Self-Evolving Visual Concept Library using Vision-Language Critics

Supplementary Material

1. Appendix

ESCHER presents a modular framework for learning a library of visual concepts. In this section, we present additional discussions on the reasoning and validity of several modules ESCHER uses internally.

- §1.1: How does ESCHER's performance change without history conditioning?
- §1.2: How much impact does the visual critic have in ESCHER's iterations?
- §1.4: How does ESCHER perform with different disambiguation heuristics?
- §1.5: ESCHER's history-conditioned prompt.
- §1.6: Qualitative results of ESCHER's learned library.
- §1.7: Practical considerations for using ESCHER.
- §1.8: ESCHER's convergence properties.
- §1.9 Bayesian formulation for CBMs.
- §1.11 Additional backbone ablation with ViT-B/32.

1.1. ESCHER without history conditioning

ESCHER utilizes a history conditioning component to store pairs of concepts which have been disambiguated more than once. This pairwise history is provided in the input context to the model alongside with the feedback from the visual critic. To understand the effectiveness of the history conditioning component, we conduct additional experiments on the LM4CV domain by ablating the history conditioning model.

Setup. We replace the history conditioned concept space with a simpler prompt that doesn't use the history conditioning. The overall structure of the original prompt is left unchanged, except for the removal of previous conversations as well as the removal of a line with feedback parsing instructions.

Results. Results are showcased in Table 1. We find that the history conditioning is useful in almost all cases other than the Food101 dataset and the Oxford Flowers102 dataset, where the history conditioning ablation achieves the same accuracy as other methods.

ViT-L-14	LM4CV	Ours	Ours-H
CIFAR-100	84.48	**89.63**	86.73
CUB-200-2011	63.26	**83.17**	81.58
Food101	94.77	**94.97**	**94.97**
NABirds	76.58	**78.21**	77.35
Oxford Flowers	94.80	**96.86**	**96.86**
Stanford Cars	86.84	**93.76**	88.09

Table 1. Top-1 accuracy for LM4CV+ESCHER with and without history conditioning. We find that history conditioning helps improve the model performance in the majority of the datasets.

1.2. Visual Critic Ablations

ESCHER relies on a visual critic for classes that are confused for each other. In this section, we investigate whether the quality of the visual critic impacts the concepts generated by ESCHER.

Setup. We consider two experiment to study the impact of the visual critic. First, we replace the score matrix S with a randomly generated score matrix and use this uninformative matrix to sample new concepts. If ESCHER relies on the VLM critic, we expect to observe considerable performance deterioration. Second, we collect logits from a well-calibrated VLM critic and qualitatively compare the true confusion matrix and the correlation matrix calculated with various heuristics. These matrices have the same shape ($\mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ and we expect to see similar trends emerge (despite differing absolute values).

Results. We showcase results in Table 2 and Figure 1. We observe that without the visual critic, each iteration devolves to a random search over disambiguation pairs. The search is not completely unguided, as the decay rate helps the model avoid repetitive queries. Yet, the ablation underperforms LM4CV+ESCHER in all datasets.

Furthermore, our qualitative study reveals that ESCHER's heuristic closely aligns with the ground-truth confusion matrix, without using any labels. However, ESCHER's heuristic is extremely sensitive to minute errors and is symmetric about the first diagonal, leading to slightly suboptimal disambiguation.

1.3. ESCHER with low quality initial libraries

ESCHER primary objective is to augment a CBM training loop by using VLM feedback to grow a library of natural language concepts. We have demonstrated that ESCHER enhances the performance of various algorithms under a fixed library of components. In this section, we explore ESCHER's behavior when the initial library of concepts is small or incomplete.

ViT-L-14	LM4CV	Ours	Ours-H-VC
CIFAR-100	84.48	**89.63**	86.76
CUB-200-2011	63.26	**83.17**	82.41
Food101	94.77	**94.97**	94.95
NABirds	76.58	**78.21**	77.29
Oxford Flowers	94.80	**96.86**	96.08
Stanford Cars	86.84	**93.76**	88.12

Table 2. Top-1 accuracy for LM4CV, LM4CV+ESCHER, and LM4CV without a visual critic or history conditioning. We find that the visual critic improves model performance in all cases.

Setup. We start with the original concepts generated with `gpt-3.5-turbo` and randomly subsample to 25% and 50% of the original number of per-class concepts. We use CbD as the testing domain for this experiment.

Results. Results are reported in Table 3. We find that ESCHER is able to sufficiently recover it's base performance and, surprisingly, in some cases the lack of initial concepts allows ESCHER to find higher quality concepts than the original ones, enabling better overall performance.

ViT-L-14	CbD	CbD 50%	CbD 25%	Ours	Ours 50%	Ours 25%
CIFAR-100	76.20	73.50	67.50	**77.80**	77.80	78.00
CUB-200-2011	62.00	60.83	62.50	**63.33**	63.00	63.17
NABirds	53.61	54.38	54.26	54.30	55.15	**55.44**
Oxford Flowers	79.41	76.47	72.55	81.37	80.39	**83.33**
Stanford Cars	75.65	75.28	74.91	**77.14**	76.77	76.52

Table 3. Top-1 accuracy for CbD+ESCHER with different qualities of initial libraries. We find that ESCHER achieves the best performance for this dataset and, in some cases, benefits from less initial concepts.

1.4. Visual Concept learning with diverse critics

ESCHER allows multiple ways of specifying the heuristic for disambiguating label pairs. Each heuristic has its own advantages and shortcomings. In this experiment, we conduct a quantitative study on the different heuristics and their impact on performance.

Setup. We target the finetuning experiments with LM4CV and train our model with three different disambiguation heuristics: *PCA+Correlation* (PCA), *Earth Mover's Distance* (EMD), and *Pearson's Correlation* (PCC). We retrain LM4CV + ESCHER using the same hyper parameters as used in the best performing *Top-k Confusion* experiments.

Results. Results are reported in Table 4. We find that ESCHER performs best with *Top-k Confusion*. This heuristic also offers fine-grained control over the sensitivity of which pairs should be disambiguated. Figure 1 and Figure 2 offer a qualitative analysis of the confusion matrices for three datasets.

ViT-L-14	Baseline	Ours	Ours +PCA	Ours +EMD	Ours +PCC
CIFAR-100	84.48	**89.63**	86.58	86.71	86.44
CUB-200-2011	63.26	**83.17**	80.34	78.91	75.13
Food101	94.77	**94.97**	94.82	94.89	94.77
NABirds	76.58	**78.21**	76.99	76.87	76.32
Stanford Cars	86.84	**93.76**	87.54	87.35	86.94

Table 4. Top-1 accuracy for LM4CV+ESCHER using various disambiguation heuristics. We find that *Top-k Confusion* offers the most fine-grained control over the sensitivity of the disambiguation and performs the best empirically.

1.5. Zero shot prompts.

We show the prompts we used to self-evolve the library in ESCHER in Figure 3. The prompts are designed to make use of a scratchpad. Specifically, when ESCHER asks an LLM to disambiguate two classes (Fig.3), it first requests a reasoning for the proposed concept, then the concept itself in a separate JSON field. Without the additional reasoning field, the LLM tends to merge the explanation and concept, producing a lengthy concept that exceeds the CLIP text encoder's context length and prematurely halts ESCHER's iterative loop.

1.6. Additional qualitative results

We highlight additional qualitative results of images and classes that become better separated because of ESCHER's iteration process in Figure 4.

1.7. Practical considerations for using ESCHER

Experimental setup. Evolving concepts with ESCHER does not require a large computational footprint. We run our experiments on a server node with 10 NVIDIA A40 GPUs (each with 40 GB of VRAM), which allowed us to parallelize experiments. We were also able to replicate our experiments on a single NVIDIA RTX 2080 Ti GPU (12 GB of VRAM). For queries to large language models (LLMs), we primarily rely on externally hosted models (such as `gpt-3.5-turbo-0125`). For local models (e.g. `meta-llama/Llama-3.3-70B-Instruct`), we use `vLLM` [3] to host a local server. However, our framework is compatible with any LLM inference framework that allows hosting an OpenAI compliant server.

LLM queries per experiment. For reference, each iteration dispatches about 100 queries (depending on the subsampling hyper-parameters). However, with history condi-

Qualitative analysis of 'Pearson's Correlation' as a disambiguation heuristic.

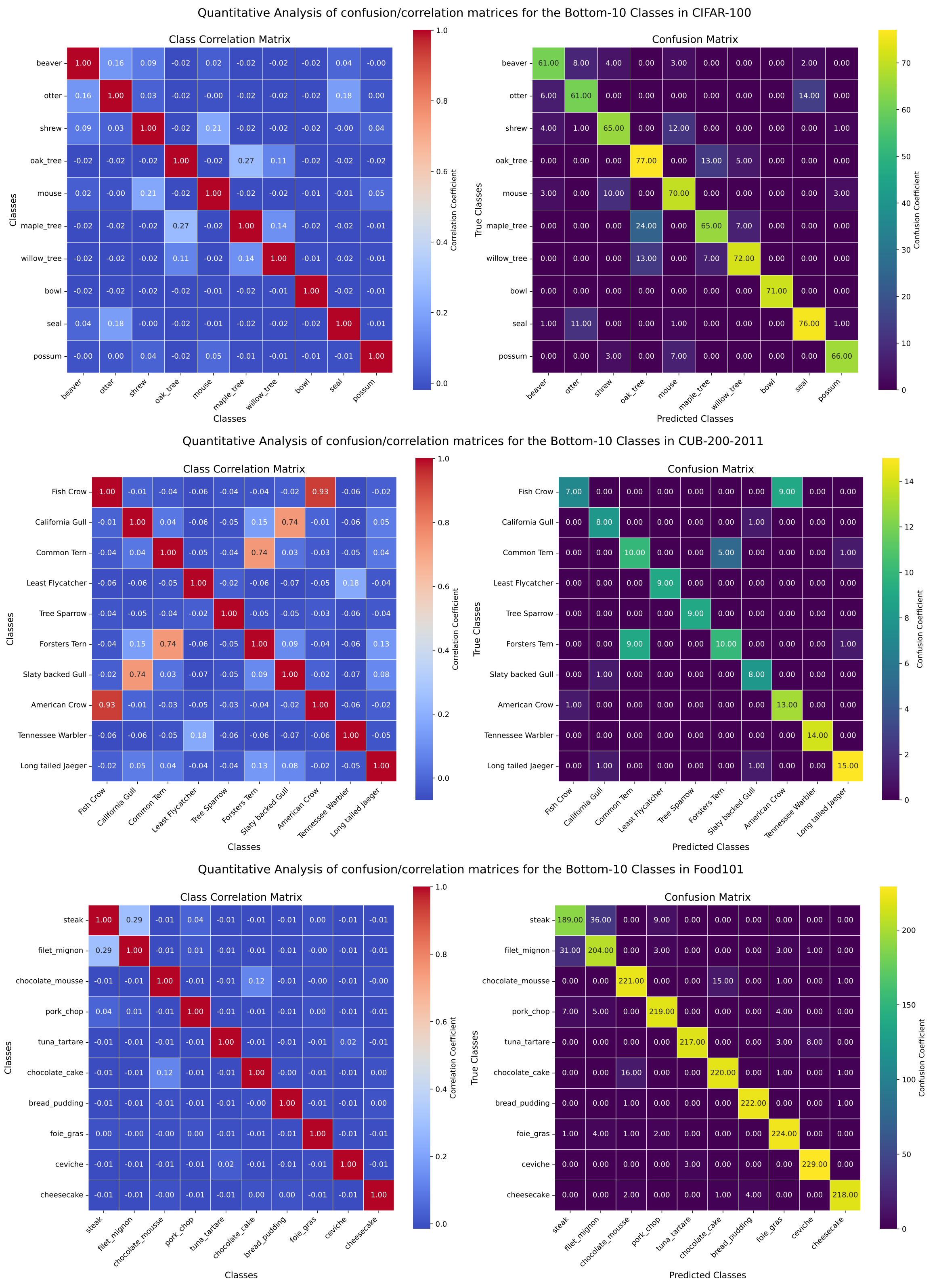


Figure 1. Qualitative analysis of *Pearson's Correlation* disambiguation metric for the 10 most underperforming classes for CIFAR-100, CUB, and Food101. ESCHER's heuristic does not require any human annotations, yet accurately approximates inter-class confusion. However, this heuristic is often over-sensitive to minute errors and is symmetric, leading to slightly suboptimal disambiguation.

Qualitative analysis of ‘Top-k pseudo-confusion’ as a disambiguation heuristic.

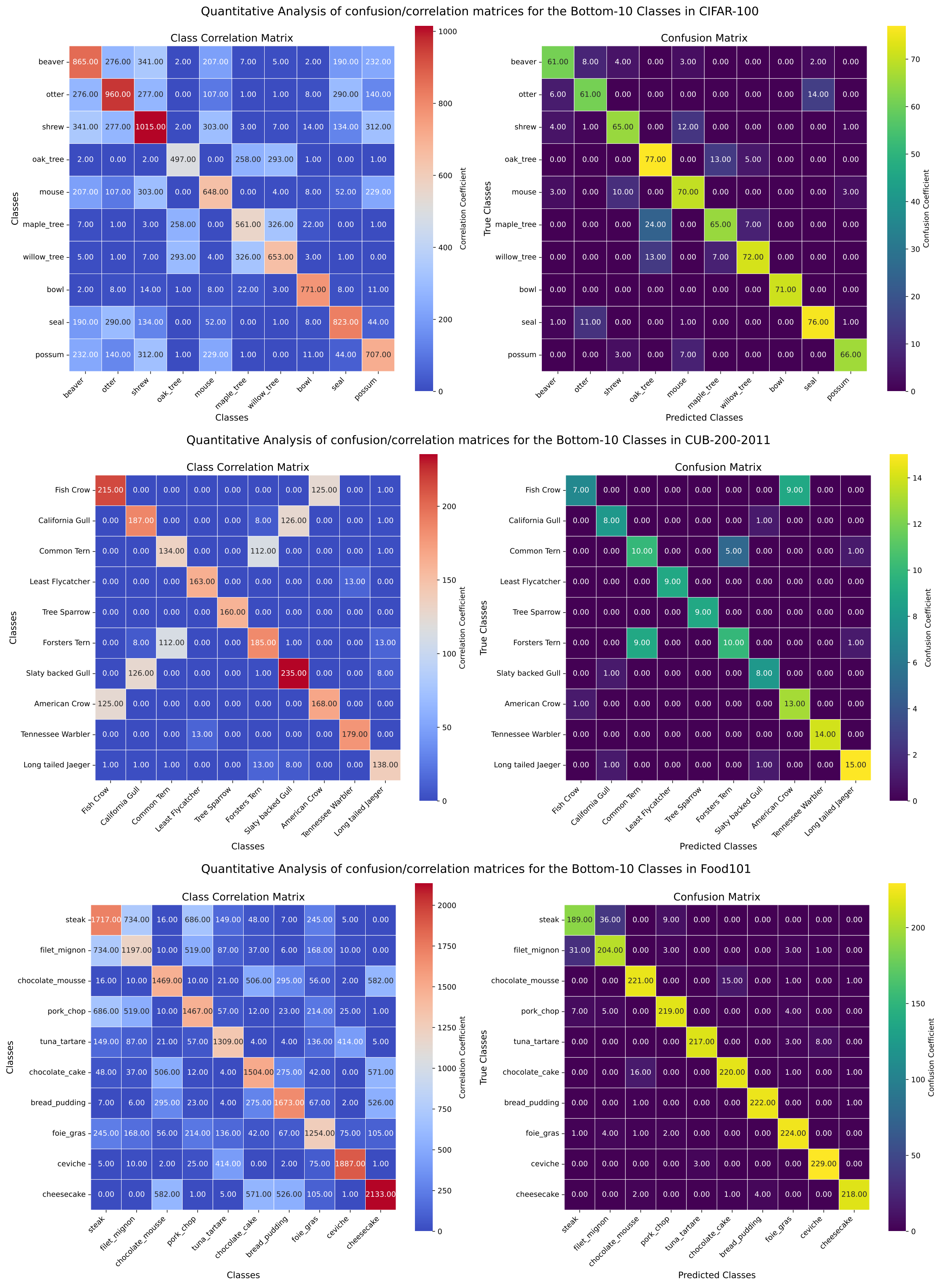
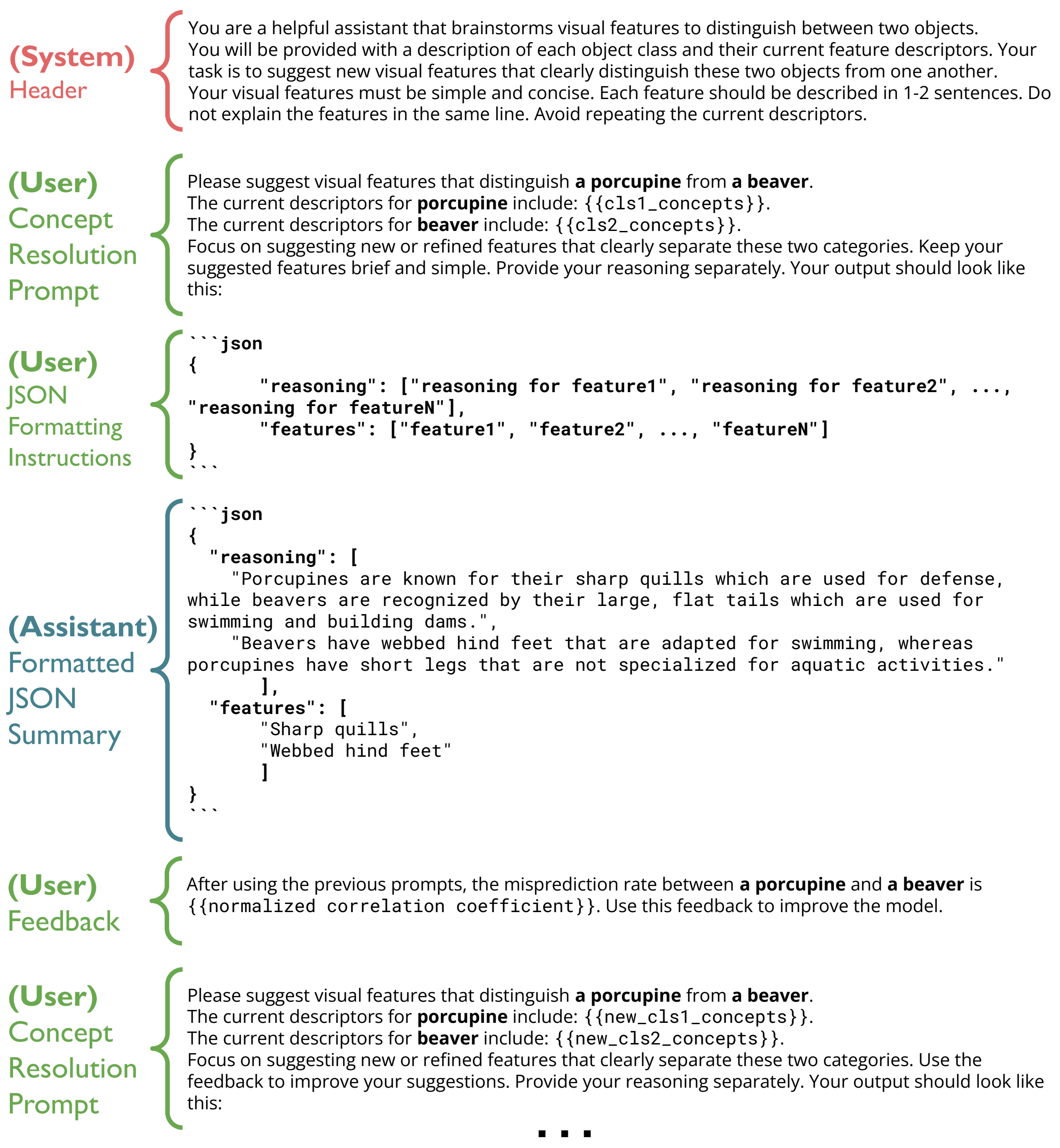


Figure 2. Qualitative analysis of the *Top-k pseudo-confusion* disambiguation heuristic calculated for the 10 most underperforming classes for CIFAR-100, CUB, and Food101. After computing the top-k class scores $\text{topk}(\hat{\mathbf{y}})[:, : k]$, we compute the confusion matrix by incrementing the (i, j) value if y_i and y_j class occur in the top-k entries for an image. Top-k pseudo-confusion tends to be sensitive to minute errors which leads to slight sub-optimality.


(System) Header

You are a helpful assistant that brainstorms visual features to distinguish between two objects. You will be provided with a description of each object class and their current feature descriptors. Your task is to suggest new visual features that clearly distinguish these two objects from one another. Your visual features must be simple and concise. Each feature should be described in 1-2 sentences. Do not explain the features in the same line. Avoid repeating the current descriptors.

(User) Concept Resolution Prompt

Please suggest visual features that distinguish **a porcupine** from **a beaver**.
The current descriptors for **porcupine** include: `{{cls1_concepts}}`.
The current descriptors for **beaver** include: `{{cls2_concepts}}`.
Focus on suggesting new or refined features that clearly separate these two categories. Keep your suggested features brief and simple. Provide your reasoning separately. Your output should look like this:

(User) JSON Formatting Instructions

~~~
```json
{
       "reasoning": ["reasoning for feature1", "reasoning for feature2", ...,
"reasoning for featureN"],
       "features": ["feature1", "feature2", ..., "featureN"]
}
```
~~~

(Assistant) Formatted JSON Summary

~~~
```json
{
  "reasoning": [
    "Porcupines are known for their sharp quills which are used for defense,
while beavers are recognized by their large, flat tails which are used for
swimming and building dams.",
    "Beavers have webbed hind feet that are adapted for swimming, whereas
porcupines have short legs that are not specialized for aquatic activities."
       ],
  "features": [
       "Sharp quills",
       "Webbed hind feet"
       ]
}
```
~~~

(User) Feedback

After using the previous prompts, the misprediction rate between **a porcupine** and **a beaver** is `{{normalized correlation coefficient}}`. Use this feedback to improve the model.

(User) Concept Resolution Prompt

Please suggest visual features that distinguish **a porcupine** from **a beaver**.
The current descriptors for **porcupine** include: `{{new_cls1_concepts}}`.
The current descriptors for **beaver** include: `{{new_cls2_concepts}}`.
Focus on suggesting new or refined features that clearly separate these two categories. Use the feedback to improve your suggestions. Provide your reasoning separately. Your output should look like this:

...

Figure 3. ESCHER's prompt with history conditioning. We find that history conditioning is beneficial when the same disambiguation pair is repeatedly identified.

tioning enabled, the size of each call to the LLM grows linearly with the number of iterations. Assuming each query is approximately 500 tokens, and each additional history entry adds 100 tokens, we can estimate the total token usage for 50 iterations to be around 128,000 tokens if we only consider just a single query per iteration and around 12.8M tokens for each experiment (approximately an hour).

Per iteration metrics. Typically, a single ESCHER iteration can take 2 to 15 minutes for all CMBs on a single GPU. ESCHER largely spends this time optimizing the CBM adapter. Another expensive operation – generating disambiguation concepts with the LLM –is parallelizable and finishes in 1-2 minutes. The number of concepts per class depends on the CBM and the dataset complexity. Each

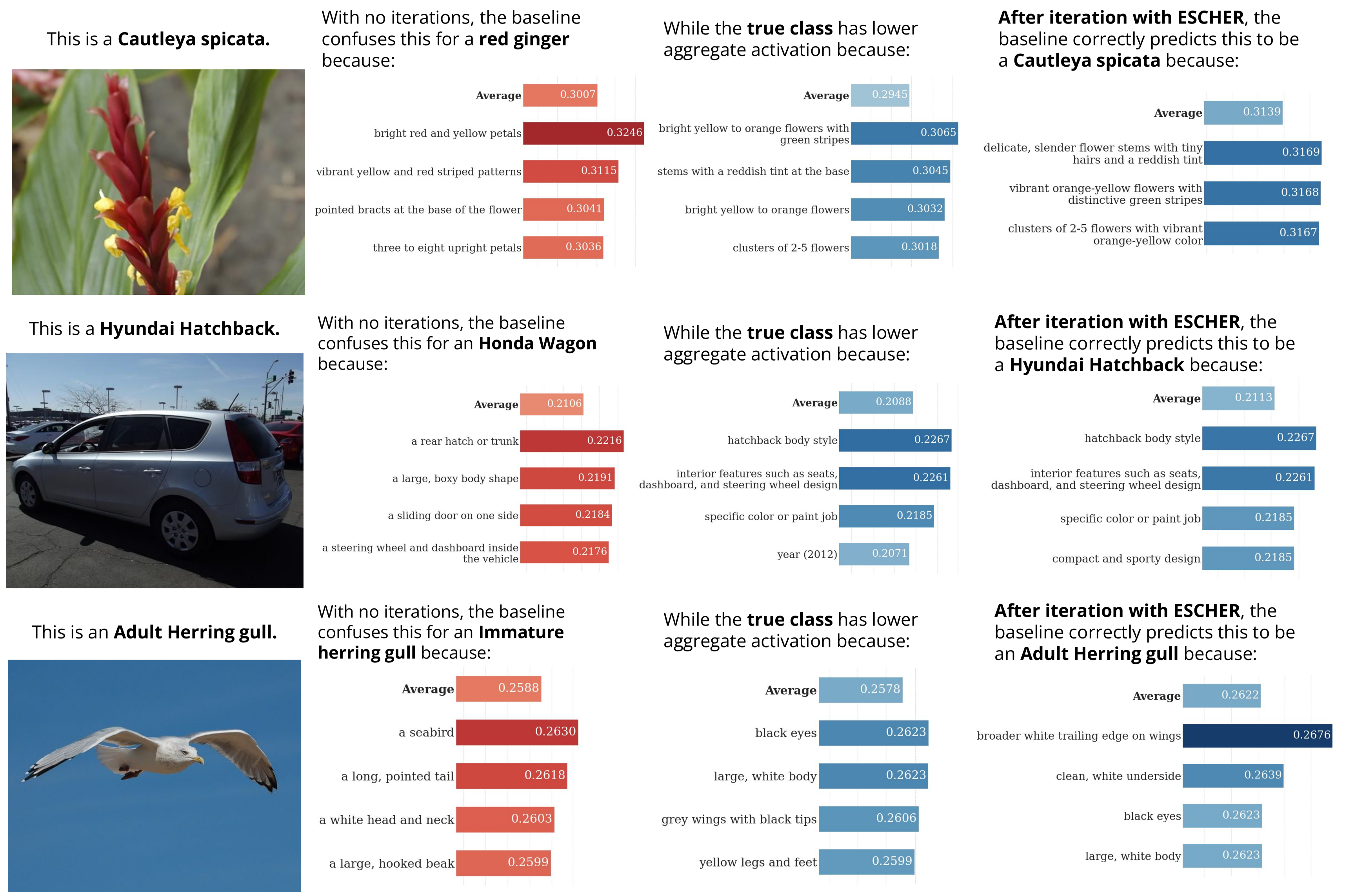


Figure 4. Qualitative analysis of the concept iteration progress on Oxford Flowers, Stanford Cars, and NABirds respectively for CbD+ESCHER. We show samples of mispredicted classes for the zero-shot baseline (CbD) along with the confidence scores. With the initial concepts, the fine-grained class is misidentified. However, after iterating with ESCHER, the baseline model is able to predict the correct class.

disambiguation query typically adds 2–5 concepts. We were able to comfortably fit an adapter with up to 5000 attributes (maximum we tested) on an NVIDIA RTX 2080 Ti GPU.

Choosing disambiguation heuristics. In addition to the observations presented in §1.4, Figure 1, and Figure 2 which highlight various advantages and shortcomings of each disambiguation heuristic, we find that, generally, grid-searching for the best disambiguation heuristic and hyper-parameters is surprisingly practical, when running for low number of iterations with a local language model (around 10 iterations).

Confusion matrix runtime. The confusion score calculation is perfectly parallelizable, and is vectorized. Our naive numpy/scipy implementation takes roughly 6 seconds for our largest dataset (400 classes). Moreover, we can exploit properties of specific heuristics to derive further speedups (e.g. symmetric confusion matrix for *Pearson's correlation*).

1.8. ESCHER's convergence properties

Intuitively, ESCHER employs the concept library to simultaneously bootstrap the CBM and LLM: A CBM with a specialized concept library produces more fine-grained class disambiguation feedback, prompting the LLM to uncover even finer concepts, which leads to an even more specialized library for the next iteration. Like other library learning algorithms [1, 2, 4], ideally – while the LLM disambiguates concepts well and the CBM remains sensitive to them – this self-reinforcing loop continues until no more relevant concepts emerge. Empirical results suggest that ESCHER discovers relevant concepts for many datasets. A deeper exploration of library learning's optimization properties are presented in [1].

1.9. Bayesian formulation for CBMs

We can abstractly define the fitness of an adapter as the likelihood $p_{\mathcal{C}}(\mathcal{D}|w_{\mathcal{Y}})$ of the adapter generating the dataset $\mathcal{D}$. Not every concept will contribute to the final class assignment. Hence, we regularize the adapter by imposing a prior probability distribution $p_{\mathcal{C}}(w_{\mathcal{Y}})$ that is enforced by sampling the concepts with an LLM, leveraging the prior knowledge of the LLM to filter out irrelevant concepts. Now, the problem of training a concept-bottleneck model can be expressed as a maximum a posteriori (MAP) estimation problem:

$$\begin{aligned} w^{\star}_{\mathcal{Y}} &= \arg\max_{w_{\mathcal{Y}}} p_{\mathcal{C}}(w_{\mathcal{Y}}|\mathcal{D}) \qquad (1) \\ &= \arg\max_{w_{\mathcal{Y}}} \underbrace{p_{\mathcal{C}}(\mathcal{D}|w_{\mathcal{Y}})}_{\text{optimize}} \cdot \underbrace{p_{\mathcal{C}}(w_{\mathcal{Y}})}_{\text{regularizer}} \end{aligned}$$

1.10. History Conditioning

Due to different training objectives and operational modalities, the concepts proposed by the LLM and the VLM's interpretation of concepts are bound to be misaligned frequently. In such cases, ESCHER often needs to disambiguate the same pair of classes multiple times. Also, it is possible for each class disambiguation query to cause collisions with other classes in later iterations. This motivates the need to keep track of past LLM proposals to each concept set as well as the VLM's response to each of the changes. ESCHER implements this using the INITIALIZEHISTORY and the UPDATEHISTORY functions.

INITIALIZEHISTORY. This function takes as input the number of classes $\mathcal{Y}$ and the number of iterations T and constructs a data structure to hold the list of descriptors for a pair of classes (i, j) at iteration t as well as the updated class-confusion heuristic generated by the VLM at iteration $t + 1$.

UPDATEHISTORY. This function plays two roles. First, it stores the list of new descriptors for the i^{th} and j^{th} class for the $t+1^{\text{th}}$ iteration after the class confusion resolution. Second, in the subsequent iteration, the updated class confusion score for the (i, j) pair is stored to measure the effectiveness of the concepts proposed in the class confusion resolution step.

1.11. Additional backbone ablation with ViT-B/32.

We report observations after evolving concepts with ESCHER using LM4CV and CbD with a new backbone LLM (4bit quantized `Llama-3.3-70B-Instruct`) and VLM (ViT-B/32). The results are presented in Table 5. We find that, while the overall accuracy is lower due to the weaker backbone models, iterating with ESCHER leads to better performance than relying on a fixed set of concepts.

ViT-B/32	CLIP	LM4CV	LM4CV +ESCHER	CbD	CbD +ESCHER
CIFAR-100	59.60	78.50	**78.71**	62.50	**64.10**
CUB-200-2011	52.83	63.62	**67.64**	54.17	**55.67**
Food101	77.23	87.95	**88.09**	79.99	**80.36**
NABirds	39.69	57.99	**59.24**	**41.48**	**41.48**
Stanford Cars	59.13	75.54	**75.56**	57.40	**60.62**

Table 5. Top-1 accuracy for evolving ESCHER with a weaker LLM (Llama-3.3-70B-4bit) and visual critic (ViT-B/32). ESCHER consistently improves the performance of LM4CV and CbD across datasets.

References

[1] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pages 835–850, 2021. 7

[2] Gabriel Grand, Lionel Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *The Twelfth International Conference on Learning Representations*, 2024. 7

[3] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. 2

[4] Catherine Wong, Kevin Ellis, Joshua B. Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. In *International Conference on Machine Learning*, 2021. 7