controls for how fine-grained candidate classes are. Through systematic experimentation, we demonstrate three fundamental issues with the current state of CbD: over-reliance on class names, inaccurate LLM-generation of per-class descriptions, and unreliable open-vocabulary recognition. We explore existing methodologies to both improve the descriptions and recognition of them. In our analysis, prompting strategies as well as conditioning on entry-level categories had a positive impact on performance. Meanwhile, recognition improved considerably when moving from standard CLIP-style VLMs to patch-wise mLLMs. Overall, our results on *TaxoBench* indicate a clear gap when moving from black box zero- or few-shot methods to CbD, highlighting a direction for future work.

REFERENCES

OpenAI (2023). Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ethan Baron, Idan Tankel, Peter Tu, and Guy Ben-Yosef. Real classification by description: Extending clip's limits of part attributes recognition. *CoRR*, 2024.

Irving Biederman, Suresh Subramaniam, Moshe Bar, Peter Kalocsai, and József Fiser. Subordinate-level object classification reexamined. *Psychological Research*, 62(2-3):131–153, July 1999. ISSN 0340-0727. doi: 10.1007/s004260050047.

Elizabeth G Campolongo, Jianyang Gu, and Net Zhang. IDLE-OO Camera Traps, 2025. URL https://huggingface.co/datasets/imageomics/IDLE-OO-Camera-Traps.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2818–2829, 2023.

Mia Chiquier, Utkarsh Mall, and Carl Vondrick. Evolving interpretable visual classifiers with large language models. In *European Conference on Computer Vision*, pp. 183–201. Springer, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Encyclopedia of Life (EOL). URL https://eol.org. Accessed March 2025.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024. ISSN 0262-8856. doi: 10.1016/j.imavis.2024.105171. URL http://dx.doi.org/10.1016/j.imavis.2024.105171.

GBIF Secretariat. GBIF Backbone Taxonomy. 2023. doi: 10.15468/39omei. URL https://doi.org/10.15468/39omei.

GBIF.org. GBIF occurrence download. May 2024. doi: 10.15468/DL.BFV433. URL https://doi.org/10.15468/dl.bfv433.

Zahra Gharaee, Scott C Lowe, ZeMing Gong, Pablo M Arias, Nicholas Pellegrino, Austin T Wang, Joakim B Haurum, Iuliia Zarubiieva, Lila Kari, Dirk Steinke, et al. Bioscan-5m: A multimodal dataset for insect biodiversity. *Advances in Neural Information Processing Systems*, 37:36285–36313, 2024.

Jianyang Gu, Samuel Stevens, Elizabeth G Campolongo, Matthew J Thompson, Net Zhang, Jiaman Wu, Andrei Kopanev, Zheda Mai, Alexander E White, James Balhoff, et al. Bioclip 2: Emergent properties from scaling hierarchical contrastive learning. *arXiv preprint arXiv:2505.23883*, 2025.

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. Analysis of the automl challenge series 2015-2018. pp. 191–236.

iNaturalist contributors and iNaturalist. iNaturalist Research-grade Observations. Accessed via GBIF.org (Occurrence Snapshot: 10.15468/DL.BFV433) 2024-05-01.

IUCN. The iucn red list of threatened species. https://www.iucnredlist.org, 2025. Version 2025-2. Accessed: 12 March 2026.

Raphi Kang, Yue Song, Georgia Gkioxari, and Pietro Perona. Is clip ideal? no. can we fix it? yes! *ICCV*, 2025.

Kakani Katija, Eric Orenstein, Brian Schlining, Lonny Lundsten, Kevin Barnard, Giovanna Sainz, Oceane Boulais, Megan Cromwell, Erin Butler, Benjamin Woodward, and Katherine L. C. Bell. FathomNet: A global image database for enabling artificial intelligence in the ocean. *Scientific Reports*, 12(1):15914, September 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-19939-2. URL https://www.nature.com/articles/s41598-022-19939-2.

Marcus Klasson, Cheng Zhang, and Hedvig Kjellström. A hierarchical grocery store image dataset with visual and semantic labels, 2019. URL https://arxiv.org/abs/1901.00711.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pp. 5338–5348. PMLR, 2020.

Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–958, 2009. doi: 10.1109/CVPR.2009.5206594.

Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014. doi: 10.1109/TPAMI.2013.140.

Chunyuan Li, Haotian Liu, Liunian Harold Li, Pengchuan Zhang, Jyoti Aneja, Jianwei Yang, Ping Jin, Houdong Hu, Zicheng Liu, Yong Jae Lee, and Jianfeng Gao. Elevater: A benchmark and toolkit for evaluating language-augmented visual models, 2022. URL https://arxiv.org/abs/2204.08790.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. URL https://arxiv.org/abs/2301.12597.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sachit Menon and Carl Vondrick. Visual classification via description from large language models. *ICLR*, 2022.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

OpenAI. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 2025. Accessed: 2026-03-05.

Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15691–15701, 2023.

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*, pp. 8748–8763. PmLR, 2021.

Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8(3):382–439, 1976. ISSN 0010-0285. doi: https://doi.org/10.1016/0010-0285(76)90013-X. URL https://www.sciencedirect.com/science/article/pii/001002857690013X.

Karsten Roth, Jae Myung Kim, Andrew Koepke, Oriol Vinyals, Cordelia Schmid, and Zeynep Akata. Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15746–15757, 2023.

Oindrila Saha, Grant Van Horn, and Subhransu Maji. Improved zero-shot classification by adapting vlms with text descriptions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17542–17552, 2024.

Atharva Sehgal, Patrick Yuan, Ziniu Hu, Yisong Yue, Jennifer J Sun, and Swarat Chaudhuri. Self-evolving visual concept library using vision-language critics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13124–13134, 2025.

Samuel Stevens, Jiaman Wu, Matthew J Thompson, Elizabeth G Campolongo, Chan Hee Song, David Edward Carlyn, Li Dong, Wasila M Dahdul, Charles Stewart, Tanya Berger-Wolf, et al. Bioclip: A vision foundation model for the tree of life. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19412–19424, 2024.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning, 2019. URL https://arxiv.org/abs/1812.02391.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples, 2020. URL https://arxiv.org/abs/1903.03096.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL https://arxiv.org/abs/2502.14786.

Ihsan Ullah, Dustin Carrión-Ojeda, Sergio Escalera, Isabelle Guyon, Mike Huisman, Felix Mohr, Jan N Van Rijn, Haozhe Sun, Joaquin Vanschoren, and Phan Anh Vu. Meta-album: Multi-domain meta-dataset for few-shot image classification. *Advances in Neural Information Processing Systems*, 35:3232–3247, 2022.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, pp. 595–604, 2015.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4582–4591, 2017.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly, 2020. URL https://arxiv.org/abs/1707.00600.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *CVPR*, 2023.

Hong-Tao Yu, Chen-Wei Xie, Yuxin Peng, Serge Belongie, and Xiu-Shen Wei. Benchmarking large vision-language models on fine-grained image tasks: From evaluation to diagnosis, 2026. URL https://arxiv.org/abs/2606.19053.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KRLUvxh8uaX.

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. ISSN 1573-1405. doi: 10.1007/s11263-022-01653-1. URL http://dx.doi.org/10.1007/s11263-022-01653-1.

A Table of Contents

The appendix is organized as follows:

- In §B, we discuss our data curation process and how we use it to produce a first of its kind zero-shot classification benchmark *TaxoBench*.
- In §C, we discuss how we set up the different experiments used to compare CbD with black box methods.
- In §D, we discuss how we set up experiments to determine how the language bottleneck of CbD can be alleviated.
- In §E, we discuss how we set up the different experiments used to perform a comprehensive analysis of possible choices for the VLM and how that impacts CbD performance.
- In §F, we discuss additional ablations we performed to better understand CbD.

B *TaxoBench* Details

B.1 Species Domain

B.1.1 Data Collection

To produce a comprehensive dataset for species classification, we use species observations from the largest publicly available dataset, i.e., the TreeOfLife-200M dataset (Gu et al., 2025). Containing over 214 million images and 952 thousand out of the over 2 million cataloged species (IUCN, 2025), the observations are sourced from a wide range of settings, including microscope images of insects (from BIOSCAN-5M (Gharaee et al., 2024)), museums (from GBIF (GBIF Secretariat, 2023; GBIF.org, 2024) and EOL (Encyclopedia of Life (EOL))), camera traps (from GBIF and EOL), and crowd-sourced species observations (from GBIF, iNaturalist (iNaturalist contributors & iNaturalist), EOL, and FathomNet Database (Katija et al., 2022)).

Unlike the TreeOfLife-200M dataset (Gu et al., 2025), which includes all these sources to produce as comprehensive a dataset as possible to train a foundational species classification model, our dataset needs to focus more on representative, in-the-wild images of species, such as camera traps and crowd-sourced species observations. Specifically, we want to ground Classify by Descriptions (CbD) performance in a real-world task and avoid impractical or unrepresentative images in lab or museum settings.

We first filtered out all of the images and species from BIOSCAN-5M to avoid images pictured in lab settings. For the remaining species, we then collected and stored all remaining observations with non-empty image URLs and data source entries. To ensure fidelity with respect to data sources, we pinged stored image URLs and recorded how many observations could be downloaded, yielding over 7 million unique images and over 800 thousand unique species. This data, however, still includes suboptimal observations: museum images and sound plots iNaturalist images. Examples of the different types of filtered images can be see in Figure 5.

To remedy this, we perform image search over the remaining observations with the following selection criteria:

1. The image's download URL and stored data source or type must not reference "museum."



Figure 3: Example of species from Buteo genus to illustrate how visually similar species sharing a genus can be.

2. The image's download URL does not include "sounds."

Our dataset then selects species over the resulting observations with at least 20 unique images to get the final 71,871 species encompassing *TaxoBench*. We made this selection criteria to focus on species with enough examples to learn robust features over and evaluate on.

B.1.2 Benchmark Construction

To perform zero-shot, fine-grained species classification, we use observations from 71,871 species collected in the previous section and organize them into evaluation datasets disjoint from a representative training set.

We must first determine an algorithm to select candidate evaluate species. Unlike the original dataset's collection criteria, we expect more images per species to perform classification (i.e., 25 images per species), resulting in 29,620 candidate species. A natural next question is how do we want to organize our benchmark and prior work. Taking inspiration from the natural taxonomic hierarchy of species to relate species' "close-ness," we decide to organize across different taxonomic levels (e.g., kingdom, phyla, class, etc.). We decide to ignore kingdom and phyla taxonomic levels and genus taxonomic levels for being too coarse or too fine-grained respectively. For example, kingdom is at the level of animals and plants which could compare insects and mammals, a trivial classification to do without sophisticated recognition. Phyla-level classification similarly is too coarse, comparing fish and insects under the same phyla *Chordata*. On the other hand, genus requires comparing too-similarly looking species as seen in Figure 3. This means our benchmark will support class, order, and family-level zero-shot classification.

To perform evaluation, we need to select classes, orders, and families and then sample within them in a diversity-first approach. This two-part procedure requires a separate search algorithm to select taxonomic groups and then another to select within groups. The former problem of group selection can be formalized into an algorithm that takes in candidate species taxonomic table $\mathcal{D}$ and a set of user-specified constraints, including the target taxonomic level to select groups over $\ell \in \{\texttt{class}, \texttt{order}, \texttt{family}\}$, number of groups to select N, maximum quotas $Q^{\max}$ for phyla $Q_p^{\max}$, class $Q_c^{\max}$, and order $Q_o^{\max}$, minimum species per group m_{species}, and spread k across the commonly observed biological kingdoms (animalia/animals, plantae/plants, and fungi), and produces a diversity-focused set of N taxonomic groups at level ℓ. To implement this, we use a diversity-constrained sampling procedure summarized in Algorithm 1. The algorithm first constructs candidate groups at the target taxonomic level ℓ, then computes quotas that limit the representation of higher-level taxa based on specified diversity caps $Q^{\max}$, and finally performs a round-robin selection that respects these diversity constraints until the desired N taxonomic groups are selected.

Due to the long-tailed distribution of observations, we found at the most coarse classification setting, class, only 50 unique classes remain after requiring classes to have at least 10 species with 25 unique observations in. Fortunately, this set of classes includes 27, 12, and 11 classes across the

Algorithm 1 Diversity-Constrained Taxonomic Group Selection

Require: Species table $\mathcal{D}$, target taxonomic level ℓ, target number of groups N, minimum species per group m_{species}, diversity caps $Q^{\max}$, kingdom distribution k
Ensure: Selected groups $\mathcal{U}^\star$ from taxonomic level ℓ

```
Initialize candidate groups G at level ℓ from D and m_species (Section ??)
Compute per-kingdom quotas N_k from k and |G|
U* ← ∅
for all kingdoms r do
    G_r ← candidate groups belonging to kingdom r
    Compute quotas Q for taxa at the level above ℓ from Q^max and G_r (Section ??)
    Partition G_r based on parent taxa (one level above ℓ)
    Shuffle each partition
    Let C be a counter for the different taxon levels capped by Q
    while quota N_k is not satisfied and a partition still satisfies Q do
        Visit partitions in round-robin order
        Let g be the next candidate group
        if adding g respects the taxonomic quotas Q based on C then
            Remove g from partition and add to U*
            Update C using taxonomic information of g
        end if
    end while
end for
if |U*| < N then
    Randomly consider remaining candidate groups
    Add any group that does not violate Q^max
end if
return U*
```

animal, plant, and fungus kingdoms respectively. This represents a diverse of a collection of species ranging across the commonly observed biological kingdoms. Thus, we decided to keep all of these classes in our class-level benchmark and align order and family selection to match this benchmark for easily comparable performance in ablation studies. Specifically, at the finer-grained order and family taxonomic levels, we define the kingdom densities k to match the class-level benchmark. We then leverage our search algorithm's ability to round-robin sample with diversity constraints to prevent any one phyla, class, or order dominating performance. We decide to set $Q^{\max}$ such that any single phyla, class, or order can not contain more than 25%, 15% class, and 10% of the total selected groups respectively. Table 9 shows the distribution of our order and family-level benchmarks across taxonomic levels, emphasizing diversity. We note despite setting the same kingdom densities across the three benchmarks the realized distributions vary.

Having selected the taxonomic groups (classes, orders, or families), we need a general algorithm to search for the maximally diverse set of species available. We introduce Algorithm 2 to sample n diverse species from a species table $\mathcal{D}$ containing candidate species from a selected taxonomic group. We also note that, for each selected species, we randomly sample 25 filtered images to make accuracies comparable across rare and common species.

After constructing class, order, and family-level classification benchmarks, we filter the original 71,871 species to not include species from these benchmarks or any of the common zero-shot classification benchmarks CUB (Wah et al., 2011), NaBirds (Van Horn et al., 2015), Flowers102 (Nilsback & Zisserman, 2008), Camera Tap datasets (Campolongo et al., 2025), Meta-Album (Ullah et al., 2022), and Rare Species (Stevens et al., 2024). This results in the 67,595 species training dataset used to fine-tune vision language models.

Candidate group construction. We first identify candidate taxonomic groups at the target level ℓ to search over. Species are grouped by their taxonomic parent at level ℓ, and any group containing fewer than m_{species} species is discarded. For each remaining group, we record its ancestor taxa (e.g., kingdom, phylum, class, order) so that diversity constraints can later be enforced.

Algorithm 2 Diversity-First Species Selection

Require: species table D for a group at taxonomic level ℓ, sample size n
Ensure: Sampled n-length species list S

```
Let L⁻(ℓ) be set of taxonomic levels more fine-grained than or below ℓ
Build a tree T from iterating a shuffled copy of D (Section B.1.2)
S ← [ ]
while |S| < n do
    Remove empty nodes from T
    if T is empty then
        break
    end if
    Visit children of root node in randomized round-robin manner
    For each visited node, randomly traverse until a leaf node is reached
    Remove species from T and add the resulting species to S if not already in S
    if no species were added then
        Randomly add remaining species from D until |S| = n
        break
    end if
end while
return S
```

Quota allocation. To prevent over-representation of any particular higher-level taxa, we compute quotas for taxa at the level immediately above ℓ. Specified quotas Q are allocated proportionally to the number of candidate groups within each taxon and rounded to integers using largest-remainder rounding. These quotas are then capped by the maximum share allowed for each taxonomic level. If resulting quotas at level above ℓ do not add to target number of species, we increase quotas in taxa where number of candidate groups from G with that taxa is more than the current quota and the current quota is not equal to the taxonomic cap from $Q^{\max}$.

Tree Construction. We construct a hierarchical tree over species using their taxonomic annotations. The leaf nodes correspond to species, while internal nodes represent higher-level taxa (e.g., genus, family, order, and class).

The provided species table $\mathcal{D}$ is first randomly shuffled to avoid ordering bias. We then iterate through the species and insert each species into the tree according to its taxonomic lineage. If a node for a particular taxon does not yet exist, it is created. This results in a tree whose structure mirrors the biological taxonomy.

B.1.3 Comparison to Existing Datasets

We compare our benchmarks against common species classification datasets including CUB (Wah et al., 2011), NaBirds (Van Horn et al., 2015), Flowers102 (Nilsback & Zisserman, 2008), Meta-Album (Ullah et al., 2022), Camera Trap (Campolongo et al., 2025), and Rare Species (Stevens et al., 2024) datasets. Table 9 shows our benchmark is larger and more varied across the different taxonomic levels. We note our family-level benchmark has similar number of total families as the existing Rare Species dataset, yet covers significantly more unique kingdoms, phylum, classes, and orders. This emphasizes the efficacy of our diversity-first search algorithm proposed in the previous section. Examples of class-level, order-level, and family-level 5-way classification problems can be seen in Figure 14, Figure 15, and Figure 16 respectively.

B.2 Clothing Domain

B.2.1 Data Collection

We source our articles of clothing images and fine-grained annotations from the 20,000-image fine-grained attribute recognition subset of DeepFashion (Liu et al., 2016). Each image has annotations for garment super category (i.e, upper versus lower versus full-body garment), garment category (i.e., one of 50 coarse garment categories), and 26 attributes that represent different visual attributes of articles of clothing like texture/pattern (e.g., floral versus striped), sleeve length (e.g., long versus

Table 9: **Dataset Comparison: Species Classification.** Commonly used zero-shot species classification datasets are compared against our benchmarks with respect to taxonomic diversity. Note $*$ means some of the classes are not associated with any species (e.g., "bad" and "mix elongated" in Plankton dataset) or certain classes are not associated with a specific taxonomic path (e.g., "bats").

Dataset	Kingdoms	Phylum	Classes	Orders	Families	Species
CUB	1	1	1	12	37	200
NaBirds	1	1	1	20	64	404
Flowers102	1	1	5	28	47	102
Rare Species	1	5	15	85	202	400
Meta-Album						
Plankton*	3	8	13	36	42	62
Insects*	1	1	2	6	57	20
Insects 2*	1	1	2	8	48	83
Fungi	2	2	2	4	7	25
PlantNet	1	1	1	12	14	25
Med. Leaf	2	2	3	13	18	23
AWA*	1	1	1	9	25	50
Camera Traps						
Desert Lion	1	1	2	12	17	31
ENA24	1	1	2	8	12	20
Island	1	2	3	11	14	16
Orinoquia	1	1	3	9	20	28
OH Animals	1	1	5	8	21	37
Our Benchmarks						
Class-level	3	14	50	282	371	500
Order-level	3	11	38	100	459	1000
Family-level	3	10	33	130	200	2000

short sleeve), dress length (e.g., maxi versus mini length), neckline (e.g., crew versus v neckline), fabric type (e.g., denim versus cloth), and general fit/style (e.g., conventional versus loose).

B.2.2 Benchmark Construction

We leverage the full 20,000 pool of images across train-val-test split to sample from. First, images are grouped by their respective garment category and exact 26-bit attribute vector. We only include a paired garment categories and attribute combinations if at least 33 images match this combination exactly. Since clothing articles are naturally long-tailed with certain styles and garments much less common, this filtering step filtered down the original 50 garment categories and 2,705 observed attribute combinations to 23 garment categories and 127 eligible attribute combinations.

Since these attribute were designed for the attribute recognition instead of classification, we can not simply concatenate these sets of attributes and place them in front of the garment category to produce a natural language fine-grained class name. For example, with simple concatenation, "solid, long-sleeve, no-dress, v-neckline, cotton, conventional blouse" would be a class name being classified upon which, while not disastrous, could hurt downstream performance. To make fine-grained class names more natural sounding, we adopt a deterministic procedure to convert sets of attributes into a natural language class name: (1) omit attributes that represent default or absence of a specific attribute (i.e., "no-dress," "no-neckline," "conventional"); (2) replace unnatural attribute names with more commonly spoken synonyms (i.e., "solid-color" instead of "solid," "graphic-print" instead of "graphic," "long-sleeved" instead of "long_sleeve," "crew-neck" instead of "crew_neckline," "fitted" instead of "tight," "loose-fitting" instead of "loose"); (3) place attribute descriptors into a fixed grammatical order based on the attribute super categories (fit $\rightarrow$ length $\rightarrow$ pattern/style $\rightarrow$ sleeve $\rightarrow$ neckline $\rightarrow$ material); and then (4) append the garment category. Thus, the previously mentioned "solid long_sleeve no_dress v_neckline cotton conventional blouse" class name becomes "solid-color long-sleeved V-neck cotton blouse." Or, "tight floral maxi_length sleeveless square_neckline chiffon dress" becomes "fitted maxi-length floral sleeveless square-neck chiffon dress."

We then devise a three level hierarchy to organize these "image"-"fine-grained class" pairs for multiple granularity classification. To do this, we need to organize fine-grained classes into a semantically meaningful hierarchy of classes. The hierarchy of classes first splits across the super categories of the filtered set of 23 garment categories: lower, upper, and full-body garments. We then group the fine-grained classes within each super category based on their base garment category (e.g., sweater, blouse, etc.). This results in a natural, annotated three level hierarchy of classes. While looking through this constructed hierarchy, we noticed a few sub-optimal included categories that needed to be pruned manually. DeepFashion considered "coat" under the "full-body garment" super category which could get confusing depending on the specific type of coat. Moreover, the categories "top" and "jacket" are too vague and overlap heavily with other categories under the "upper-body garment" super category when visualized. Likewise, "sweatshorts," "sweatpants," "jeggings" overlap with existing categories in the "lower-body garment" super category. We visualize the final class hierarchy across the "upper-body garment" (see Figure 6), "lower-body garment" (see Figure 7), and "full-body garment" (see Figure 8). *Intermediate* difficulty tasks are sampled by selecting fine-grained classes from different garment categories under the same garment super category (e.g., "upper-body garment"). *Fine*-grained difficulty tasks are done between the available fine-grained classes under a single garment category. Figure 17 and Figure 18 respectively present an example of an intermediate/coarse-grained and fine-grained task from the clothing domain.

B.3 Groceries Domain

Data Collection

Unlike the other domains, we were provided a full class hierarchy to use with images for each natural language class name from the GroceryStoreDataset (Klasson et al., 2019). We visualize the collected class hierarchy in Figure 9.

Benchmark Construction

An important note is unlike the other domains, this class hierarchy does not have a balanced depth. For example, certain categories like "melon" have multiple types while others like "avocado" do not have included sub-categories. Thus, the *Intermediate* difficulty tasks are sampled by selecting fine-grained classes (aka leaf nodes in Figure 9) from different categories under the same super category (i.e., "Fruit," "Vegetables," or "Packaged foods"). However, if like "avocado" there is no sub-categorization at depth 3 in the hierarchy, the depth 2 category itself becomes the class name to classify against. *Fine*-grained difficulty tasks are only done between leaf nodes that have a depth 2 category like "melon" or "apple." Figure 17 and Figure 18 respectively present an example of an intermediate/coarse-grained and fine-grained task from the groceries domain.

B.4 Furniture Domain

Data Collection

We used the WordNet subtree rooted at `furniture.n.01` to retrieve candidate ImageNet Winter 2021 synsets, full paths to a specific object through Wordnet hierarchy. For each synset, we retained image files from the corresponding archive when at least 33 images were available and recorded the WordNet definition, lemma names, and WNID. This tree-search retrieval procedure resulted in 133 candidate synsets containing 134,151 images.

Benchmark Construction

We removed 61 unsuitable or redundant synsets. For example, we removed "double bed" and "twin bed" because they both are two vague and differentiating similar sized beds is difficult from visual cues alone. We manually reorganized furniture-relevant candidates into a hierarchy based on the remaining 72 synsets available (see Figure 10). This curation was necessary because the WordNet subtree is noisy at times and does not align with common sense semantic organization (e.g., mobility devices appearing in chair-related branches).

Intermediate difficulty tasks are sampled by selecting fine-grained classes (aka leaf nodes in Figure 10) from different categories under the same super category (i.e., "Seating‘," "Sleeping Furniture," "Storage furniture," or "Tables"). *Fine*-grained difficulty tasks are defined to classify between the available fine-grained classes under different categories. Figure 17 and Figure 18 respectively

present an example of an intermediate/coarse-grained and fine-grained task from the furniture domain.

B.5 KITCHENWARE & TABLEWARE DOMAIN

Data Collection

We constructed the kitchenware and tableware benchmark from the union of the WordNet subtrees rooted at `kitchen_utensil.n.01` and `tableware.n.01` and sampled the resulting referenced synsets from ImageNet Winter 2021. We deduplicated candidate synsets by WNID, so that a synset reachable from both roots was retained only once. For each candidate synset, we retrieved the corresponding ImageNet Winter 2021 archive, extracted standard image formats, and retained classes with at least 33 available images. We collected 90 candidate synsets containing 84,768 images using this process.

Benchmark Construction

Similar to how we constructed furniture dataset, we filter down from 90 to 30 leaf classes to get rid of sub-optimal classes. We then manually organize them into a sensible semantically-aligned hierarchy, leveraging existing WordNet annotations (see Figure 11). We also sample *Intermediate* and *Fine*-grained difficulty tasks using the categories and fine-grained categories similar to the Furniture domain dataset construction. Figure 17 and Figure 18 respectively present an example of an intermediate/coarse-grained and fine-grained task from the kitchenware and tableware domain.

B.6 MUSICAL INSTRUMENTS DOMAIN

Data Collection

Similar to the preceding data collection processes for "Furniture" and "Kitchenware & Tableware" domains, we constructed the musical instrument benchmark by sampling the WordNet subtree rooted at `musical_instrument.n.01` and grabbing the corresponding data in ImageNet Winter 2021 archives. We only retained sample synsets with at least 33 images stored. This broad retrieval stage produced 102 candidate synsets containing 85,943 images.

Benchmark Construction

We filter down from 102 to 32 leaf classes. We then manually organize them into a sensible semantically-aligned hierarchy, leveraging existing WordNet annotations (see Figure 12). We also sample *Intermediate* and *Fine*-grained difficulty tasks using the categories and fine-grained categories similar to the "Furniture" and "Tableware & Kitchenware" domain dataset construction. Figure 17 and Figure 18 respectively present an example of an intermediate/coarse-grained and fine-grained task from the musical instruments domain.

B.7 TOOLS DOMAIN

Data Collection

We constructed the tools benchmark from the WordNet subtree rooted at `tool.n.01` and sampled ImageNet Winter 2021 archive. We retained classes with at least 33 available images. This broad retrieval stage produced 208 candidate synsets containing 110,523 images.

Benchmark Construction

We then manually re-organized and filtered the large 208 available synsets into 93 leaf classes. To expedite our hierarchy construction, we built off the existing hierarchy defined in WordNet (see Figure 13). We also sample *Intermediate* and *Fine*-grained difficulty tasks using the categories and fine-grained categories similar to the "Furniture" and "Tableware & Kitchenware" domain dataset construction. However, due to the impressive scale in this domain, our constructed benchmark sometimes has depth 4 leaf nodes. In this case, we only include intermediate tasks if the node has at least one child with multiple children. For example, "Drills" is an intermediate task but not "Drill bits." Figure 17 and Figure 18 respectively present an example of an intermediate/coarse-grained and fine-grained task from the tools domain.

C Classify by Descriptions vs Black Box Methods Approaches Experimental Details

We conduct experiments to examine how Classify by Descriptions (CbD) compares with black box zero and few-shot image classification. We note, for all the experiments showcased in this section, ViT B/16 backbones are used unless otherwise stated.

C.1 Black-Box Models

To see how far CbD is from state-of-the-art performance, we need to compare them with simple black-box methods where we do not base classification predictions on intermediate natural language concepts/descriptors.

C.1.1 Zero-Shot Classification

In the zero-shot classification setting, we adopt the classification procedure introduced in the original CLIP paper (Radford et al., 2021). Specifically, given a query image i and a set of classes C, we construct a prediction, for each class $c \in C$, $s(i, t(c))$ where $t(c) =$ "A photo of $\{c\}$". The predicted class c^* for image i is just $c^* = \arg\max_{c \in C} s(i, t(c))$. We test this procedure with a variety of off-the-shelf VLMs, including CLIP (Radford et al., 2021), OpenCLIP (Cherti et al., 2023), BioCLIP 2 (Gu et al., 2025), and use scientific names for class names C.

C.1.2 Few-Shot Classification

We use a nearest neighbor approach to perform few-shot classification in a black-box manner. Specifically, given image sets I_c for each class $c \in C$, we can construct a per class image prototype $\bar{i}_c = \frac{1}{|I_c|} \sum_{i_c \in I_c} \phi(i_c)$ where ϕ is a frozen image encoder. Query images i are then predicted based on which classes' prototype has the highest normalized cosine similarity with the query image's embedding $\phi(i)$.

C.2 Classify by Descriptions

We perform Classify by Descriptions (CbD) (Menon & Vondrick, 2022) to represent zero-shot CBs. Prompt taken from CbD we used to produce descriptors for a provided class can be found in Figure 19. We use the closed-source GPT-4.1-mini LLM (, 2023; OpenAI, 2025) with a fixed random seed and temperature 0.6 for reproducibility. Our per-image class predictions are generated using the inference procedure described in CbD. Our class-less version of CbD keeps the same inference setup but does not prepend class names to the beginning of produced visual descriptors.

D Language Bottleneck Experimental Details

We conduct a series of experiments to investigate how important the language bottleneck is to Classify by Descriptions (CbD) performance. To accomplish this, we explored search algorithms, prompting strategies, and entry-level categorization.

D.1 Search Algorithms for descriptors

Methods for searching descriptors can be categorized as either self-supervised (Sehgal et al., 2025) or few-shot (Chiquier et al., 2024). Since ESCHER does not support binary task, we filtered out all binary tasks from *TaxoBench*. We also performed this experiment on a single seed of randomly sampled tasks and subsampled the class, order, and family-level collections of species classification tasks down to 50 each due to compute constraints. In the same vein, we used Qwen-3-32B-Instruct (Yang et al., 2025) with a maximum sequence length of 65536 to perform the LLM-based evolutionary search in both ESCHER (Sehgal et al., 2025) and LLM-Mutate (Chiquier et al., 2024) and locally hosted it on two NVIDIA A6000 GPUs.

Table 10: **Fine-Grained Species Classification.** Species-classification accuracy (%) is reported for different off-the-shelf VLMs and inference setups. Candidate species are sampled according to three degrees of taxonomic relatedness: class, order, and family.

Method	Class names	Model	No. of shots	Relatedness of species: Class	Order	Family
Random	–	–	–	20.00	20.00	20.00
Black-box Zero-shot classification	Yes	CLIP		28.06 ± 0.04	27.09 ± 0.30	26.37 ± 0.14
		OpenCLIP		25.00 ± 0.36	24.86 ± 0.19	24.53 ± 0.14
		BioCLIP 2		$\mathbf{96.16 \pm 0.14}$	$\mathbf{95.34 \pm 0.12}$	$\mathbf{95.14 \pm 0.09}$
Few-shot classification	No	CLIP	1	53.33 ± 0.60	45.96 ± 0.45	41.61 ± 0.25
			2	62.41 ± 0.41	54.48 ± 0.35	49.32 ± 0.28
			4	70.48 ± 0.39	62.30 ± 0.42	57.05 ± 0.15
			8	75.61 ± 0.46	68.48 ± 0.39	63.31 ± 0.17
Interpretable zero-shot CbD	Yes	CLIP		35.48 ± 0.71	32.19 ± 0.31	29.66 ± 0.17
		OpenCLIP		34.14 ± 0.61	30.97 ± 0.26	28.53 ± 0.13
		BioCLIP 2		$\mathbf{62.88 \pm 0.89}$	$\mathbf{60.65 \pm 0.39}$	$\mathbf{58.65 \pm 0.37}$
	No	CLIP		34.44 ± 0.69	30.30 ± 0.33	27.16 ± 0.17
		OpenCLIP		32.68 ± 0.59	30.08 ± 0.28	27.00 ± 0.13
		BioCLIP 2		28.84 ± 0.46	28.04 ± 0.31	25.17 ± 0.25

Table 11: **Fine-Grained Everyday Item Classification Tasks.** Accuracy (%) is reported for different off-the-shelf VLMs and inference setups across six everyday-item domains. Panel (a) evaluates intermediate-related candidate sets, while Panel (b) evaluates fine-related candidate sets. Domain averages are macro-averaged over classification episodes; Avg. is the macro average over the six domains.

(a) Intermediate-related candidate sets

Method	Class names	Model	No. of shots	Clothing	Groceries	Furniture	Kitchenware & tableware	Musical instruments	Tools
Random	–	–	–	21.43	7.68	30.00	29.17	36.25	35.00
Black-box Zero-shot image classification	Yes	CLIP		79.07 ± 0.71	62.42 ± 0.89	85.67 ± 1.94	86.85 ± 3.49	84.94 ± 2.31	72.72 ± 2.46
		OpenCLIP		84.56 ± 0.50	66.44 ± 0.68	92.48 ± 1.54	91.99 ± 1.68	88.97 ± 1.84	79.26 ± 2.74
Few-shot image classification	No	CLIP	1	49.59 ± 3.47	68.21 ± 1.77	66.63 ± 2.65	60.32 ± 4.06	66.37 ± 3.15	65.82 ± 3.02
			2	58.43 ± 2.94	79.38 ± 0.81	75.87 ± 2.14	73.42 ± 3.13	75.36 ± 2.35	76.29 ± 2.15
			4	70.27 ± 2.31	85.93 ± 0.68	80.97 ± 1.91	80.39 ± 2.07	82.98 ± 1.93	82.64 ± 1.64
			8	76.20 ± 1.92	89.43 ± 0.44	85.51 ± 1.60	84.32 ± 2.45	87.71 ± 1.28	85.96 ± 1.27
Interpretable zero-shot CbD	Yes	CLIP		81.42 ± 0.69	68.25 ± 1.11	86.91 ± 1.91	87.07 ± 2.90	91.86 ± 1.01	72.99 ± 1.64
		OpenCLIP		85.91 ± 0.63	67.24 ± 1.36	92.87 ± 1.33	90.96 ± 2.98	92.92 ± 1.20	81.12 ± 2.07
	No	CLIP		53.86 ± 0.96	26.81 ± 1.82	62.57 ± 3.81	62.06 ± 5.06	61.39 ± 3.37	58.98 ± 1.69
		OpenCLIP		59.31 ± 1.03	32.46 ± 1.38	69.45 ± 3.53	67.59 ± 4.09	63.41 ± 2.42	61.52 ± 2.49

(b) Fine-related candidate sets

Method	Class names	Model	No. of shots	Clothing	Groceries	Furniture	Kitchenware & tableware	Musical instruments	Tools
Random	–	–	–	25.63	30.28	27.69	25.85	27.38	23.51
Black-box Zero-shot classification	Yes	CLIP		54.17 ± 0.43	54.71 ± 0.37	69.81 ± 0.33	69.14 ± 0.92	53.59 ± 0.91	42.52 ± 0.27
		OpenCLIP		65.92 ± 0.36	61.16 ± 0.36	80.84 ± 0.44	72.67 ± 0.79	60.66 ± 0.58	50.43 ± 0.41
Few-shot classification	No		1	34.27 ± 0.63	56.70 ± 1.13	50.95 ± 0.80	44.67 ± 1.58	40.45 ± 1.38	40.39 ± 0.86
			2	38.13 ± 0.79	64.61 ± 1.17	59.89 ± 1.01	51.05 ± 1.26	47.93 ± 1.47	46.25 ± 0.74
			4	44.64 ± 0.83	71.28 ± 0.75	67.37 ± 0.80	58.49 ± 1.27	54.73 ± 1.73	53.07 ± 0.75
			8	50.44 ± 0.55	77.12 ± 0.69	72.40 ± 0.54	64.12 ± 1.00	60.31 ± 1.03	58.54 ± 0.56
Interpretable zero-shot CbD	Yes	CLIP		55.4 ± 0.39	57.23 ± 0.30	72.29 ± 0.37	70.16 ± 0.59	64.22 ± 0.77	43.64 ± 0.42
		OpenCLIP		67.44 ± 0.35	59.88 ± 0.33	82.71 ± 0.34	76.55 ± 0.68	67.09 ± 0.69	53.33 ± 0.51
	No	CLIP		34.25 ± 0.24	38.50 ± 0.21	41.85 ± 0.49	43.19 ± 0.64	38.99 ± 0.48	31.73 ± 0.34
		OpenCLIP		38.52 ± 0.30	40.06 ± 0.21	49.97 ± 0.40	42.58 ± 0.69	37.80 ± 0.62	35.58 ± 0.35

D.1.1 SELF-SUPERVISED EVOLUTIONARY SEARCH

We use the existing code and implementation details in ESCHER (Sehgal et al., 2025) to evaluate how well self-supervised CBs can perform.

In our experiments, we found the standard implementation with default hyperparameters (such as 50 evolution iterations and Pearson coefficient selection metric) resulted in an explosion of in-context sequence lengths passed to hosted LLM with only a few iterations. We realized the main culprit was that, at each iteration, the LLM is asked to produce whichever new descriptors might be helpful based on the history of all prior descriptors. Since ESCHER adds whatever is generated each time, this history has descriptors repeated across iterations, severely increasing the input token sequence length. Moreover, unbounded generation of "new" descriptors leads to major jumps in descriptor list lengths for each class which further explodes input sequence lengths. To make this generation more realistic and tractable, a new mutation prompt (seen in Figure 22) was introduced to bound per-iteration descriptor generation and only track added descriptors in each round. We argue the history is no less informative (i.e., retaining all unique descriptors added throughout evolutionary search) and bounded generation will approach unbounded as evolution iterations increase from a few (3-7) to the standard 50 proposed in the original implementation.

D.1.2 FEW-SHOT EVOLUTIONARY SEARCH

Unlike self-supervised CBs, few-shot CBs can use the visual features of a select few labeled samples from each classes' evaluation set. However, how exactly these visual features are extracted and translated into proposed per-class descriptor lists depends on the method being discussed.

In few-shot evolutionary search, candidate pools of per-class attribute sets are evolved over using a pre-defined objective to score candidates for sampling and informed mutation. LLM-Mutate (Chiquier et al., 2024) implements such an evolutionary search process using cross entropy over labeled images as the objective function. Due to the non-differentiable nature of the evolutionary search process, we realized non-differentiable objectives can be considered and, thus, used classification accuracy directly to get the best possible results from this type of methodology. To keep performance comparable, we use the same LLM instance ran for our self-supervised evolutionary search experiment and use the method's standard 100 evolutionary iterations per search. We also use a standard 8 few-shot support images per class to train on.

Table 12: **Search Algorithms for Classify by Descriptions.** Raw mean macro accuracies are reported for both self-supervised and few-shot descriptor evolutionary search.

Method	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine
Zero-shot															
CbD	30.02	27.62	27.72	38.32	30.87	25.28	32.64	75.73	40.64	57.00	41.62	51.60	40.90	47.67	29.82
Self-sup.															
ESCHER	29.17	27.95	25.36	57.08	29.48	19.05	27.90	63.67	43.70	67.50	36.10	41.90	40.48	34.78	28.76
Few-shot															
LLM-Mutate	**37.38**	**31.65**	**31.42**	**64.25**	**42.58**	**37.50**	**52.72**	**80.16**	**52.46**	**63.83**	**44.84**	**61.58**	**49.05**	**60.04**	**40.83**

Table 13: **Prompting Strategies for Zero-Shot Concept Bottlenecks.** Raw mean macro accuracies are reported for different prompting strategies.

Method	No. of shots	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools	
		Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine
CbD		34.15	30.37	27.04	47.11	37.14	26.76	38.16	63.45	41.91	59.27	42.68	59.37	38.82	56.80	32.05
full sentences		37.13	30.94	28.65	**72.70**	34.33	**44.12**	**42.44**	**83.88**	**62.41**	**80.93**	51.95	**84.79**	**43.90**	58.45	**36.85**
few-shot in-context	4	40.36	35.62	32.63	57.43	37.53	34.64	41.49	71.80	53.80	76.73	50.17	72.76	38.47	64.00	35.76
	8	**42.69**	**36.21**	**32.84**	58.17	**38.00**	37.30	42.42	72.16	54.34	76.40	**52.59**	72.46	39.38	**66.55**	36.01

D.2 Prompting Strategies

We prompt GPT-4.1-mini (OpenAI, 2025) for the full sentences and few-shot in-context prompting ablations to ensure the LLM is not the reason for better performance. Since the few-shot in-context prompting strategies requires constructing a support set disjoint from the test images, we needed to redo the visual descriptor for each seed sampled. To keep API costs reasonable, we used 5 seeds and averaged across for scores.

D.3 Entry-Level Categorization

In our paper, we see class names (i.e., scientific names in our benchmark) help CbD perform better with off-the-shelf VLM models, especially with models supervised over such class names (Gu et al., 2025). We proposed categories common to all class names could provide performance gains, grounding descriptors in a specific domain.

To test this theory, we had to produce such common categories for each of our classification problems. As described in Section 5.3, we can leverage the common natural language intermediate categories for everyday item classification. However, for species classification, we needed to iterate over the classes, orders, and families in our benchmark and producing corresponding common nouns for each. We used separate prompts with in-context examples to prompt GPT-4.1-mini (OpenAI, 2025) (see Figure 23, Figure 24, and Figure 25 for the exact prompt used for class, order, and family-level category generation).

In the experimental results, these per-category nouns are prepended instead of class names to descriptors. To keep results comparable with the preceding section, we replicate the 5 seed sampling for classification tasks. Results are presented in Table 14.

Table 14: **Entry-level Categorization.** Raw mean macro accuracies are reported to see the improvement associated with entry-level categorization.

Method	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine
Nothing	34.15	30.37	27.04	47.11	37.14	26.76	38.16	63.45	41.91	59.27	42.68	59.37	38.82	56.80	**32.05**
Entry-level Categories	**37.20**	**31.96**	**28.12**	**58.02**	**40.28**	**31.29**	**40.48**	**67.32**	**49.00**	**65.90**	**41.73**	**60.70**	**42.67**	**58.75**	31.59

E Open-vocabulary Descriptor Recognition Experimental Details

We conduct a series of experiments to investigate how detrimental to VLM is to downstream performance. To accomplish this, we explored different CLIP-style models, multi-modal large language models, and a dedicated VLM architecture to the CbD task.

E.1 Attribute-Centric CLIP Models

To see if any of the recent advancements in CLIP-style VLMs significantly improve open-vocabulary recognition capabilities over CLIP (Radford et al., 2021) and OpenCLIP (Cherti et al., 2023), we evaluate DCSM (Kang et al., 2025), BLIP-2 (Li et al., 2023), EVA02-CLIP (Fang et al., 2024), and SigLIP-2 (Tschannen et al., 2025) and see if they produce more reliable image–text similarity scores. We note, for BLIP-2, we utilize the learned image–query–token latent embeddings and perform cosine similarity with corresponding text feature from the underlying text encoder. Then, for SigLIP-2, we followed the model's documentation and set padding to "max length" instead of the default "True." Moreover, to keep performance comparable with the prompting strategies and entry-level categorization sections, we did a 5 seed sampling for classification tasks and averaged over the 5 seeds' results.

Table 15: **Attribute-Focused VLMs.** Raw mean macro accuracies comparing CLIP, DCSM. BLIP-2, EVA02-CLIP, and SigLIP-2 for no class name CbD.

Model	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine
CLIP	34.15	30.37	27.04	47.11	37.14	26.76	38.16	63.45	41.91	59.27	42.68	59.37	38.82	**56.80**	32.05
DCSM	28.78	25.20	23.26	40.00	32.64	18.84	33.51	55.96	35.83	41.47	34.88	58.69	34.75	52.52	28.93
BLIP-2	32.59	30.09	27.27	58.18	35.63	21.95	**43.59**	71.19	51.64	62.50	46.67	57.13	37.34	52.17	31.63
EVA02-CLIP	34.43	**31.61**	27.50	**65.81**	**42.76**	27.04	39.81	**72.22**	**51.93**	**65.57**	45.34	**62.35**	**44.01**	54.48	**36.93**
SigLIP-2	**35.10**	31.46	**27.79**	60.98	41.89	**31.39**	38.68	70.56	49.37	64.77	**46.69**	58.55	36.49	56.05	36.49

Table 16: **Multi-modal LLMs.** Raw mean macro accuracies for CLIP, an open-source and a proprietary mLLM with the no class name CbD inference scheme. See Appendix for details.

Model	Species			Clothing		Groceries		Furniture		Kitchenware & tableware		Musical instruments		Tools	
	Class	Order	Family	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine	Int.	Fine
CLIP	35.90	30.47	27.23	38.32	34.65	25.28	37.98	81.3	41.45	57.00	41.62	48.30	40.90	56.20	31.72
Qwen-3-VL-32B	43.71	38.44	32.89	**65.80**	**54.49**	**44.72**	47.10	67.85	49.08	82.18	58.57	68.78	54.75	60.04	38.64
GPT-4.1	**47.14**	**39.85**	**34.76**	44.78	41.54	16.87	**47.97**	**72.50**	**50.26**	**86.67**	**61.95**	**86.65**	**64.86**	**82.74**	**51.24**

E.2 Multimodal Large Language Models

Unlike CLIP VLMs, multi-modal LLMs (mLLMs) can not produce similarity scores between a query image and text. mLLMs instead produce next token probabilities. Unsurprisingly, descriptors with higher perplexity will result in lower normalized log probabilities than smaller ones. To avoid this issue, we ask LLMs to provide a 1-5 score on whether each descriptor matches the provided query image. We expect minimal bias towards one or the other based solely on ease to generate.

We perform zero-shot classification with recent open-source proprietary GPT-4.1 mLLM (OpenAI, 2025) and mLLM Qwen-3-VL-32B-Instruct (Yang et al., 2025). The exact prompts we used to perform true CbD with GPT-4.1 can be found in Figure 20. To mitigate API costs, we combined the per-class descriptors in a given classification task into shuffled 50 descriptor chunks and scored them all in a single forward pass with the image. We also used the 1 seed for classification tasks except did not subsample the class, order, and family-level species classification tasks down to 50 each.

Originally, we tried using the same inference scheme with Qwen-3-VL to speed up inference. However, we found the resulting performance was disastrous with random chance being 26.59% and 26.72% for intermediate and fine settings and this instantiation of Qwen-3-VL inference only getting 25.31% and 26.72%. Moreover, we noticed a tendency to output high scores, i.e., 5 which resulted in many ties. Given this, we slightly altered the prompt (See Figure 21) and, due to less expensive inference cost when compared with GPT-4.1, only did forward passes between descriptors of the same class. These changes significantly improved performance as seen in Table 16 and emphasizes the sensitivity of mLLMs to what is in-context during inference.

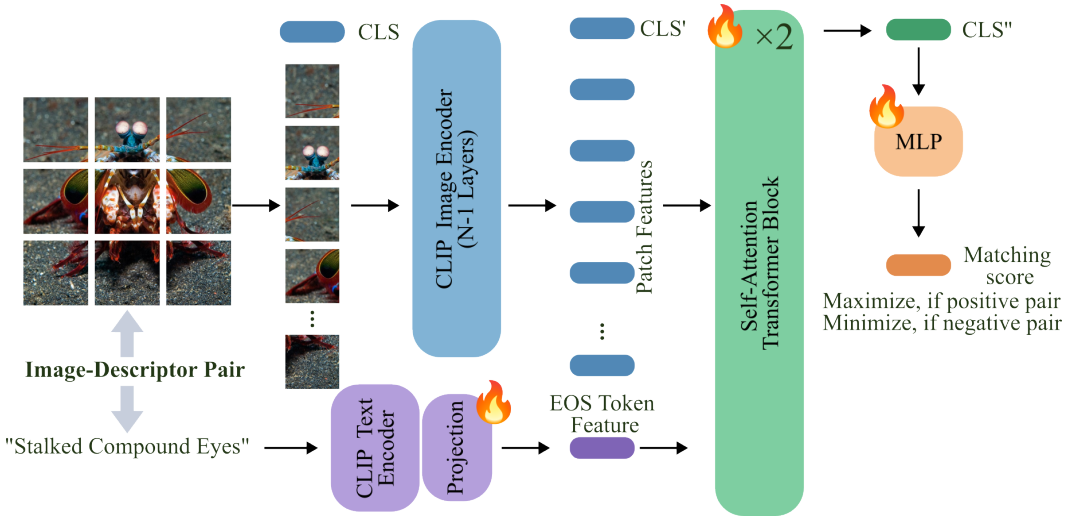

Figure 4: Diagram of our proposed pipeline. Every image-text pair seen is passed through frozen CLIP encoders to get penultimate image features and a linearly projected text feature. Light-weight transformer blocks attend across features to produce a pooled feature and map to a similarity score.

Table 17: **Dedicated Data and Model for Zero-Shot Species Classification.** Raw mean macro accuracies for different VLMs with and without training including our dedicated model architecture for CbD.

Method	Species		
	Class	Order	Family
CLIP	34.16	30.37	27.04
AdaptCLIPZS	35.14	30.88	27.42
AdaptCLIPZS^{FT}	35.13	31.10	27.51
DCSM	28.78	25.50	23.26
DCSM^{FT}	29.79	27.68	24.86
ours	35.85	32.76	29.57

E.3 Dedicated Model Training

Inspired by the improved performance of mLLMs over standard CLIP-style VLMs, we propose an architecture for matching images and text descriptions that explicitly looks at local image regions. The baseline CLIP approach is to produce a single global image embedding and a single global text embedding, and then measure cosine similarity between the two embeddings. Instead, we take embeddings of each patch from the image, and the embedding of the text description. We pass the patches and text embeddings through transformer blocks (see Figure 4).

E.3.1 Data Preparation

Searching over 67,595 training species, we construct a corresponding dataset of 748,904 images with the selection criteria from Section B.1.1 enforced. To produce the corresponding text-image pairs, we need per-class descriptor lists. We just use the standard CbD prompt (See Figure 19). Our training dataset aligns sets of images and natural language descriptors based on shared class names.

We note our dataloader randomly adds horizontal flips, affine transformations, and color perturbation to make our trained models more robust. Such transformations were inspired by AdaptCLIPZS (Saha et al., 2024) repository.

E.3.2 Training Procedure

We have training procedures based on what type of model is being trained. We build off the official training code of AdaptCLIPZS (Saha et al., 2024) to fine-tune CLIP. Since we are also trying to better recognize zero-shot descriptions of various species, we simply adapt the repo to use our training dataset and keep the hyperparameters from the training instance on iNaturalist dataset (e.g., 5 epochs).

For fine-tuning DCSM (Kang et al., 2025), we use the repository's set hyperparameters and adjust code to train on our benchmark for 5 epochs with batch size 256. Moreover, to avoid out of memory issues, we leverage the pairwise contrastive loss setup described in our paper's attribute-centric training section.

For our method, we use a light-weight transformer with 2 bi-directional BERT (Devlin et al., 2019) transformer blocks, 8 attention heads, and a hidden dimension of 768. Our model, like DCSM, outputs a singular similarity score when provided an image and text and, thus, uses the same training objective, batch size, and number of epochs. We train with the Adam optimizer initialized at zero and warmed up to 3e-4 until decaying back to zero, inspired by BERT pretraining (Devlin et al., 2019).

F Additional Experiments

F.1 Few-Shot Search for Descriptors: Discriminative versus Representative

The gains of LLM-Mutate (Chiquier et al., 2024) in Table 12 over ESCHER (Sehgal et al., 2025) and CbD. However, how does it improve the descriptors? Are the mutated descriptors a better match for the visual appearance or simply more discriminative? Specifically, are these descriptors

Table 18: **Dedicated Data and Model.** Normalized macro accuracies for CLIP-style VLMs with and without training including our dedicated architecture for CbD. See Section E.3 for more details.

Method	Class	Order	Family
CLIP	17.70	12.96	8.80
AdaptCLIPZS	18.93	13.60	9.28
AdaptCLIPZS^{FT}	18.91	13.88	9.39
DCSM	10.98	6.88	4.08
DCSM^{FT}	12.24	9.60	6.08
ours	**19.81**	**15.95**	**11.96**

learning more representative visual features or more discriminative among the in-context species? To test this, we sampled 25 classes with at least 25 species per class from our 50 class-level species classification tasks in *TaxoBench*. This resulted in 15 animal classes, 1 fungi class, and 7 plant classes. We then alphabetically sorted the 25 species names per class and partitioned the sorted list into 5 lists of 5 species. These represent the in-context species for LLM-Mutate to evolve over. We report the mean and standard error of classification accuracies across the 5 lists of species per class and categorized performance across the different kingdoms. To test representative quality of these evolved descriptors, we shuffle each classes' original 5 lists of 5 species into a corresponding 5 lists such that the sets are completely disjoint. We then record performance under these classification problems using the evolved descriptors.

In Table 19, the shuffled classes perform barely above the original zero-shot performance when using the learned descriptor sets from LLM-Mutate. We posit the evolutionary search might not actually be finding better visually matching descriptors for a species, but rather making them more discriminative for the classification task. Thus, even optimizing descriptors directly on a labeled training set does not necessarily produce a significantly better bottleneck.

Table 19: **Representative vs Discriminative Descriptors.** Performance gains are specific to classes shown in evolutionary search.

	Kingdoms			
	Animalia	**Fungi**	**Plantae**	**Total**
In-Context	47.60 ± 0.67	45.28 ± 5.41	42.58 ± 1.51	46.11 ± 0.37
Shuffled	36.96 ± 1.21	35.20 ± 4.79	35.54 ± 1.40	36.49 ± 1.28
Zero-Shot	34.98 ± 1.83	32.16 ± 4.61	34.08 ± 1.48	34.62 ± 1.34

Table 20: **Recognition of CUB Attributes.** Performance comparison of CLIP (Radford et al., 2021), AdaptCLIPZS (Saha et al., 2024), DCSM (Kang et al., 2025), and our method on recognizing per-image attribute annotations.

Category	Random	CLIP	AdaptCLIPZS	AdaptCLIPZS^{FT}	DCSM	DCSM^{FT}	ours
color	20.85	28.58	29.03	**33.83**	25.65	30.36	<u>32.52</u>
shape	29.17	34.17	34.17	32.92	<u>35.00</u>	32.50	**35.83**
pattern	40.00	<u>41.67</u>	40.33	40.33	**44.67**	37.33	41.33
size	33.33	33.33	33.33	<u>36.67</u>	33.33	<u>36.67</u>	**40.00**
average	30.84	34.44	<u>35.94</u>	34.66	26.25	34.22	**37.42**

F.2 Open-Vocabulary Recognition of Human-Labeled Attributes

While our proposed model architecture shows modest improvements over previous attribute-centric CLIP-style model architectures in Table 17, our performance gains are confounded by the fact that the underlying descriptors are not verified. Thus, in theory, our model could be improving without recognizing accurate descriptors. To remove this confounding variable, we performed an experiment on CUB's human annotated attributes, we had to make a few experimental design choices. We first filter out noisy attributes. We keep the high quality attributes established in the concept bottleneck paper Koh et al. (2020). We then organize attributes into groups of attributes referring to the same body part and type of attribute. For example, all attributes referring to beak color or body size

were combined into groups. Any groups with only one member were removed. Having established a series of classification problems referring to different attributes with clear possible choices, we sampled 10 images for each categories' possible values. Using the image annotations, we measured performance on how often a model scored the annotated attribute value the highest. Given the more concise, primitive concepts in this setting, we evaluated fine-tuned models trained over texts generated using CbA prompt (see Figure 26). Table 20 shows our proposed architectures in aggregate outperforms the prior methods DCSM and AdaptCLIPZS.

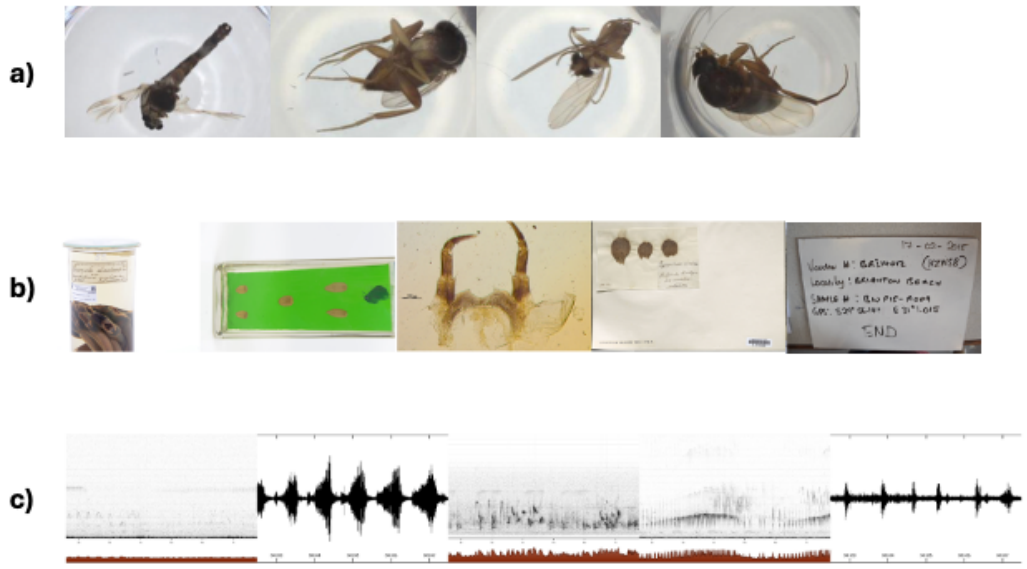

Figure 5: Examples from each type of suboptimal image included in TreeOfLife-200M dataset: (a) images taken in unrealistic lab settings (taken from BIOSCAN-5M), (b) images from museum collections, and (c) iNaturalist sound plots for birds.

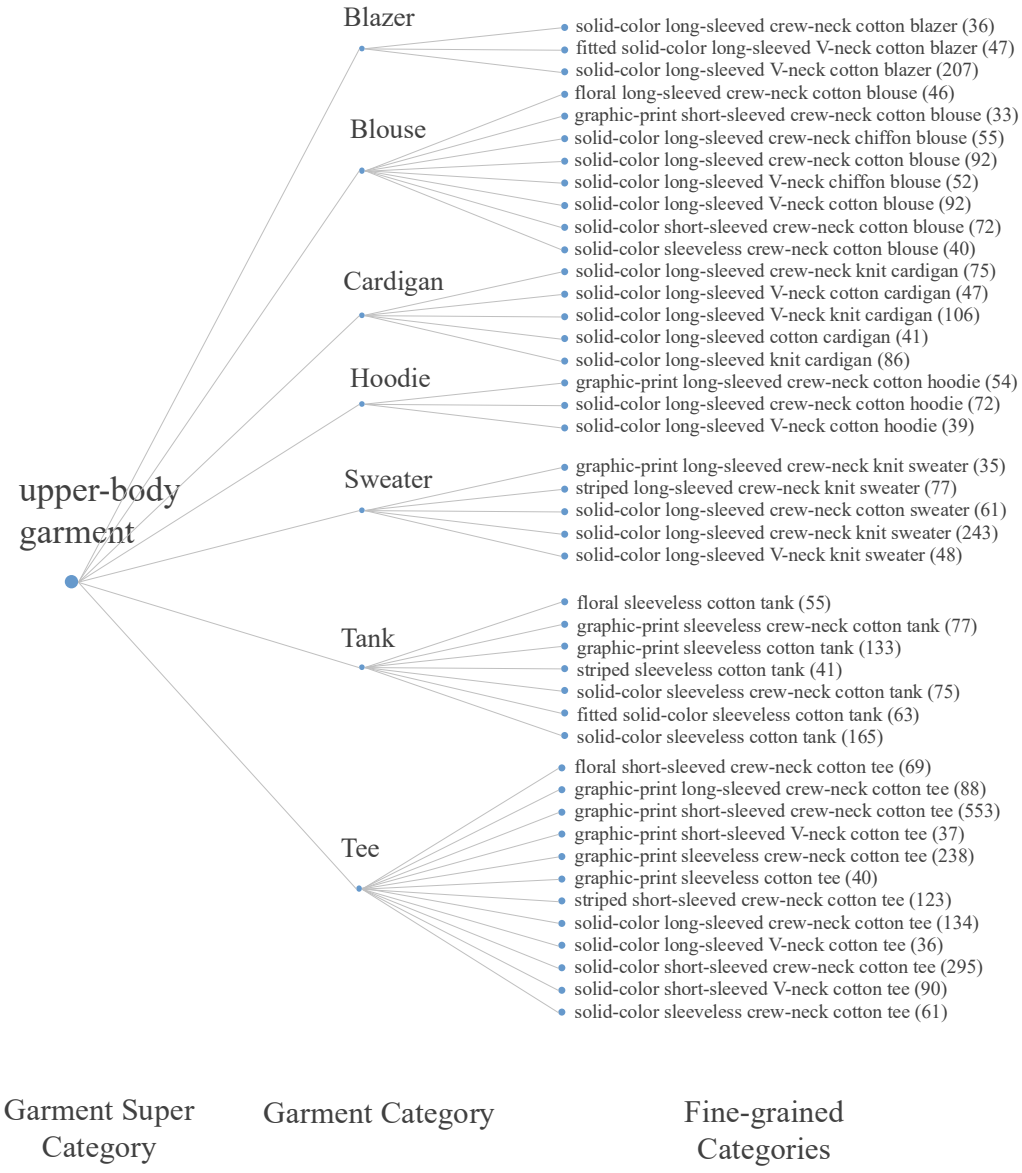

Figure 6: Class Hierarchy for Upper-Body Garment Super Category

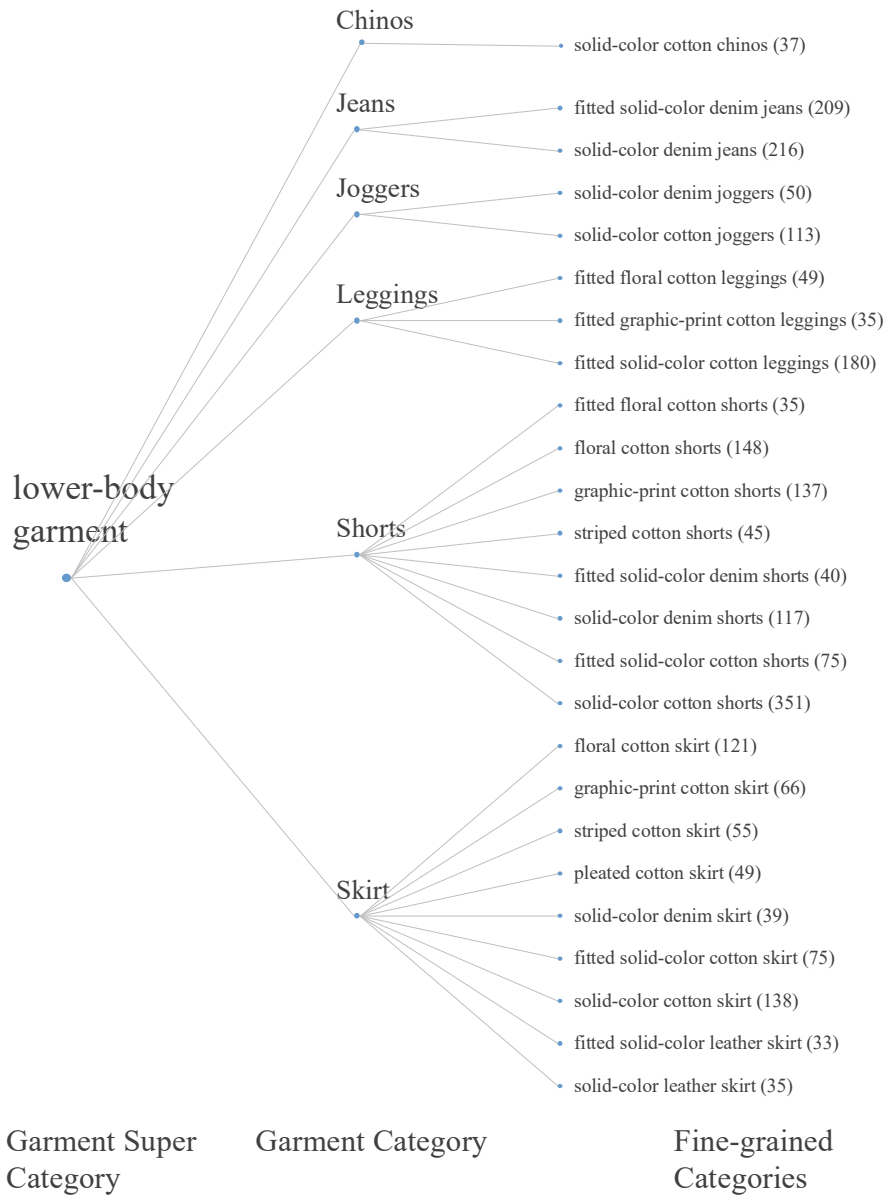

Figure 7: Class Hierarchy for Lower-Body Garment Super Category

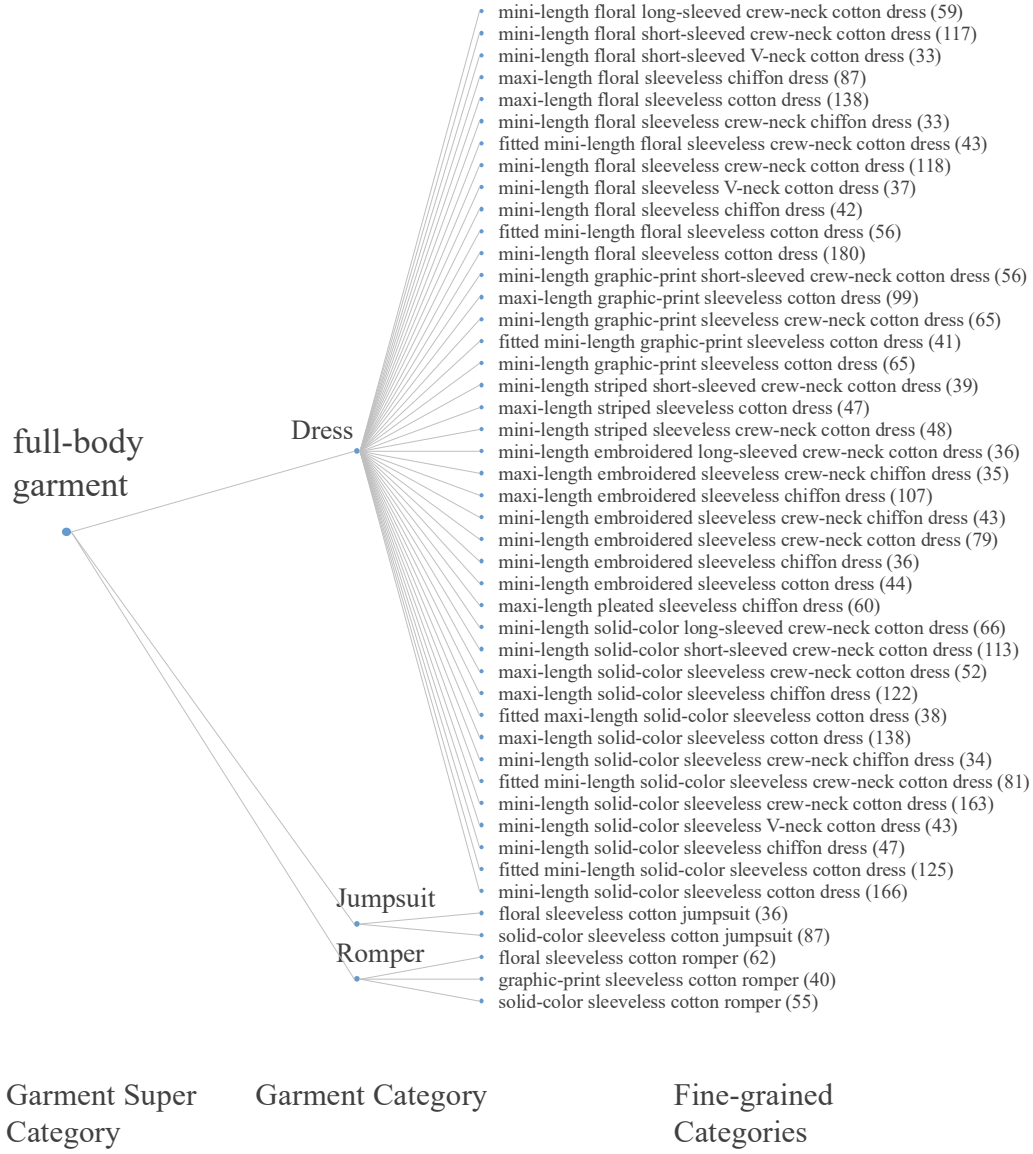

Figure 8: Class Hierarchy for Full-Body Garment Super Category

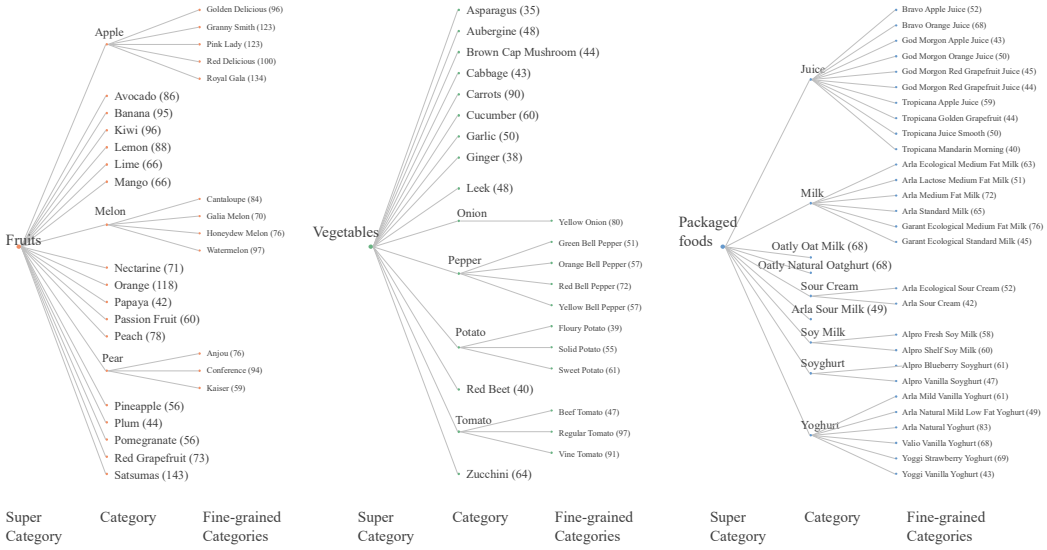

Figure 9: Class Hierarchies for Groceries Domain

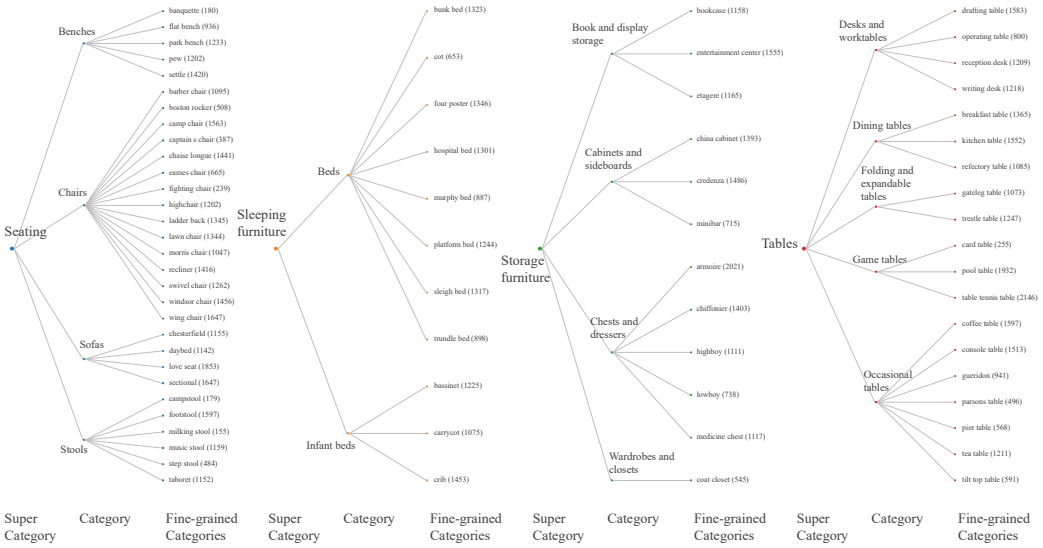

Figure 10: Class Hierarchies for Furniture Domain

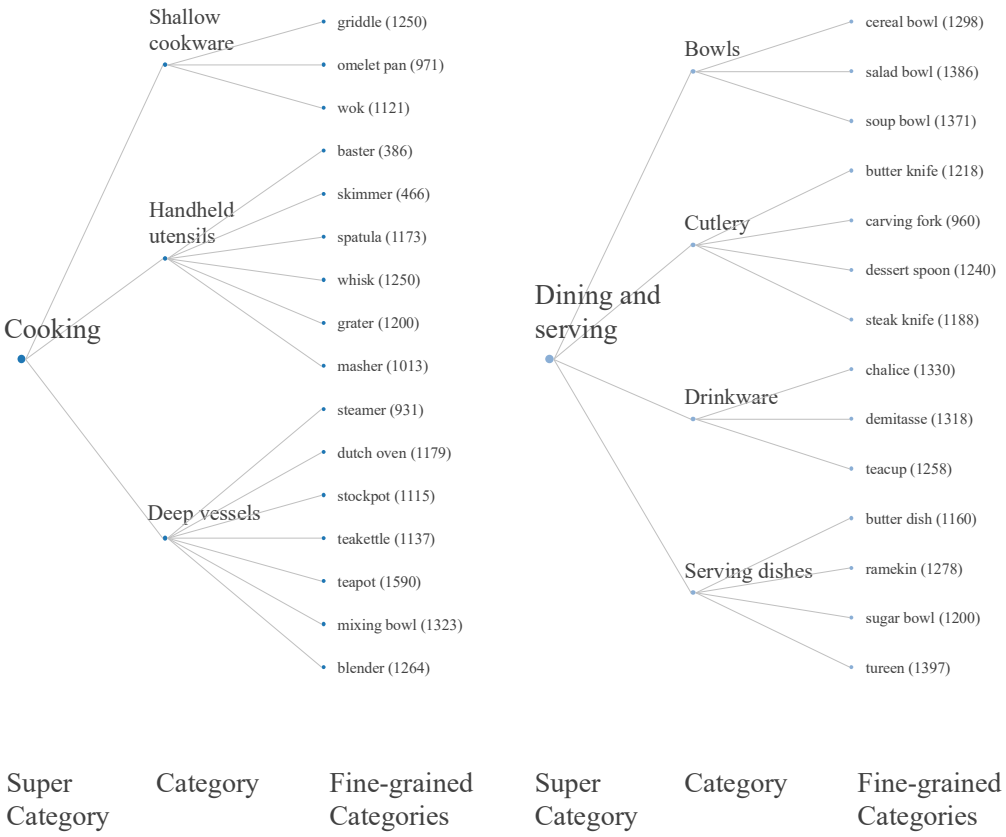

Figure 11: Class Hierarchies for Kitchenware & Tableware Domain

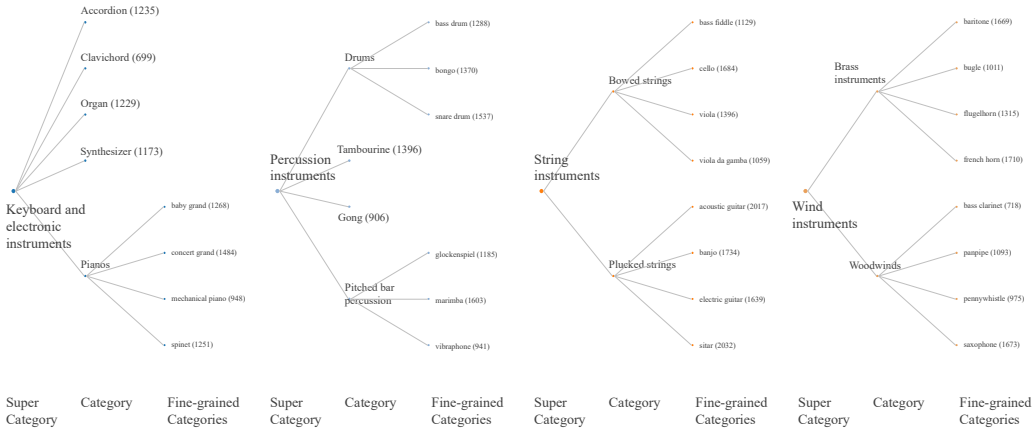

Figure 12: Class Hierarchies for Musical Instruments Domain

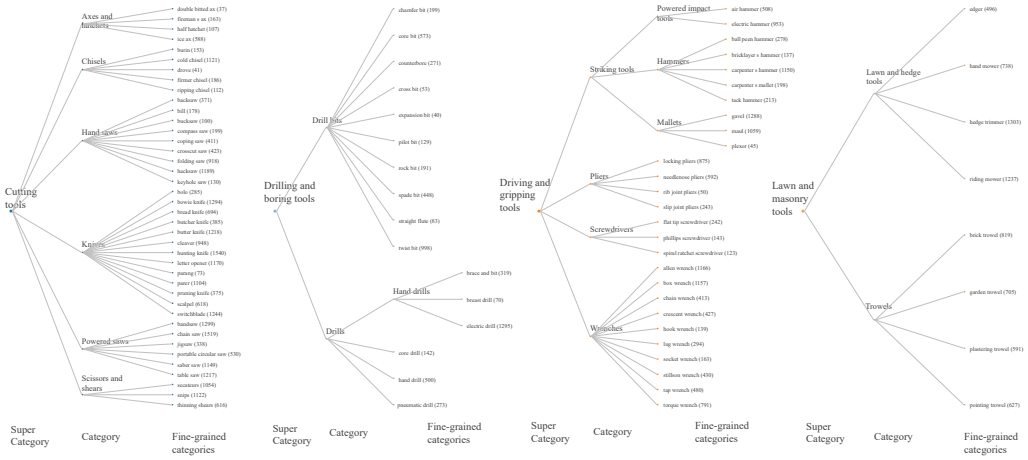

Figure 13: Class Hierarchies for Tools Domain

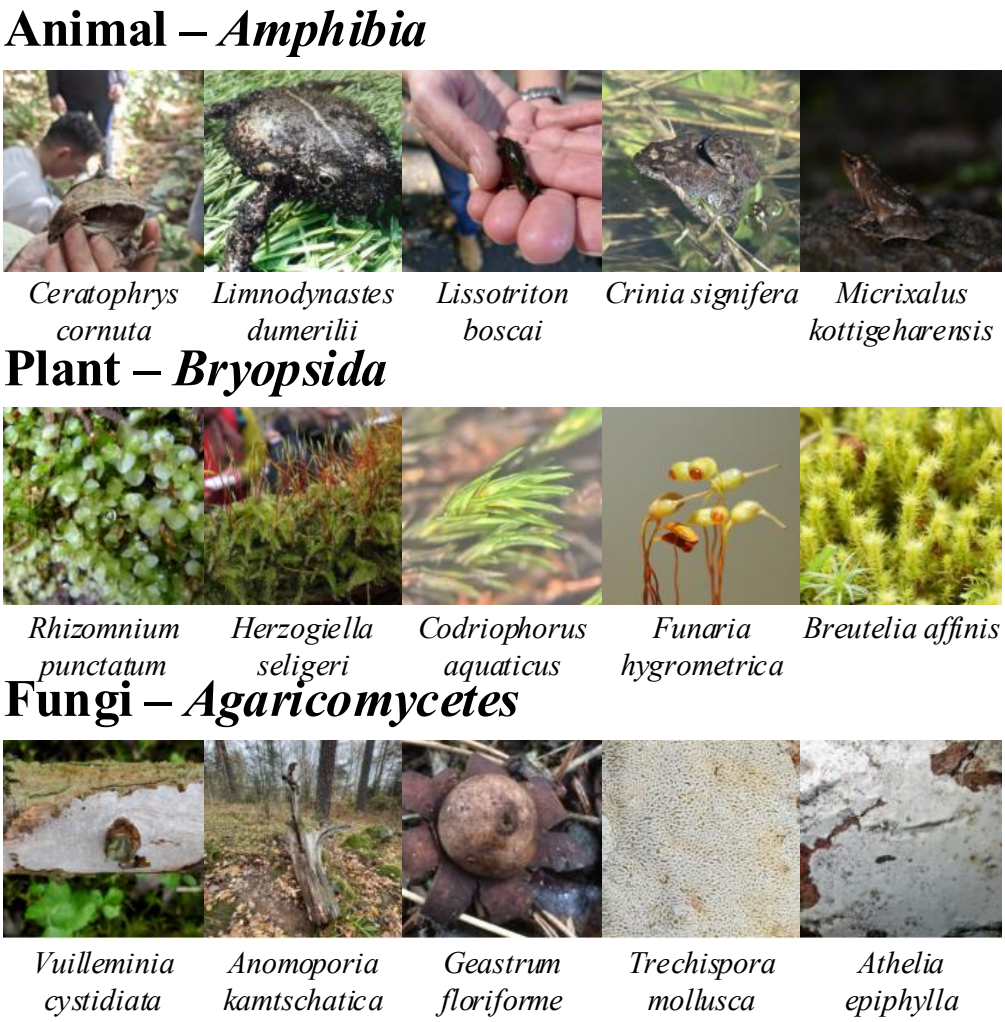

Figure 14: Examples of 5-way class-level species classification problems

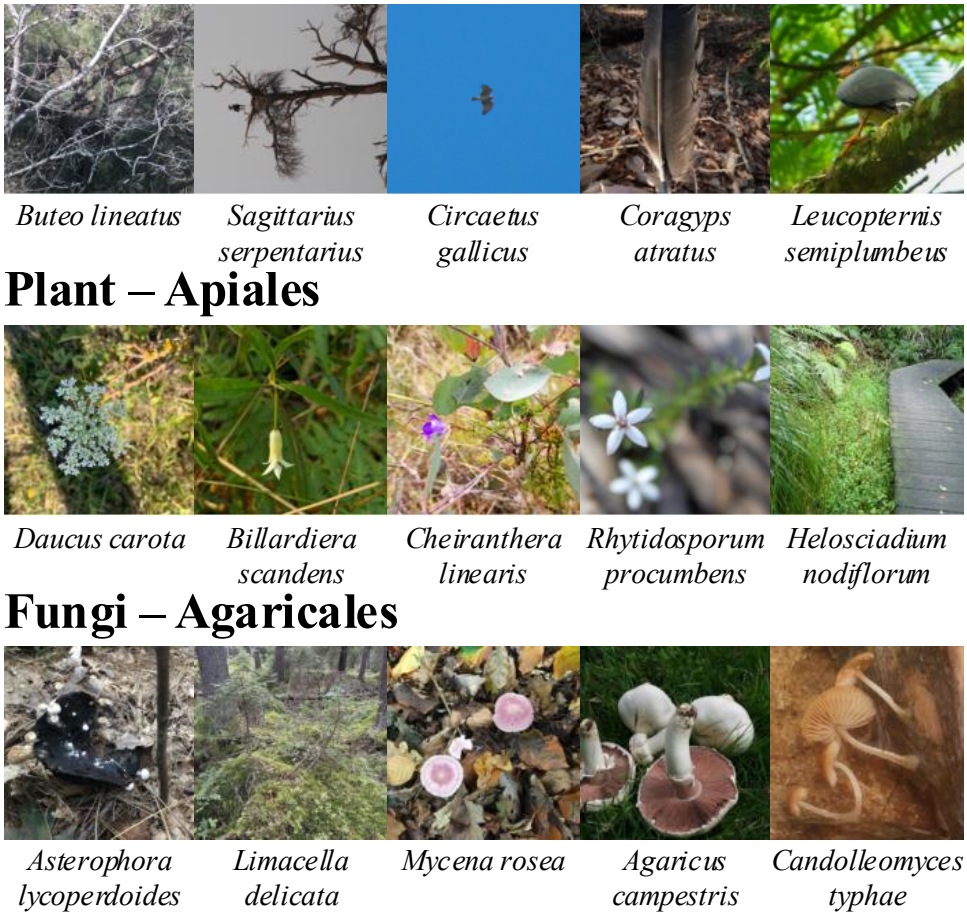

Figure 15: Examples of 5-way order-level species classification problems

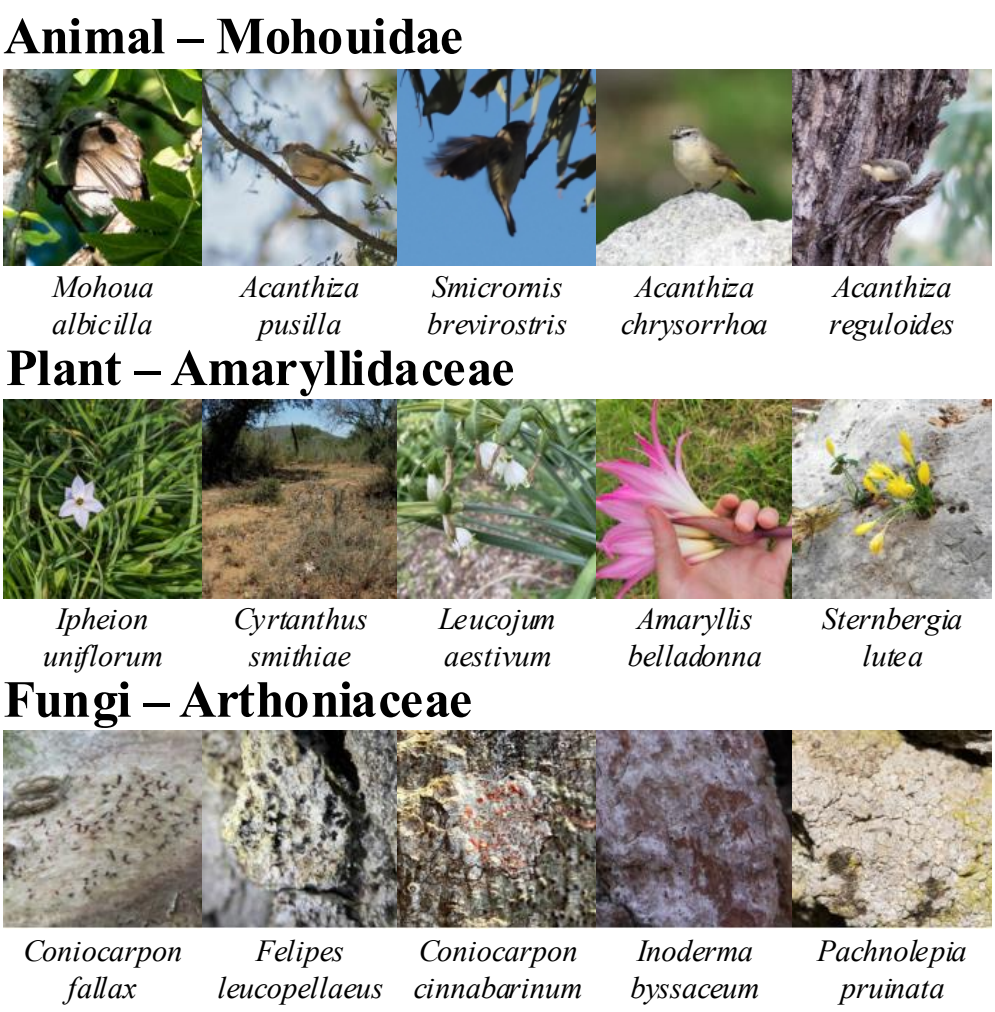

Figure 16: Examples of 5-way family-level species classification problems

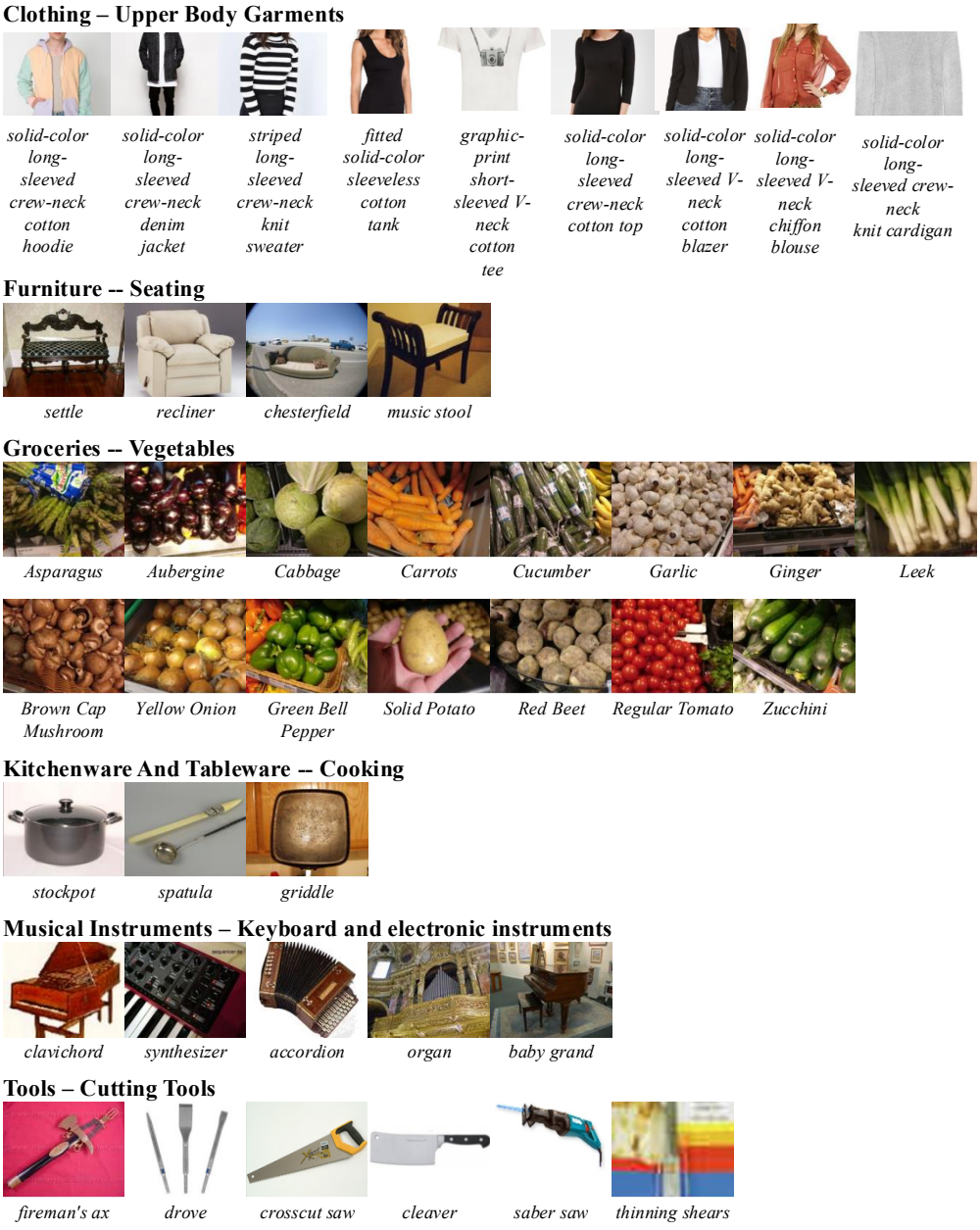

Figure 17: Examples of intermediate difficulty classification problems from the 6 everyday-item domains

Clothing – Hoodies



graphic-print long-sleeved crew-neck cotton hoodie



solid-color long-sleeved V-neck cotton hoodie



solid-color long-sleeved crew-neck cotton hoodie

Furniture – Benches



banquette



park bench



pew



flat bench



settle

Groceries – Apples



Golden Delicious



Pink Lady



Granny Smith



Royal Gala



Red Delicious

Kitchenware And Tableware – Deep Vessels



blender



teapot



steamer



stockpot



dutch oven



mixing bowl



teakettle

Musical Instruments – Pianos



mechanical piano



spinet



baby grand



concert grand

Tools – Hand Drills



breast drill



brace and bit



electric drill

Figure 18: Examples of fine-grained classification problems from the 6 everyday-item domains

```
Q: What are useful visual features for distinguishing a lemur in a
photo?
A: There are several useful visual features to tell there is a
lemur in a photo:
- four-limbed primate
- black, grey, white, brown, or red-brown
- wet and hairless nose with curved nostrils
- long tail
- large eyes
- furry bodies
- clawed hands and feet

Q: What are useful visual features for distinguishing a television
in a photo?
A: There are several useful visual features to tell there is a
television in a photo:
- electronic device
- black or grey
- a large, rectangular screen
- a stand or mount to support the screen
- one or more speakers
- a power cord
- input ports for connecting to other devices
- a remote control

Q: What are useful features for distinguishing a { class } in a
photo?
A: There are several useful visual features to tell there is a {
class } in a photo:
-
```

Figure 19: **Classify by Descriptions (CbD) Prompt.** { class } represents the class name in question.

```
You are shown ONE { type } photograph.


For every numbered visual descriptor, score how well the visible
subject matches it:
1 = not visible or clearly does not match
2 = weak/uncertain evidence
3 = partial or moderate match
4 = strong match
5 = clearly and directly visible

Rules:  - Use only what is visible in the photograph.
- If a feature is not visible or you are unsure, choose a lower
score.
- The numbered descriptors are independent; they are not categories
or options.
- Return ONLY a JSON list of exactly {{ number of items } }
integers, each 1 through 5.
- Do not write prose, markdown, descriptor text, names, or labels.
Descriptors:  { items }
```

Figure 20: **GPT-4.1 Prompt for Attribute Recognition.** { items } represents the bulleted list of generated visual descriptors for current batch of classes. { number of items } is the number of generated visual descriptors in this forward pass.

```
You are shown ONE { type } photograph.

For each numbered visual descriptor, judge how well the visible {
type } in the photo matches that descriptor.

Scoring rubric:
1 = Not visible or clearly does not match.
2 = Weak match; only a small part may match, or evidence is very
uncertain.
3 = Partial/moderate match; some visible evidence matches, but not
strongly or not completely.
4 = Strong match; descriptor is mostly supported by visible
evidence.
5 = Excellent match; descriptor is clearly and directly visible.

Rules:  - Use ONLY what is visible in the photo.
- If the relevant feature is not visible, score 1.
- If unsure, use a lower score.
- Output ONLY a Python/JSON-style list of integers.
- The list length MUST equal the number of descriptors.
- Each entry MUST be an integer from 1 to 5.
- Do NOT include explanations, descriptor text, names, or markdown.
Descriptors:  { items }
```

Figure 21: **Qwen-3-VL Prompt for Attribute Recognition.** { items } represents the bulleted list of generated visual descriptors for current batch of classes. { number of items } is the number of generated visual descriptors in this forward pass. { type } is either "species" or "product" depending on whether we are doing species classification on an everyday item domain.

```
You are a helpful assistant that helps people distinguish between
objects by generating high quality visual descriptions and
attributes.
Our algorithm is having trouble distinguishing a { clsA } from a {
clsB }.  Please generate new visual attributes or descriptors of {
clsA } and { clsB } that could help distinguish the two classes.
Our algorithm runs many iterations.  In each iteration, it produces
a list of descriptors for { clsA } and a list of descriptors for
{ clsB }.  For each iteration, you will be given the following
information:

    1. A list of added descriptors for { clsA }
    2. A list of added descriptors for { clsB }
    3. The proportion of real instances of { clsA } that were
       mistaken for { clsB }.

Your goal is to output a 1-2 descriptors for { clsA } and a 1-2
descriptors for { clsB } that will better visually distinguish the
two classes.  Output two lists of descriptors:  one for { clsA },
the other for { clsB }.
```

Figure 22: **LLM Prompt for more controlled self-supervised mutation of class descriptor lists.** { clsA } and { clsB } represent the two classes being currently confused with each other based on the chosen selection metric (e.g., Pearson coefficient).

```
You are given a scientific taxonomic class name (typically Latin).

Your task is to return the most common entry-level category noun in
plain English that corresponds to this class.

Rules:
- Output must be a common noun used in everyday language.
- Output should be lowercase.
- Output should be a single word whenever possible.
- A two-word phrase is allowed when necessary.
- No punctuation.
- Return ONLY the noun/phrase and nothing else.

Examples:

Input:
Aves
Output:
bird

Input:
Insecta
Output:
insect

Input:
Mammalia
Output:
mammal

Input:
Arachnida
Output:
arachnid

Input:
Amphibia
Output:
amphibian

Input:
Liliopsida
Output:
flowering plant

Now process the following taxonomic class:

{ class }
```

Figure 23: **Prompt for Entry-level Category Generation Class Level.** { class } represents the passed in scientific class name (e.g., Aves or Mammalia).

```
You are given a scientific taxonomic order name (typically Latin).

Your task is to return the most common entry-level category noun in
plain English that corresponds to this order.

Rules:  - Output must be a common noun used in everyday language.
- Output should be lowercase.
- Output should be a single word whenever possible.
- A two-word phrase is allowed when necessary.
- No punctuation.
- Return ONLY the noun/phrase and nothing else.

Examples:

Input:
Rodentia
Output:
rodent

Input:
Artiodactyla
Output:
even-toed ungulate

Input:
Perissodactyla
Output:
odd-toed ungulate

Input:
Accipitriformes
Output:
bird of prey

Input:
Anseriformes
Output:
waterfowl

Input:
Coleoptera
Output:
beetle

Input:
Araneae
Output:
spider

Now process the following taxonomic order:
{ class }
```

Figure 24: **Prompt for Entry-level Category Generation Order Level.** { class } represents the passed in scientific class name (e.g., Araneae or Coleoptera).

```
You are given a scientific taxonomic family name (typically Latin).

Your task is to return the most common entry-level category noun in
plain English that corresponds to this family.

Rules:  - Output must be a common noun used in everyday language.
- Output should be lowercase.
- Output should be a single word whenever possible.
- A two-word phrase is allowed when necessary.
- No punctuation.
- Return ONLY the noun/phrase and nothing else.

Examples:

Input:
Felidae
Output:
cat

Input:
Canidae
Output:
dog

Input:
Ursidae
Output:
bear

Input:
Cervidae
Output:
deer

Input:
Hominidae
Output:
great ape

Input:
Accipitridae
Output:
hawk or eagle

Now process the following taxonomic family:
{ class }
```

Figure 25: **Prompt for Entry-level Category Generation for Family Level Classification.** { class } represents the passed in scientific family (e.g., Accipitridae or Hominidae).

```
# Generate visual descriptors
## Task
Generate visual descriptors of a provided species and return them
in a JSON format.

## Instructions
1.  **Extract only visual descriptors** that can be observed in
photos (size, color, shape, texture, patterns, etc.)
2.  **Keep descriptors concise** - use short phrases, not full
sentences
3.  **Use relative sizes** instead of exact measurements
(``medium-sized'' instead of ``1-2 m tall'')
4.  **Use common sense** about referenced and related species to
help generate descriptors
5.  **Do not reference species, genus of species, or families of
species in descriptors**

## Output Format
json { ``class'':  ``[organism name]'',
``visual_descriptors'':  [
``descriptor1'',
``descriptor2'',
``descriptor3'',
``descriptor4'',
``descriptor5'' ] }

### Examples
### Example 1:

**Species:** Notharchus tectus
**Output:**
json { ``class'':  ``Notharchus tectus'',
``visual_descriptors'':  [
``relatively large head'',
``contrasting black and white pattern'',
``short, dark-colored legs'',
``short neck'',
``dark eyes'',
``stocky body'',
``prominent crest'',
``medium-length tail'',
``white underparts'',
``black upperparts'',
``short, black cone-shaped beak'' ] }
...
## Species:  { class }
## Response:  Return only the JSON schema - no additional text or
explanations.
```

Figure 26: **Classify by Attribute (CbA) Prompt.** { class } represents the passed in class name.