PEEB: Part-based Image Classifiers with an Explainable and Editable Language Bottleneck

Anonymous ACL submission

Abstract

CLIP-based classifiers rely on the prompt containing a `{class name}` that is known to the text encoder. That is, CLIP performs poorly on new classes or the classes whose names rarely appear on the Internet (e.g., scientific names of birds). For fine-grained classification, we propose to (1) express the class name into a set of pre-defined text descriptors that describe the visual parts of that class; and (2) match the embeddings of the detected parts to their textual descriptors in each class to compute a logit score for classification. In a zero-shot setting where the class names are unknown, PEEB outperforms CLIP by a large margin (~10× in accuracy). Compared to part-based classifiers, PEEB is not only the state-of-the-art on the supervised-learning setting (88.80% accuracy) but also the first to enable users to *edit* the class definitions to form a new classifier without retraining. Compared to concept bottleneck models, PEEB is also the state-of-the-art in both zero-shot and supervised learning settings.

1 Introduction

Fine-grained bird classification (Wah et al., 2011; Van Horn et al., 2015) is a long-standing challenge in computer vision. Yet, state-of-the-art bird classifiers often have one or more of the following three limitations. First, many models both CNN-based (Krause et al., 2016) and ViT-based (He et al., 2022) are inherently *black-box*. That is, they have no built-in mechanisms that explain to users how a decision is made, e.g., which bird traits make a model think a given bird is `Indigo Bunting`? Second, many bird classifiers claim to be explainable (Chen et al., 2019; Donnelly et al., 2022) by comparing the input image with a set of learned, part-based prototypes or natural language concepts. Yet, such prototypes are feature vectors and therefore not editable by users. Third, textual concept-based models operate at the image level and it is unknown what image details match a given descriptor (Menon and Vondrick, 2022; Yang et al., 2023). Fourth, most classifiers require either training-set images in a supervised-learning setting or demonstration images in a zero-shot setting (Xian et al., 2018; Zhu et al., 2018). This requirement is impractical when building a classifier for a novel class whose photos do not yet exist in the database.

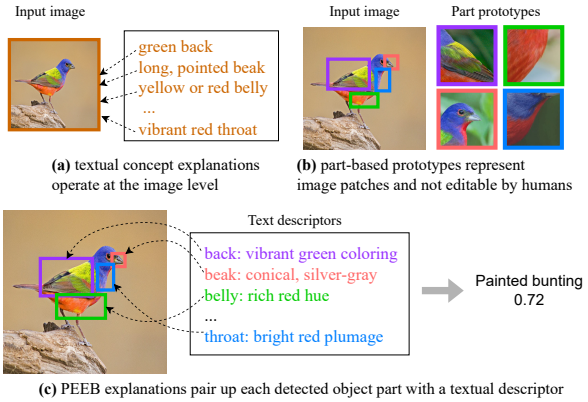


Figure 1: Existing explanations are either (a) textual but at the image level; or (b) part-level but not textual. Combining the best of both worlds, PEEB (c) first matches each detected object part to a text descriptor, then uses the part-level matching scores to classify the image.

To address the above problems, we propose PEEB, a bird image classifier that is both explainable and editable via natural language. PEEB classifies images based only on the textual descriptor of bird *parts* provided by humans (no images needed) and grounds the descriptors to the visual bird parts for more fine-grained explanations (Fig. 1). While PEEB leverages CLIP's encoders (Radford et al., 2021), it uses no class names (e.g., `Indigo Bunting`) in the prompt. In contrast, CLIP-based classifiers (Radford et al., 2021) and its extensions (Pratt et al., 2022; Menon and Vondrick, 2022) rely so heavily on the *known* class names in the prompt that their accuracy drops significantly when the names are removed or replaced by uncommon alters (Sec. 5.1).

Using GPT-4, we construct a *textual* descriptor

(OpenAI, 2023) to describe each bird part of every species (see Appendix C). PEEB first generates 12 *visual* part embeddings, then localizes the 12 bird parts in the input image based on these embeddings (Fig. 2). The unnormalized distance (logits) between the input image and every class would be the sum of the 12 dot products between the paired visual and textual part embeddings (Fig. 3). Besides being editable by humans, PEEB outperforms concept-based explainable classifiers across different zero-shot settings and prototypical part networks in a supervised fashion.

To our knowledge, all existing public bird-image datasets (listed in Table A4) are limited in size (less than 100K images per dataset) and in the number of classes (less than 1,500 species per dataset), impeding large-scale, vision-language, contrastive learning research. Therefore, for our pre-training, we build Bird-11K, an unprecedentedly large bird-image dataset of ~290K images and ~11K species, i.e., basically *all* bird species on Earth (Sec. 3). Bird-11K is constructed from 7 existing bird datasets and 55K new images that we collect from the Macaulay Library.

Our main findings are:[1]

1. CLIP-based classifiers depend mostly on class names in the prompt: the CUB accuracy of M&V classifier (Menon and Vondrick, 2022) drops substantially from 53.78% to 5.89% and 5.95% after class names are removed or replaced with scientific names in the prompts, respectively (Sec. 5.1).

2. Our part-level pre-trained PEEB outperforms CLIP-based classifiers by +8 to +29 points in bird classification across CUB, NABirds, and iNaturalist datasets (Sec. 5.2).

3. PEEB allows defining new classes in text during the test time without re-training the models (Fig. 2). Besides interpretability and editability, PEEB outperforms other *text concept-based* methods in the generalized-zero-shot setting (Sec. 5.3).

4. On a zero-shot setting by Elhoseiny et al. (2017), PEEB also outperforms other existing state-of-the-art *text concept-based* methods (Sec. 5.4), especially on "hard" splits, showing strong generalization capabilities.

5. When compared with explainable classifiers in supervised learning on CUB, PEEB scores an 88.80% accuracy, which is competitive to the best CUB classifiers trained using supervised learning (81–87% accuracy) in the literature (Sec. 5.5) that are often not editable.

[1]Code and dataset are released on https://anonymous.4open.science/r/peeb-Bird-11K/README.md.

2 Related Work

Standard CNNs and Transformers It is common to build bird classifiers based on standard CNNs such as ResNets (He et al., 2016) or ViTs (He et al., 2022). Although high-performing, these models do not admit an inherent explanation interface (Gunning et al., 2021) and therefore rely on post-hoc interpretability methods, which tend to offer inaccurate and unstable, after-the-fact explanations (Rudin, 2019; Bansal et al., 2020). In our work, the textual part descriptors form a natural-language bottleneck interface that enables users to observe and edit the bird attributes that contribute to each final prediction. That is, users can re-program the classifier without having to re-train any network (see Fig. 2).

Prototypical Part Networks There are bird classifiers, such as ProtoPNet (Chen et al., 2019), designed with an explainability objective to learn prototypes i.e., learnable concepts (Nauta et al., 2021; Donnelly et al., 2022; Nguyen et al., 2022; Nauta et al., 2022; Kim et al., 2022; Wang et al., 2021). Yet, because such prototypes are real-valued vectors, it is unknown how much users could interpret and use them in a downstream task. In contrast, PEEB is a bird classifier that relies on part-based concepts but allows users to define the textual descriptors of the birds of interest, while prior prototype-based classifiers require complete re-training if any prototype needs modifications.

Textual-based Concept Bottlenecks Recent vision-language models (VLMs) are often considered interpretable due to their reliance on natural language concepts. However, several works (Samuel et al., 2021; Yuksekgonul et al., 2023; Esfandiarpoor and Bach, 2023), which rely on specific conditions such as differences in class-wise captions (Esfandiarpoor and Bach, 2023) or learned concept weights (Yang et al., 2023; Panousis et al., 2023; Oikarinen et al., 2023), are constrained and lack the capability for generalization. Another line of work with diverse approaches to process textual concepts for training typically lacks immediate editability and often requires re-training or additional

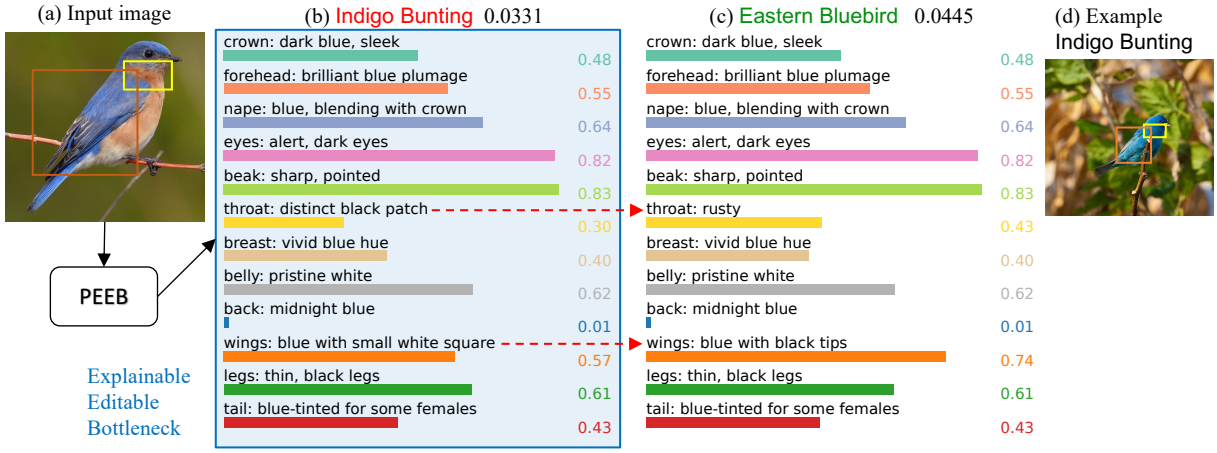


Figure 2: Given an input image (a) from an unseen class of `Eastern Bluebird`, PEEB misclassifies it into `Indigo Bunting` (b), a visually similar blue bird in CUB-200 (d). To add a new class for `Eastern Bluebird` to the list of 200 classes that PEEB considers when classifying, we clone the 12 textual descriptors of `Indigo Bunting` (b) and **edit** (- -▸) the descriptor of `throat` and `wings` (c) to reflect their identification features described on `AllAboutBirds.org` *("Male Eastern Bluebirds are vivid, deep blue above and rusty or brick-red on the throat and breast")*. After the edit, PEEB correctly predicts the input image into `Eastern Bluebird` (0.0445) out of 201 classes (c).

steps for new concepts (Ji et al., 2018; Zhu et al., 2018; Paz-Argaman et al., 2020; Chen et al., 2020; Kousha and Brubaker, 2021; Rao et al., 2023; Han et al., 2023). CLIP-based classifiers rely heavily on having correct class names rather than *descriptors* in text prompts (Pratt et al., 2022; Menon and Vondrick, 2022) and thus are neither explainable nor editable to humans. Unlike CLIP-based models, PEEB reveals what image details are being used for classification by matching descriptors and visual bird *part* (e.g. `beak` in Fig. 3).

Attribute-based Classifiers The Attribute Label Embedding (ALE) approach (Akata et al., 2015) employs a fixed set of attributes and trains an attribute-to-label weight matrix for zero-shot classification. Several studies (Xu et al., 2020; Hanouti and Le Borgne, 2023) highlight its effectiveness on datasets like CUB, SUN (Xiao et al., 2010), and AWA (Xian et al., 2019). However, ALE has a limitation in terms of flexibility i.e. expanding label numbers necessitates overhauling the attribute set, weight matrix, and model re-training. This rigidity contrasts with our target for a scalable and adaptable framework, leading us to explore alternatives beyond ALE despite its success.

3 Bird-11K Dataset

3.1 Dataset construction

We combine bird images from 7 distinct datasets with ~55K images (10,534 classes) collected from Cornell's Macaulay Library, to form a unified **Bird-11K** dataset [2] (Table A4) for large-scale pre-training. To the best of our knowledge, Bird-11K, comprising 440,934 images spanning 11,183 classes, is the first bird dataset that encompasses almost all species on Earth. Since PEEB learns to match visual parts to textual descriptors, it requires that bird images be distinctly visible and sufficiently large for accurate part localization and matching (See appendix E.3 for ablation study). However, small and "hard-to-see" bird images in Bird-11K make the dataset noisy and the training complex. Thus, we employ OWL-ViT$_{\text{large}}$ (Minderer et al., 2022) to detect bird objects in all images using the query *"bird"* and filter out images with the detected bird's bounding boxes smaller than 100×100 pixels.

To circumvent class ambiguity, we retain only the child species and exclude all parent classes. For instance, it is infeasible to systematically map the parent class *Cardinal* to child classes such as *Yellow Cardinal* or *Northern Cardinal* so we keep only the child classes for more diverse training. Following these filtration steps, the refined Bird-11K dataset retains 294,528 images across 10,811 classes (Table A4). For each species in Bird-11K, we generate a set of part-based descriptors using GPT-4. Details of the descriptor generation are provided in Appendix C.

[2] We do not redistribute the published datasets but release a script to reconstruct Bird-11K.

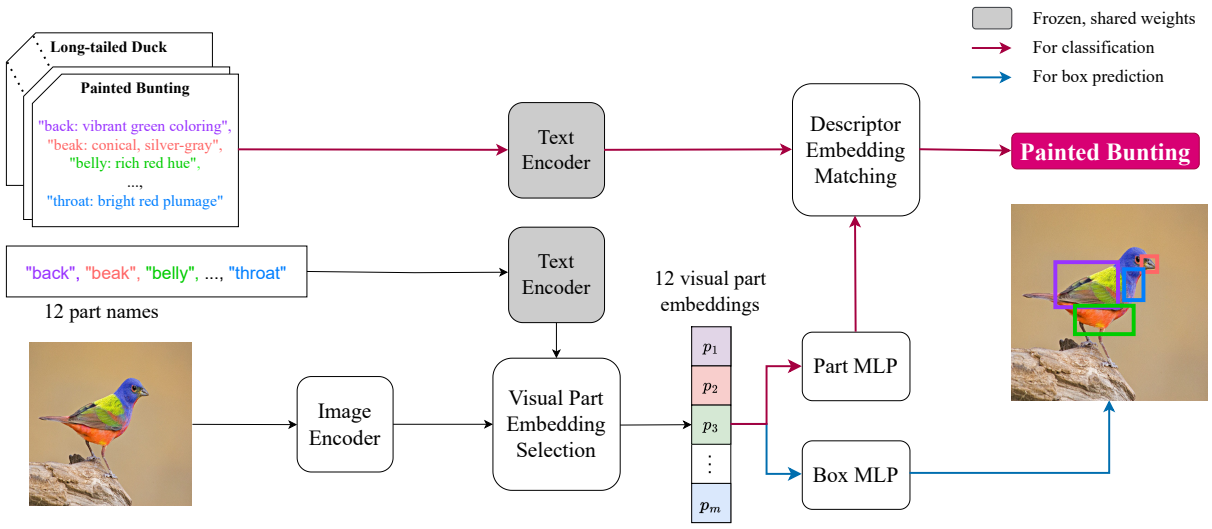


Figure 3: During inference, 12 visual part embeddings are selected based on the highest cosine similarity with encoded part names. These visual part embeddings are processed through the Box MLP for box prediction (⟶). Simultaneously, the same embeddings are forwarded to the Part MLP, and the output is then matched with encoded descriptors to make final predictions (⟶). A detailed diagram is provided in Fig. A1.

3.2 Dataset splits for contrastive pre-training

Two distinct zero-shot settings emerged in the recent literature. The first setting is conventional zero-shot (**ZSL**), which ensures a model is not exposed to any test classes during training. Adhere to the conventional setup, we execute different exclusion strategies on Bird-11K to make sure the test classes are never exposed to the pre-trained models. For evaluation, we employ the traditional CUB split proposed by Akata et al. (2015) and two harder splits: Super-Category-Shared/Exclusive (SCS/SCE) by Elhoseiny et al. (2017). For example, in ZSL on CUB, we exclude all CUB classes in Bird-11K for pre-training and fine-tune only on the corresponding training set given by a ZSL split.

The second setting is generalized zero-shot (**GZSL**), where models like CLIP are trained on large-scale datasets that may inadvertently include a subset of the testing classes or images. In the GZSL test, we exclude all test sets from CUB, NABirds, and iNaturalist, and directly evaluate the pre-trained model without further fine-tuning.

4 Method

4.1 Backbone: OWL-ViT bird-part detector

OWL-ViT (Minderer et al., 2022) is an open-vocabulary object detector that detects objects in an image, given text queries, even if those objects are unseen during training. OWL-ViT consists of four components: a standard Vision Transformer *image encoder*, and an architecturally identical *text encoder*, a *box regression head*, and a *box classification head*. While the box regression head is a three-layer Multilayer Perceptron (MLP) followed by a GELU activation (Hendrycks and Gimpel, 2016) for the first two linear layers, the box classification head is simply a linear layer to project the visual embeddings to the same dimensional space with text embeddings.

4.2 Part-based, explainable, and editable bird classifier (PEEB)

Architecture The PEEB model is composed of two encoders—an *image encoder* and a *text encoder*—as well as three key components: a *linear projection* block, a *part MLP* (Multi-Layer Perceptron), and a *box MLP*. Part MLP layer is designed for mapping between visual parts and descriptors that can be directly used for classification (⟶ in Fig. 3). This design allows PEEB to perform arbitrary ways of classification. The remaining components of the model are adopted from the OWL-ViT framework. Details of the components are provided in Appendix A.

Model Inference For a given image, we first employ the 12 part names as textual queries and select the visual part embeddings based on the cosine similarity. These selected visual part embeddings are then simultaneously fed into both the part MLP and the box MLP. The box MLP will predict the bounding box coordinates of each part. The output

from the part MLP is matched against the encoded text descriptors, with predictions being made based on their aggregated cosine similarity. Specifically, we compute the dot product between the selected visual part embeddings and text embeddings of part descriptors, summing up the resulting twelve scores to form a logit for each class. Predictions are then obtained by applying the $\arg\max$ to these logits. An overview diagram of PEEB's inference process can be found in Fig. 3.

4.3 Training strategy

We empirically find that solely training the part MLP layer does not achieve the desired classification accuracy, prompting us to update the image encoder. However, re-training this encoder impacts the linear projection and box MLP layers. As a result, we have to train all components together. All the components are initialized from the corresponding components in OWL-ViT, except for our proposed part MLP. Our training strategy has two phases: two-stage **pre-training** on the large-scale Bird-11K dataset and **fine-tuning** on downstream tasks. We provide full hyperparameter details of PEEB in appendix A.8.

Objectives There are three objectives to train PEEB: (a) Train the part MLP layer contrastively to maximize the similarity between related part-descriptor pairs while minimizing the unrelated pairs using *symmetric cross-entropy (CE) loss* (Radford et al., 2021); (b) Train the linear projection layer to mimic OWL-ViT's behaviors (i.e. the similarity matrix) for part selection with symmetric CE loss; and (c) Train the box MLP layer for bounding box regression with DETR losses (Zheng et al., 2020) i.e. a linear combination of ℓ_1 corner-to-corner distance loss and GIoU loss (Rezatofighi et al., 2019).

Challenges One of the problems emerges when we jointly train all components together: the model learns at a significantly slow pace since PEEB needs to learn to optimize two symmetric cross-entropy losses while maintaining the high-quality predicted boxes. To address this problem, we split the pre-training phase into two stages: (1) train the image encoder and part MLP layer with the first objective; then (2) train the linear projection and box MLP layers with the second and third objectives to accordingly adjust their weights to the changes in the image encoder. Notably, the text encoder is always frozen because it was designed for open vocabulary, so its generalizability to unseen texts (i.e., descriptors of an unseen bird) should be preserved.

4.3.1 Pre-training on Bird-11K dataset

Stage 1, contrastive learning with teacher model: The *image encoder* and *part MLP* layer are jointly trained using the symmetric CE loss, which is particularly suitable for PEEB as it learns the mapping between visual parts and descriptors. In this stage, we follow the teacher model – OWL-ViT to select part embeddings to ensure the selected part embeddings are semantically meaningful, e.g., can be used for box prediction (Fig. A2).

Stage 2, eliminate teacher dependency:

As the image encoder is modified in stage 1, the linear project and the box MLP needs to be updated accordingly. Specifically, we consider the teacher model OWL-ViT as ground-truths and train the linear projection layer using symmetric CE loss (Fig. A3, 1a–c, 2a–c). For box MLP, given the absence of human-annotated boxes for individual parts, we obtain pseudo labels sourced from OWL-ViT$_{\mathsf{large}}$ as ground-truths for the training with DETR losses (Fig. A3, 2d). In this training step, the image encoder is frozen while the part MLP layer is not involved. After two-step training, PEEB can perform zero-shot classification while providing the mappings between visual parts and descriptors as faithful explanations.

4.3.2 Fine-tuning on target datasets

We can further fine-tune the pre-trained model on downstream tasks, e.g., CUB, NABirds and iNaturalist to compare with other baselines (e.g. prototype-based approaches). In this phase, to adapt to the downstream tasks, all components except the text encoder are trained jointly and the loss function for part MLP is changed from symmetric CE to CE while other losses are kept intact.

5 Experiments & Results

We conduct systematic experiments to compare the generalization ability of PEEB with *explainable* methods on two zero-shot settings: GZSL (Sec. 5.1 – 5.3) and ZSL (Sec. 5.4). Notably, our method demonstrates significantly superior performance in GZSL and generalizes better in ZSL setting.

Moreover, following Donnelly et al., 2022, we finetune and evaluate PEEB on downstream tasks to measure the transferability compared to other part- and text concept-based methods (Sec. 5.5).

Results show that PEEB achieves decent performance while providing more faithful explanations (Figs. 4 and A9 to A11). According to the evaluation for visual part localization (Appendix E.7) and qualitative analysis (Appendix G), we find that the box prediction performance of PEEB is comparable with OWL-ViT$_{\text{base32}}$.

5.1 CLIP-based classifiers depend mostly on class names (not part descriptors)

M&V shows that incorporating descriptors generated by GPT-3 (Brown et al., 2020) to class names increases CLIP accuracy on downstream tasks. Yet, it remains unknown how useful the descriptors are compared to the randomized descriptors for M&V's method.

Experiment To address the concern, we conduct two systematic studies: (1) randomly swap a set of part descriptors among classes in CUB, NABirds, and iNaturalist datasets and measure the contribution of descriptors by the difference in model performance on CUB test set (i.e., 200-way classification) using original and random descriptions. (2) Replace the common names with scientific names in CUB, NABirds, and iNaturalist and measure the difference in model performance. The first experiment aims to study the contributions of descriptions to the CLIP-based method, and the second experiment tries to understand the dependencies of the class names.

Results When random descriptors are used, M&V 's accuracy drops marginally by -0.9 points, while PEEB's performance significantly deteriorates, indicating a strong dependence on accurate descriptions. However, when only descriptors are used without class names (Table 1) or class names are replaced by scientific names (Table 2), M&V 's accuracy reduces drastically to nearly random chance, underscoring the reliance of CLIP-based classifiers on class names.

5.2 Part-based pre-trained PEEB outperforms CLIP-based classifiers

The dependence on class names suggests CLIP is potentially exposed to class names during training. Thus, we compare PEEB with the CLIP-based classifiers in the GZSL setting for a fair comparison.

Experiments We conduct two-stage pre-training for PEEB on Bird-11K where CUB, NABirds, and iNaturalist test sets are excluded for evaluation. Note that the contrastive pre-training is on part level, that is, PEEB does not have direct access to the class labels.

Results PEEB outperforms the baselines across all three datasets, achieving improvements of (+10 to +12 points), (+28 to +29 points) and (+8 to +9 points) on CUB, NABirds and iNaturalist, respectively (Table 2).

Table 1: We evaluate model accuracy on the CUB test set using both original and incorrect descriptors. The results highlight M&V's minimal dependency on descriptors in contrast to our method's significant performance drop with shuffled descriptors.

	CLIP	M&V		PEEB (**ours**)
Using class names	✓	✓	✗	✗
Original Descriptors	52.02	53.78	5.89	**64.33**
Incorrect Descriptors	n/a	52.88	0.59	0.88

Table 2: In GZSL setting, our method's top-1 accuracy is +8 to +29 points higher than the two baselines. When using novel class names (or scientific names which are less common), our method is around 10× better than the others.

Methods	CUB	CUB$_{sci}$	NABirds	NABirds$_{sci}$	iNaturalist	iNaturalist$_{sci}$
CLIP (2021)	52.02	5.95	39.35	4.73	16.36	2.03
M&V (2022)	53.78	7.66	41.01	6.27	17.57	2.87
PEEB (ours)	**64.33**		**69.03**		**25.74**	

5.3 PEEB is superior to text concept-based classifiers on GZSL setting

The advancements in Large Language Models (LLMs) led to the emergence of **text concept-based** approaches (Yuksekgonul et al., 2023; Yan et al., 2023; Esfandiarpoor and Bach, 2023; Panousis et al., 2023) aiming to provide a more flexible and explainable classifier. We compare PEEB in the GZSL setup on CUB test set with text concept-based methods.

Experiment We conduct an experiment with the same setting as in Sec. 5.2. While PEEB does not use specific concepts, the descriptors it utilizes (described in Appendix C) function as open-vocabulary concepts. Most text concept-based methods in GZSL require pre-defined concept sets, limiting their applicability to similar datasets like NABirds or iNaturalist. Hence, our evaluation is focused solely on the CUB dataset.

Results As indicated in Table 3, our PEEB model exhibits superior GZSL performance, outperforming recent text concept-based approaches by +3

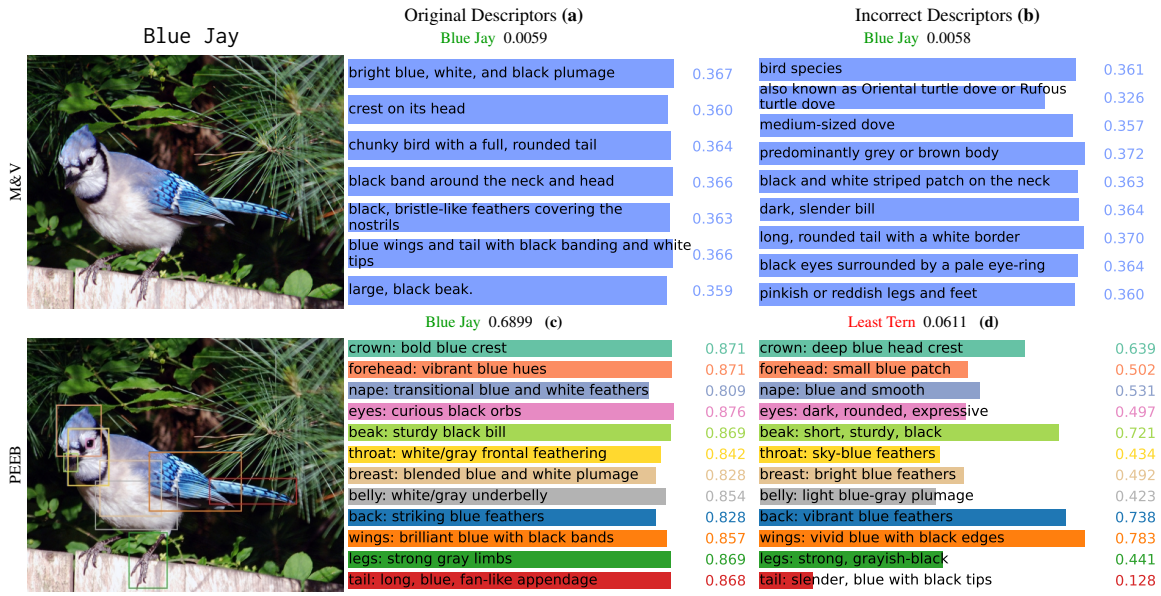


Figure 4: Given the correct descriptors, M&V correctly classifies the input image into `Blue Jay` (a). Yet, interestingly, when randomly swapping the descriptors of this class with those of other classes, M&V's top-1 prediction remains unchanged (b), suggesting that the class names (hidden) in the prompt have the most influence over the prediction (not the descriptors). In contrast, PEEB changes its top-1 prediction from `Blue Jay` (c) to `Least Tern` when the descriptors are randomized (d).

to +10 points. Unlike the traditional approaches, which primarily rely on predefined concept pools, PEEB utilizes domain-specific, natural language descriptors. This distinctive approach enables PEEB to adapt more flexibly to various datasets such as NABirds and iNaturalist. In contrast, concept pool-based methods often necessitate manual concept updates and potential re-training for such extensions.

Table 3: PEEB achieves SOTA performance in the GZSL setup among the **text concept-based** methods. *Note: * One shot learning results. † k-means results with k = 32.*

Method	CUB	Concept-type
FuDD (2023)	54.30	Class-wise Captions
Han et al. (2023)	56.13	Evolving Descriptions
PCBM (2023)	61.00	Pre-defined Concept Pool
Yan et al. (2023)	60.27*	Pre-defined Concept Pool
LaBo (2023)	54.19†	Pre-defined Concept Pool
PEEB (Ours)	**64.33**	Class-wise Descriptors

5.4 PEEB generalizes to traditional ZSL

Some vision-language approaches enable the model to process certain textual-based concepts (Ji et al., 2018; Zhu et al., 2018; Paz-Argaman et al., 2020; Chen et al., 2020; Kousha and Brubaker, 2021; Rao et al., 2023), and therefore, ZSL is used to measure their generalization capability. In this evaluation, we employed the traditional ZSL split on CUB from Akata et al. and the Super-Category-Similar/Exclusive (SCS/SCE) proposed by Elhoseiny et al. on CUB and NABirds. The SCS (easy) and SCE (hard) split are designed based on the species hierarchy to intentionally have two levels of difficulties from the ZSL test.

Experiment Adhere to the conventional ZSL setting; we exclude all the CUB or NABirds classes for pre-training and fine-tune the model following Akata et al. or Elhoseiny et al.'s split. We randomly select ~10% from the training set as the validation set and choose the checkpoints based on the lowest validation loss.

Results PEEB outperforms all baselines across 3 test splits (from CUB and NABirds) by (+4 to +10 points) in terms of harmonic mean, indicating that PEEB is more generalized to not only seen classes (80.78 vs 65.80) but also unseen classes (all other results in Table 4). The easy tests (SCS) guarantee the presence of classes similar to (but distinct from) the ones in the training set. Therefore, all baselines that learn better on the training set tend to have better accuracy in the easy test. Conversely, the hard splits (SCE) ensure the test classes are from different categories of the training classes. This distinction makes the hard test a more accurate metric for assessing a model's generaliza-

tion ability. PEEB excels over all baselines by (+5 to +15 points) (accuracy) for SCE split and +2.64 points (accuracy) compared to CLORE$_{CLIP}$.

Furthermore, we also evaluate the model that is pre-trained without all CUB and NABirds classes and compare them with CLIP and M&V methods on CUB$_{sci}$ and NABirds$_{sci}$. Interestingly, PEEB outperforms both baselines with (+10 to +12) points on CUB and (+1 to +3) points on NABirds (Appendix E.1). This finding further substantiates the generalization capability of our method.

Table 4: PEEB consistently outperforms other vision-language methods under Harmonic mean and especially in the hard split (SCE) by (+5 to +15) points, highlighting its generalization capability on ZSL.

Methods	CUB			NABirds		
	Seen	Unseen	Harmonic	Seen	Unseen	Harmonic
Data split by Akata et al. (2015)						
CLORE$_{CLIP}$ (2022)	65.80	39.10	49.05	n/a		
PEEB (**ours**)	**80.78**	**41.74**	**55.04**			
SCS/SCE splits						
	SCS (Easy)	SCE (Hard)	Harmonic	SCS (Easy)	SCE (Hard)	Harmonic
S^{2}GA-DET (2018)	42.90	10.90	17.38	39.40	9.70	15.56
GRZSL (2018)	44.08	14.46	21.77	36.36	9.04	14.48
ZEST (2020)	**48.57**	15.26	23.22	38.51	10.23	16.17
CANZSL (2020)	45.80	14.30	21.12	38.10	8.90	14.43
DGRZSL (2021)	45.48	14.29	21.75	37.62	8.91	14.41
DPZSL (2023)	45.40	15.50	23.11	**40.80**	8.20	13.66
PEEB (**ours**)	44.66	**20.31**	**27.92**	28.26	**24.34**	**26.15**

5.5 Finetuning pre-trained PEEB on CUB-200 yields a competitive explainable classifier in supervised learning

We compare our model with explainable methods, such as ProtoPNet (Chen et al., 2019), using the conventional pre-training and fine-tuning approach on Bird-11K and the CUB dataset. We focus on evaluating how effectively the pre-trained PEEB transfers to downstream tasks, providing insights into its adaptability and performance.

Experiment We further fine-tune all components of the pre-trained PEEB on the CUB dataset for comparison with other explainable methods. We fine-tune the pre-trained model for 30 epochs and select the best checkpoints based on validation loss. We provide the detailed hyperparameters in Table A2.

Results In the CUB dataset, our model sets a new standard with 86.73% and 88.80% accuracy with two different backbones (Table 5), exceeding Deformable ProtoPNet (86.4%) and even ProtoTree (87.20%). PEEB distinctly maps descriptors to visual inputs, facilitating easier debugging and clearer understanding as demonstrated in Fig. 4.

Table 5: PEEB is a state-of-the-art model (here, top-1 accuracy on CUB-200) w.r.t. explainable classifiers in supervised learning. * *Five ensembled models.*

Methods	Model size	Backbone	Accuracy
Base (ViT) (2021)	22M	DeiT-S (2021)	84.28
– Concept bottleneck classifiers			
Concept Bottleneck Models (2020)	11M	ResNet-18 (2016)	80.10
CPM (2023)	155M	ViT-B/16 (2021)	72.00
CDM (2023)	155M	ViT-B/16	74.31
LaBo (2023)	427M	ViT-L/14	81.90
– Part-based classifiers			
ProtoPNet (2019)	22M	DeiT-S	84.04
ProtoTree (2021)	92M*	ResNet-50 (2016)	87.20*
TesNet (2021)	79M	Dense121 (2017)	84.80
Deformable ProtoPNet (2022)	23M	ResNet-50	86.40
ProtoPFormer (2022)	22M	DeiT-S	84.85
ViT-Net (2022)	26M	DeiT-S	84.26
PEEB (**ours**)	155M	OWL-ViT$_{\text{base32}}$	**86.73**
PEEB$_{B16}$ (**ours**)	155M	OWL-ViT$_{\text{base16}}$	**88.80**

6 Discussion and Conclusion

Explainability and editability PEEB stands out as a transparent and editable classifier to users by grounding the text descriptors to the visual parts in the image (Fig. 4-bottom). This transparent decision-making process plays an important role in user understanding. For instance, in Fig. 4 (upper right), the challenge of understanding why the model predicts accurately persists, particularly when we already know that the descriptors are incorrect. In contrast, PEEB's explanations not only make errors like the mismatch between `throat` and `wings` more apparent but also enable users to adjust descriptions, thereby improving model accuracy without the need for retraining (Fig. 2).

This study introduced PEEB, a pioneering part-based, explainable, and editable bird classifier that leverages textual descriptors for bird parts. By grounding natural language descriptors with visual features, PEEB brings transparency to its decision-making. Besides, PEEB achieves superior performance in both GZSL and ZSL settings compared to existing state-of-the-art explainable models. It underscores PEEB's robustness and versatility in handling a variety of classification scenarios, especially in fine-grained tasks like avian classification. Moreover, our work contributes to the broader research community by developing the Bird-11K dataset, which encompasses a diverse range of bird species and presents a valuable resource for further explorations in fine-grained classification and beyond.

7 Limitations

Text encoder may not fully comprehend the bird descriptors Our text encoder, pre-trained on a broad image-text dataset, may not fully capture the intricate details specific to birds. Furthermore, CLIP text encoders trained by contrastive learning are known to suffer from the *binding* problem and do not understand some logical operators such as "and", "or", or negation. PEEB accuracy depends directly on the quality of the text encoder.

Dependency on image encoder for part visibility The image encoder's role in determining the visibility of bird parts in an image poses another limitation. Our model operates under the assumption that 12 parts are always visible in a bird image, requiring it to score these parts even when they might not be visually present. In an ideal scenario, the model should learn to assign the absence part a low score. This approach, which lacks direct supervision, relies heavily on unsupervised learning derived from class labels. Consequently, the limited dataset size of approximately 290K training images in Bird-11K may not sufficiently support robust unsupervised learning.

Hallucinations in GPT-4 descriptors The accuracy of our model is directly impacted by the quality of GPT-4 descriptors. Our empirical analysis across 20 bird classes revealed that, on average, 45% of these descriptors do not accurately reflect the birds' features (Appendix F.1). However, we observed that revising certain descriptors in the CUB dataset led to a significant improvement of +10 points in classification accuracy for those classes (Appendix F.2). This primitive observation suggests that PEEB can be further improved if trained with human-labeled descriptors.

Application beyond bird classification While PEEB is designed for fine-grained classification in general, our current work focuses exclusively on bird classification. This is not due to any inherent limitation in the model's design but rather a result of the limited availability of fine-grained, open-source datasets in other domains. While the model can be applied to a wide range of fine-grained classification tasks, the lack of public, large-scale datasets for fine-grained classification (e.g., dogs or butterflies) has directed our focus towards this specific area for the time being.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2015. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438.

Naman Bansal, Chirag Agarwal, and Anh Nguyen. 2020. Sam: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 8673–8683.

Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. 2014. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32.

Zhi Chen, Jingjing Li, Yadan Luo, Zi Huang, and Yang Yang. 2020. Canzsl: Cycle-consistent adversarial networks for zero-shot learning from natural language. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 874–883.

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

Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. 2022. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10265–10275.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Mohamed Elhoseiny, Yizhe Zhu, Han Zhang, and Ahmed Elgammal. 2017. Link the head to the" beak": Zero shot learning from noisy text description at part precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5640–5649.

Reza Esfandiarpoor and Stephen H Bach. 2023. Follow-up differential descriptions: Language models resolve ambiguities for image classification. *arXiv preprint arXiv:2311.07593*.

David Gunning, Eric Vorm, Jennifer Yunyan Wang, and Matt Turek. 2021. Darpa's explainable ai (xai) program: A retrospective. *Applied AI Letters*, 2(4):e61.

Chi Han, Hengzhi Pei, Xinya Du, and Heng Ji. 2022. Zero-shot classification by logical reasoning on natural language explanations. *arXiv preprint arXiv:2211.03252*.

Songhao Han, Le Zhuo, Yue Liao, and Si Liu. 2023. Llms as visual explainers: Advancing image classification with evolving visual descriptions. *arXiv preprint arXiv:2311.11904*.

Celina Hanouti and Hervé Le Borgne. 2023. Learning semantic ambiguities for zero-shot learning. *Multimedia Tools and Applications*, pages 1–15.

Ju He, Jie-Neng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. 2022. Transfg: A transformer architecture for fine-grained recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 852–860.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Zhong Ji, Yanwei Fu, Jichang Guo, Yanwei Pang, Zhongfei Mark Zhang, et al. 2018. Stacked semantics-guided attention model for fine-grained zero-shot learning. *Advances in neural information processing systems*, 31.

Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. 2022. Vit-net: Interpretable vision transformers with neural tree decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR.

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

Shayan Kousha and Marcus A Brubaker. 2021. Zero-shot learning with class description regularization. *arXiv preprint arXiv:2106.16108*.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. 2016. The unreasonable effectiveness of noisy data for fine-grained recognition. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 301–320. Springer.

Sachit Menon and Carl Vondrick. 2022. Visual classification via description from large language models. *arXiv preprint arXiv:2210.07183*.

Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. 2022. Simple open-vocabulary object detection with vision transformers. *ECCV*.

Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. 2022. This looks like that, because... explaining prototypes for interpretable image recognition. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I*, pages 441–456. Springer.

Meike Nauta, Ron Van Bree, and Christin Seifert. 2021. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943.

Giang Nguyen, Mohammad Reza Taesiri, and Anh Nguyen. 2022. Visual correspondence-based explanations improve ai robustness and human-ai team accuracy. *arXiv preprint arXiv:2208.00780*, 1.

Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. 2023. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*.

OpenAI. 2023. Gpt-4 technical report.

Konstantinos P Panousis, Dino Ienco, and Diego Marcos. 2023. Hierarchical concept discovery models: A concept pyramid scheme. *arXiv preprint arXiv:2310.02116*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Tzuf Paz-Argaman, Yuval Atzmon, Gal Chechik, and Reut Tsarfaty. 2020. Zest: Zero-shot learning from text descriptions using textual similarity and visual summarization. *arXiv preprint arXiv:2010.03276*.

Gerald Piosenka. 2022. Birds 525 - species image classification.

Sarah Pratt, Rosanne Liu, and Ali Farhadi. 2022. What does a platypus look like? generating customized prompts for zero-shot image classification. *arXiv preprint arXiv:2209.03320*.

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

Yunbo Rao, Ziqiang Yang, Shaoning Zeng, Qifeng Wang, and Jiansu Pu. 2023. Dual projective zero-shot learning using text descriptions. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(1):1–17.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

Dvir Samuel, Yuval Atzmon, and Gal Chechik. 2021. From generalized zero-shot learning to long-tail with class descriptors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 286–295.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Parhaam Vaibhav Rokde, Matthew Jansen. 2023. Indian birds species image classification. Dataset originally sourced from eBird, Cornell Lab of Ornithology. https://media.ebird.org/.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604.

Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. 2021. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12884–12893.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. 2011. The caltech-ucsd birds-200-2011 dataset.

Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. 2021. Interpretable image recognition by constructing transparent embedding space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 895–904.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. 2018. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. 2019. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492.

Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. 2020. Attribute prototype network for zero-shot learning. *Advances in Neural Information Processing Systems*, 33:21969–21980.

Mengqi Xue, Qihan Huang, Haofei Zhang, Lechao Cheng, Jie Song, Minghui Wu, and Mingli Song. 2022. Protopformer: Concentrating on prototypical parts in vision transformers for interpretable image recognition. *arXiv preprint arXiv:2208.10431*.

An Yan, Yu Wang, Yiwu Zhong, Chengyu Dong, Zexue He, Yujie Lu, William Yang Wang, Jingbo Shang, and Julian McAuley. 2023. Learning concise and descriptive attributes for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3090–3100.

Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. 2023. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19187–19197.

Mert Yuksekgonul, Maggie Wang, and James Zou. 2023. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*.

Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Xiaogang Wang, Hongsheng Li, and Hao Dong. 2020. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*.

Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. 2018. A generative adversarial approach for zero-shot learning from noisy texts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1004–1013.

A Architecture details

A.1 Image encoder and text encoder

We employ the image encoder and text encoder form OWL-ViT. In order to maintain a general understanding of natural languages and avoid overfitting our training samples, we keep the text encoder frozen for all training and experiments. This setup allows our design to be flexible about the choice of text encoder, e.g., one can easily replace the text encoder without changing other architecture.

A.2 Linear projection (for part embedding selection)

The image embedding will be forwarded to a **Linear Projection** layer, which is composed of a learnable logit scale, a learnable logit shift, and an Exponential Linear Unit (ELU) activation function. These processed image embeddings are then have the same dimension as the text embeddings. We select a single image embedding for each text query. In this context, the text queries correspond to the component names of the target object, which includes twelve distinct parts. This selection is based on the cosine similarity between the projected image embeddings and the text embeddings. Finally, the chosen images embeddings (before projection) will be sent to the **Part MLP** for classification and **Box MLP** for box prediction (Fig. A1, Step 1).

A.3 Part MLP

We introduce a **Part MLP** block to facilitate part-based classification. It comprises a three-layer MLP with GELU activations (Hendrycks and Gimpel, 2016). All the linear layers in the MLP are designed to match the dimensions of the visual embedding except for the last layer, which is specifically tailored to align with the size of the text embedding. This component takes an image embedding as input and projects it into the same space as the text embedding so that it can be directly compared with the text embedding.

A.4 Box MLP

The **Box MLP** retained from OWL-ViT consists of a three-layer MLP. It takes the visual embedding as input and generates a four-element vector corresponding to the center coordinates and size of a bounding box (e.g., `[x, y, width, height]`). It is important to note that the image embedding inputs of **Box MLP** and **Part MLP** layers are the same, as shown in Fig. A1, Step 2.

A.5 Visual part embedding selection

As shown in Fig. A1 step 1, 1c, the image embeddings are first projected by a Linear Projection layer and compute the dot product with the encoded part names. The image embeddings (before linear projection) are chosen as visual part embeddings by selecting the embedding that has the highest similarity scores with the corresponding part after the linear projection.

A.6 Descriptor embedding matching

To enhance the model's flexibility, we do not use a linear layer for classification. Instead, we adopt a strategy similar to CLIP: we compute the similarity matrix of the projected visual embeddings (image embeddings after processing by the **Part MLP**) and the text embeddings. Then, we sum the corresponding similarities of each part in the class; the class with the highest score is considered the predicted class as shown in Fig. A1, step 2, 2d. This design enables our proposed method to perform arbitrary ways of classification.

A.7 Implementation details

Our experiments are conducted under PyTorch (Paszke et al., 2019). We employ HuggingFace's (Wolf et al., 2020) implementation of OWL-ViT and use their pre-trained models. The DETR losses implementation (Carion et al., 2020) is employed directly from their official implementation.

A.8 Training hyperparameters

We provide the hyperparameters of all models trained in this work. Table A1 shows the details of the pre-training models. Table A2 presents the details of the fine-tuned models. All trainings utilize optimizer AdamW with Plateau Scheduler.

A.9 Computational budget and infrastructures

We use 8 Nvidia RTX A100 GPUs for our experiments. The pertaining approximate takes ~24 hours on Bird-11K. The fine-tuning takes 2 to 4 hours with one single GPU.

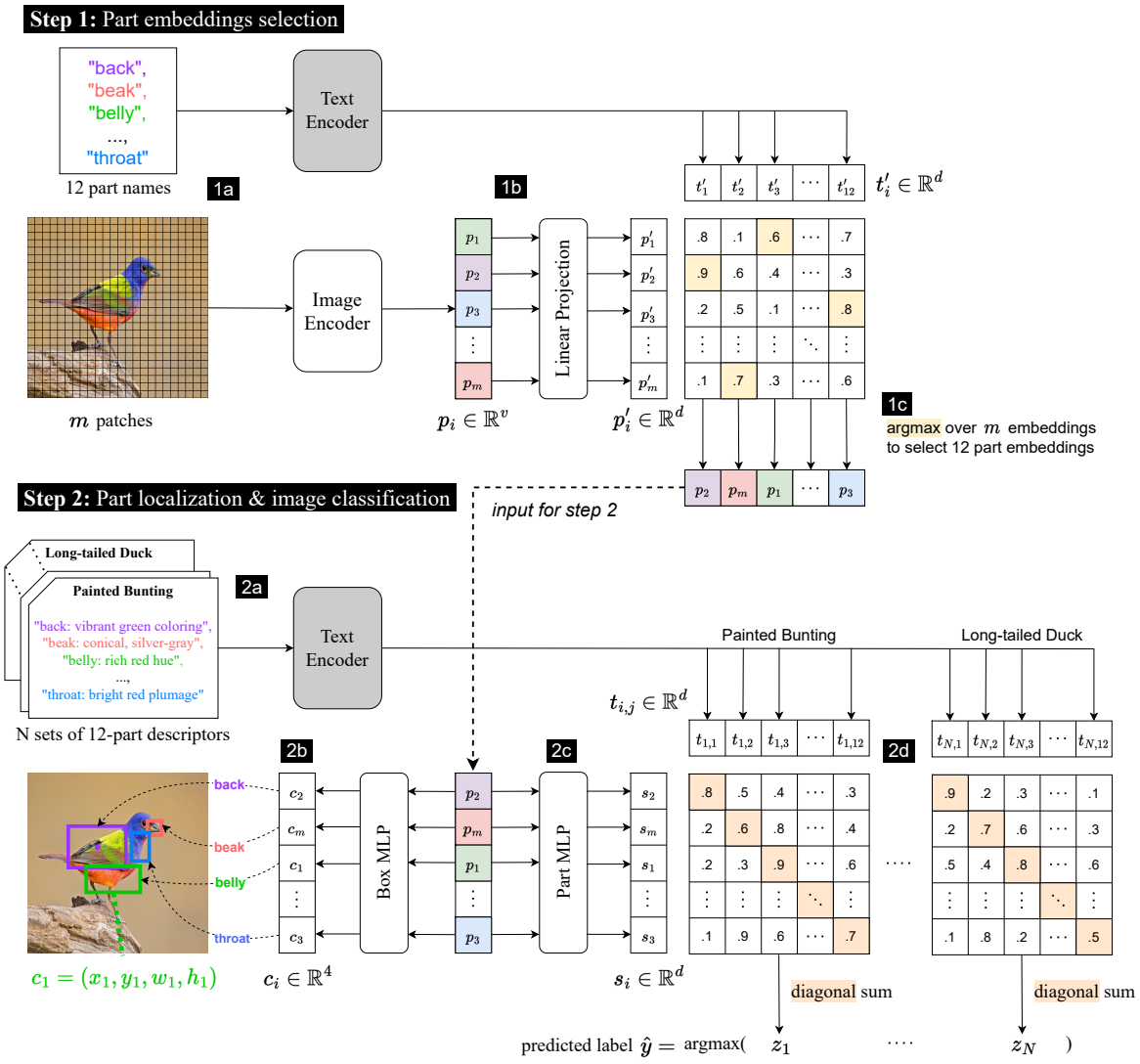


Figure A1: During the test time, we perform 2 steps. **Step 1**: (a) Encode an input image and texts (i.e. 12 part names) by the image and text encoder to get patch embeddings p_i and text embeddings t'_i. (b) Feed p_i to linear projection to get p'_i in the same dimensional space with t'_i and compute dot product between $\{p'_i\}$ and $\{t'_i\}$. (c) $\arg\max$ over m embeddings to select 12 part embeddings.
Step 2: (a) Encode input texts (i.e. N sets of 12-part descriptors) with the same text encoder to get t_i. (b) Feed the selected part embeddings to box MLP to localize parts (in center format). (c) Also feed the selected part embeddings to part MLP to get s_i in the same dimensional space with t_i (d) Compute dot product between $\{s_i\}$ and $\{t_i\}$, then diagonal sum for each class and $\arg\max$ over logits to get predicted label $\hat{y}$.

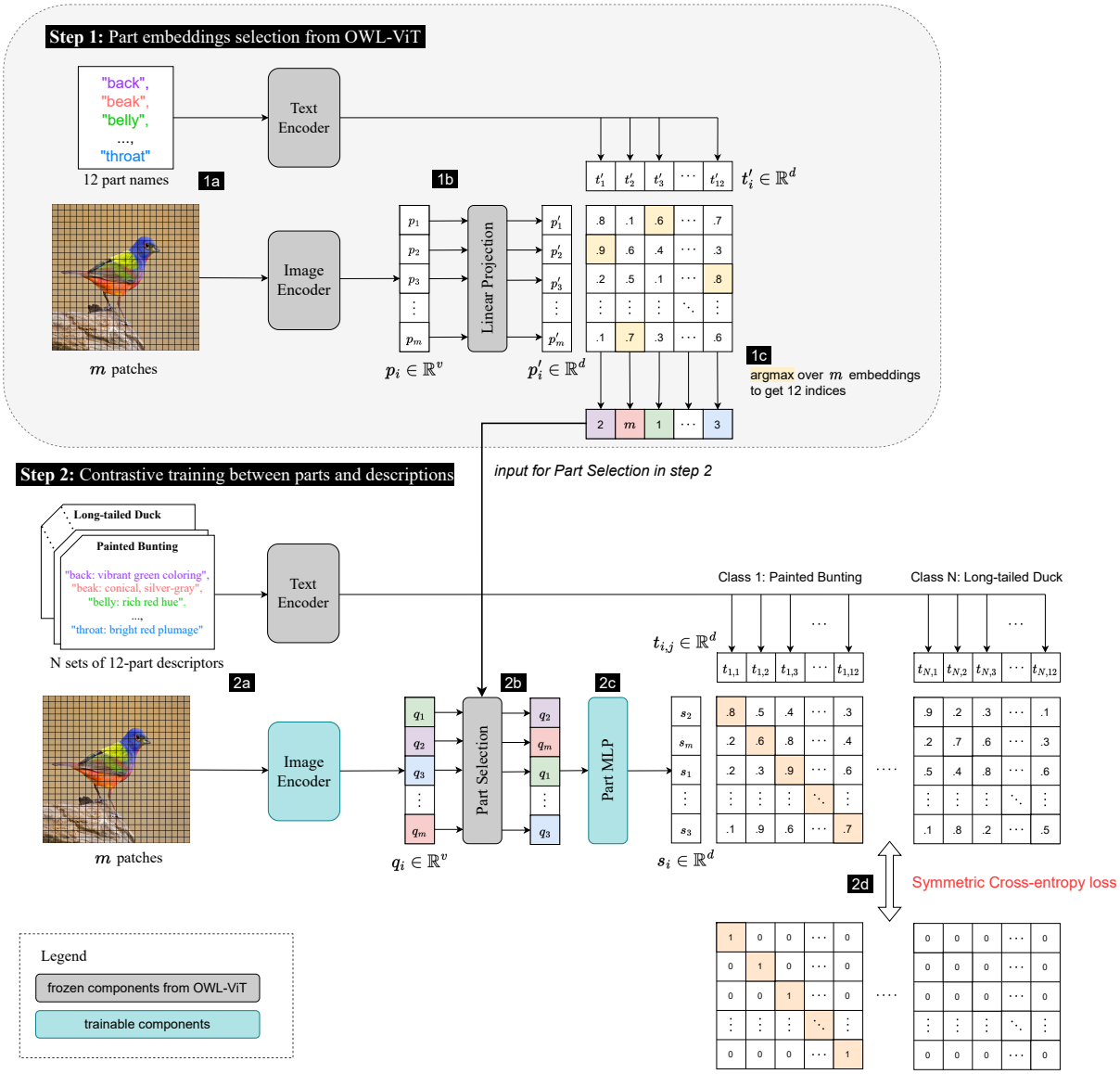


Figure A2: In pre-training stage 1, the objective is to let the Image Encoder learn the general representation of different parts of the birds. Therefore, in pre-training stage 1, we train the *Image Encoder* and part MLP contrastively. During the training, the **Step 1** utilizes a teacher model (OWL-ViT$_{\text{base32}}$) to help PEEB select 12 part embeddings. In **Step 2**, we update the model with symmetric Cross-Entropy loss. Here's the flow of **Step 1**: (1a) We utilize the teacher model to encode 12 part names and the image to derive the text embedding $\boldsymbol{t}'_i$, and the patch embedding $\boldsymbol{p}_i$. (1b) Then the patch embeddings $\boldsymbol{p}$ is forwarded to linear projection to obtain $\boldsymbol{p}'$, matching the dimension of $\boldsymbol{t}'$. (1c) We compute the dot product between $\boldsymbol{p}$ and $\boldsymbol{t}'$ and apply $argmax$ over $\boldsymbol{p}$ to derive 12 indices. In **Step 2**: (2a), We first encode the descriptors and the image with the *Text Encoder* and *Image Encoder* to obtain descriptor embeddings $\boldsymbol{t}$ and patch embeddings $\boldsymbol{q}$. (2b), Then we select the 12 patch embeddings based on the 12 indices from (1c). (2c), The 12 patch embeddings then forwarded to part MLP to derive $\boldsymbol{s}$, which has the same dimension as $\boldsymbol{t}$. Then, we compute the similarity matrix for the patch embedding and the descriptor embedding by computing the dot product between $\boldsymbol{s}$ and $\boldsymbol{t}$. (2d), we construct a one-hot encoded matrix based on the descriptors as the ground truth label and minimize the Symmetric Cross-Entropy loss between the similarity matrix in (2c) and the ground truth label.

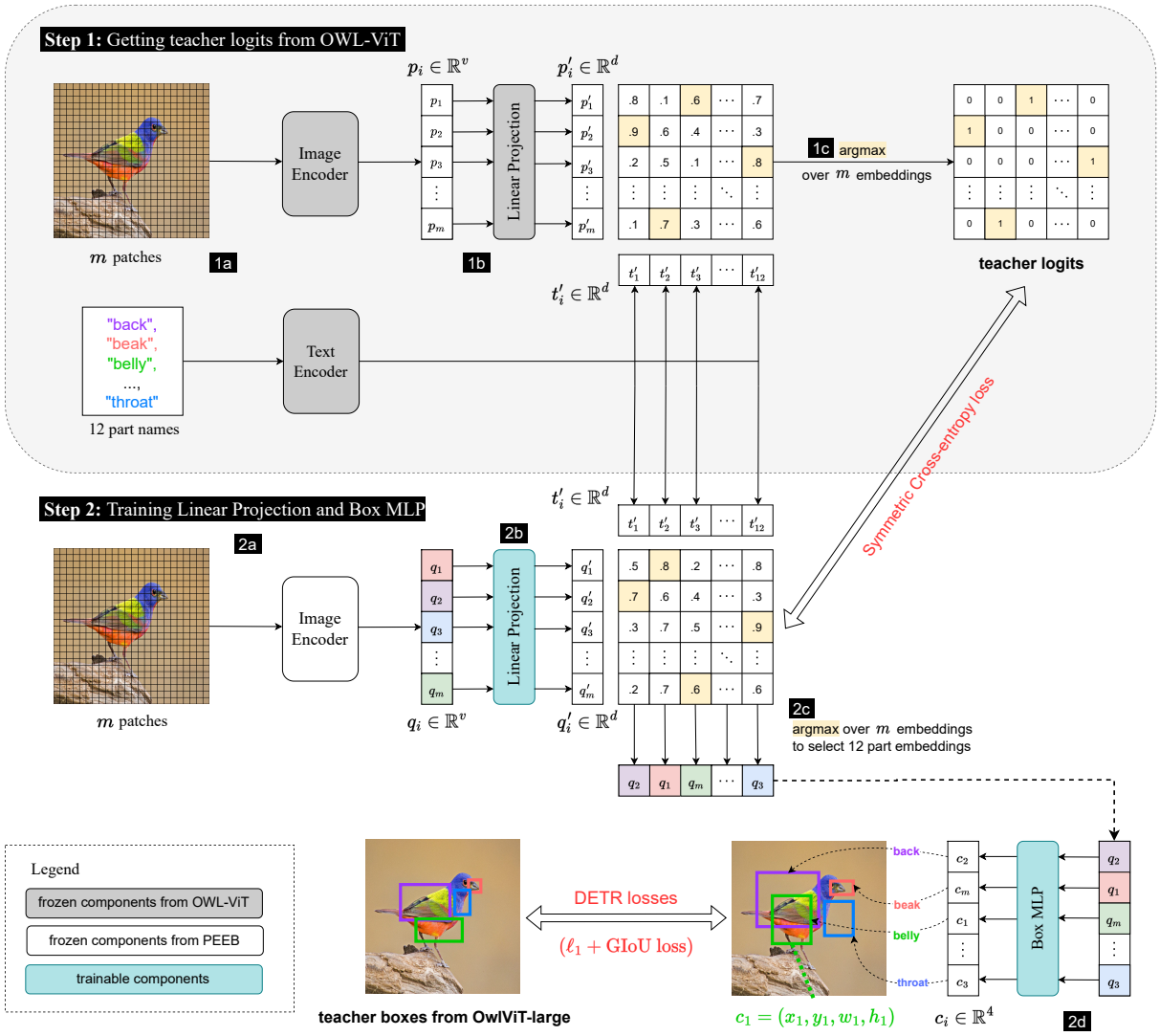


Figure A3: In pre-training stage 2, the goal is to eliminate the teacher model to obtain a standalone classifier. Therefore, the targeted components are linear projection and box MLP. Since these two components are taking care of different functionalities for patch embedding selection and box prediction, respectively, stage 2 training is a multi-objective training. We employ Symmetric Cross-Entropy loss to learn the patch embedding selection and DETR losses to refine the box predictions. In **Step 1**: (1a), We first encode the 12 part names and the image with *Text Encoder* and *Image Encoder* to obtain the text embedding $\boldsymbol{t}'_i$ and patch embedding $\boldsymbol{p}_i$. (1b) Then the patch embeddings $\boldsymbol{p}$ is projected by linear projection to obtain $\boldsymbol{p}'$. (1c) We then compute dot product between $\boldsymbol{p}'$ and $\boldsymbol{t}'$ and one-hot encode the matrix via the dimension of $\boldsymbol{p}'$ to obtain the "teacher logits". In **Step 2**: (2a), We encoder the image with *Image Encoder* to obtain patch embedding $\boldsymbol{q}_i$. (2b) The patch embeddings are then being projected by linear projection to derive $\boldsymbol{q}'$. (2c), We compute the dot product between projected patch embeddings $\boldsymbol{q}'$ and part name embeddings $\boldsymbol{t}'$ to obtain the similarity matrix. Then, we employ Symmetric Cross-Entropy loss between the similarity matrix and the "teacher logits" derived in (1c). (2d), Meanwhile, we select the 12 part embeddings by taking argmax over $\boldsymbol{q}'$. Then, the selected part embeddings are forwarded to box MLP to predict the coordinates of each part. We compute the DETR losses for the predicted coordinates and update the model.

Table A1: Pre-training details of our pre-trained models.

Model	Epoch	Batch size		LR	Weight decay	# in-batch classes		Early stop	Training set
		Train	Val			Train	Val		
Pre-training stage 1									
PEEB$_{[-\text{test}]}$	32	32	50	$2e^{-4}$	0.01	48	50	5	Bird-11K$_{[-\text{test}]}$
PEEB$_{[-\text{CUB}]}$	32	32	50	$2e^{-4}$	0.001	48	50	10	Bird-11K$_{[-\text{CUB}]}$
PEEB$_{[-\text{NAB}]}$	32	32	50	$2e^{-4}$	0.001	48	50	10	Bird-11K$_{[-\text{NAB}]}$
Pre-training stage 2									
PEEB$_{[-\text{test}]}$	32	32	50	$2e^{-5}$	0.01	48	50	5	Bird-11K$_{[-\text{test}]}$
PEEB$_{[-\text{CUB}]}$	32	32	50	$2e^{-5}$	0.001	48	50	5	Bird-11K$_{[-\text{CUB}]}$
PEEB$_{[-\text{NAB}]}$	32	32	50	$2e^{-5}$	0.001	48	50	5	Bird-11K$_{[-\text{NAB}]}$

Table A2: Details of our fine-tuned models.

Model	Fine-tune from	Epoch	Batch size	LR	Weight decay	Early stop	Training set
PEEB$^{\text{CUB}}_{[-\text{test}]}$	PEEB$_{[-\text{test}]}$	30	32	$2e^{-5}$	0.001	5	CUB
PEEB$^{\text{Akata}}_{[-\text{cub}]}$	PEEB$_{[-\text{CUB}]}$	5	32	$2e^{-5}$	0.001	5	CUB ZSL (2015)
PEEB$^{\text{SCS}}_{[-\text{cub}]}$	PEEB$_{[-\text{CUB}]}$	5	32	$2e^{-5}$	0.001	5	CUB-SCS
PEEB$^{\text{SCE}}_{[-\text{cub}]}$	PEEB$_{[-\text{CUB}]}$	5	32	$2e^{-5}$	0.001	5	CUB-SCE
PEEB$^{\text{SCS}}_{[-\text{nab}]}$	PEEB$_{[-\text{NAB}]}$	5	32	$2e^{-5}$	0.001	5	NABirds-SCS
PEEB$^{\text{SCE}}_{[-\text{nab}]}$	PEEB$_{[-\text{NAB}]}$	5	32	$2e^{-5}$	0.001	5	NABirds-SCE

B Model and dataset notations

B.1 Dataset notations

Following the conventional setup of ZSL, we execute certain exclusions to make sure none of the test classes or descriptors are exposed during pre-training. That is, Bird-11K$_{[-\text{CUB}]}$ and Bird-11K$_{[-\text{NAB}]}$ exclude all CUB and NABirds classes, respectively. For GZSL, we exclude all test sets in CUB, NABirds, and iNaturalist, denoted as Bird-11K$_{[-\text{test}]}$. We provide detailed statistics for the three pre-training sets in Table A3.

Table A3: Three pre-training splits for PEEB.

Training set	Number of images		Number of classes	
	Train	Val	Train	Val
Bird-11K$_{[-\text{test}]}$	234,693	29,234	10,740	9,746
Bird-11K$_{[-\text{CUB}]}$	244,182	28,824	10,602	9,608
Bird-11K$_{[-\text{NAB}]}$	216,588	27,996	10,326	9,332

B.2 Model notations

We adopt a strategy based on the datasets excluded during training to simplify our model naming convention. Specifically:

- PEEB$_{[-\text{test}]}$ is pre-trained model using Bird-11K$_{[-\text{test}]}$ datset.
- PEEB$_{[-\text{CUB}]}$ is pre-trained model using the Bird-11K$_{[-\text{CUB}]}$ dataset.
- PEEB$_{[-\text{NAB}]}$ is pre-trained model using the Bird-11K$_{[-\text{NAB}]}$ dataset.

We named fine-tuned models after the pre-trained model and the fine-tuned training set. For example, PEEB$^{\text{CUB}}_{[-\text{test}]}$ is fine-tuned from PEEB$_{[-\text{test}]}$, on CUB training set.

C Generating part-based descriptors

CUB annotations initially comprise 15 bird parts. However, distinctions between the left and right part are not essential to our method, we merge them into a single part (i.e., "left-wing" and "right-wing" are merged into "wings") Hence, we distilled the original setup into 12 definitive parts: *back, beak, belly, breast, crown, forehead, eyes, legs, wings, nape, tail, throat*. To compile visual part-based descriptors for all bird species within Bird-11K, we prompted GPT-4 (OpenAI, 2023) with the following input template:

```
A bird has 12 parts: back, beak, belly, breast,
crown, forehead, eyes, legs, wings, nape, tail
and throat. Visually describe all parts of
{class name} bird in a short phrase in bullet
points using the format 'part: short phrase'
```

Where `{class name}` is substituted for a given bird name.

The output is a set of twelve descriptors corresponding to twelve parts of the query species. e.g. The response for `Cardinal` is:

```
Cardinal: {
    back: vibrant red feathers,
    beak: stout, conical, and orange,
    belly: light red to grayish-white,
    breast: bright red plumage,
    crown: distinctive red crest,
    forehead: vibrant red feathers,
    eyes: small, black, and alert,
    legs: slender, grayish-brown,
    wings: red with black and white
        accents,
    nape: red feather transition to
        grayish-white,
    tail: long, red, and wedge-shaped,
    throat: bright red with sharp
        delineation from white belly
}
```

D Bird-11K dataset

We provide a brief statistic of Bird-11K in Table A4. Bird-11K is a diverse and long-tailed avian dataset that only includes bird images. The descriptors generated by GTP4 are in English and only describe the visual features of the corresponding class. We propose Bird-11K for academic research only.

Table A4: Number of images and species of different bird datasets. Our proposed dataset Bird-11K includes almost all avians on Earth.

Dataset	**Images**	**Species**
CUB-200-2011 (Wah et al., 2011)	12,000	200
Indian Birds (Vaibhav Rokde, 2023)	37,000	25
NABirds v1 (Van Horn et al., 2015)	48,000	400
Birdsnap v7 (Berg et al., 2014)	49,829	500
iNaturalist 2021-birds (Van Horn et al., 2021)	74,300	1,464
ImageNet-birds (Deng et al., 2009)	76,700	59
BIRDS 525 (Piosenka, 2022)	89,885	525
Macaulay Library at the Cornell Lab of Ornithology	55,283	10,534
Bird-11K (Raw Data)	440,934	11,097
Bird-11K (pre-training set)	294,528	10,811

Data splits We provide data splits and metadata, e.g., file names, image size, and bounding boxes, along with the instruction of Bird-11K construction in our repository. Note that the Bird-11K dataset is for pre-training purposes; it is important to execute exclusion based on the test set.

License and terms

- CUB (Wah et al., 2011): The dataset can be freely used for academic and research purposes; commercial use is restricted.
- Indian Birds (Vaibhav Rokde, 2023): CC0: Public Domain.
- NABirds v1 (Van Horn et al., 2015): For non-commercial research purposes, other use is restricted [3] here for detail: .
- Birdsnap v7 (Berg et al., 2014): The dataset creator provides no specific license or terms of use. We only use this dataset for academic research until more specific details can be obtained.
- iNaturalist 2021-birds (Van Horn et al., 2021): CC0: Public Domain.
- ImageNet-birds (Deng et al., 2009): BSD-3-Clause license.
- BIRDS 525 (Piosenka, 2022): CC0: Public Domain
- Cornell eBird: We used the following 55,384 recordings from the Macaulay Library at the Cornell Lab of Ornithology. The data is for academic and research purposes only, not publicly accessible unless requested. (Please refer to our Supplementary Material for the full list):

ML187387391, ML187387411, ML187387421, ML187387431, ML262407521, ML262407481, ML262407531, ML262407491, ML262407511, ML257194111 ML257194071, ML257194081, ML257194061, ML495670791, ML495670781, ML495670801, ML495670771, ML183436431, ML183436451, ML183436441 ML183436411, ML183436421, ML256545901, ML256545891, ML256545841, ML256545851, ML256545831, ML169637941, ML238083081, ML169637881 ML169637911, ML238083111, ML238083051, ML169637971, ML299670841, ML64989231, ML299670831, ML64989241, ML299670791, ML64989251 ML246866001, ML246865941, ML246866011, ML246865961, ML246865971, ML333411961, ML240835531, ML240835541, ML240835701, ML240835591 ML245260391, ML245260341, ML245260371, ML245260411, ML245260421, ML245260431, ML245260441, ML240866351, ML240866331, ML240866321 ML240866341, ML240866371, ML248318661, ML248318571, ML248318591, ML248318581, ML248318631, ML245204281, ML245204311, ML245204371 ML245204381, ML245204291, ML245603571, ML245603521, ML245603511, ML245603491, ML245603501, ML245603601, ML245257771, ML245257651 ML245257631, ML245257661, ML245257761, ML247221051, ML247221061, ML247221071, ML247221081, ML240365811, ML240365751, ML240365781 ML240365761, ML300579541, ML247298551, ML247298541, ML247298561, ML247298611, ML247298571, ML247298591, ML247298601, ML247298631...

E Analysis

E.1 PEEB outperforms M&V in CUB and NABirds in ZSL setting

To rigorously evaluate the ZSL capabilities of our pre-trained models, we introduce a stress test on the CUB and NABirds datasets. The crux of this test involves excluding all classes from the target dataset (CUB or NABirds) during the pre-training. The exclusion ensures that the model has no prior exposure to these classes. Subsequently, we measure the classification accuracy on the target dataset, comparing our results against benchmarks set by CLIP and M&V in the scientific name test. In this experiment, we consider the scientific name test a ZSL test for CLIP and use them as the baseline because the frequencies of scientific names are much lower than common ones.

Experiment To conduct this test, we pre-train our model on Bird-11K$_{[-\text{CUB}]}$ and Bird-11K$_{[-\text{NAB}]}$, which deliberately exclude images bearing the same class label as the target dataset. Specifically, we test on our pre-train model PEEB$_{[-\text{CUB}]}$ and PEEB$_{[-\text{NAB}]}$ (see Table A1 for details), respectively.

Results The primary objective is to ascertain the superiority of our pre-trained model, PEEB, against benchmarks like CLIP and M&V. For CUB, our method reported a classification accuracy of 17.9%, contrasting the 5.95% and 7.66% achieved by CLIP and M&V, respectively, as shown in Table A5. The PEEB score, which is marginally higher (+10) than M&V, highlights the advantages of our method that utilizes component-based classification. On the NABirds, our method surpassed the CLIP and M&V by (+1) point. The performance disparity between CUB and NABirds can be attributed to two factors: the elevated complexity of the task (555-way classification for NABirds versus 200-way for CUB) and the marked reduction in training data. An auxiliary observation, detailed in Appendix E.3, indicates that our pre-trained model necessitates at least 250k images to achieve admirable classification accuracy on CUB, but we only have 210k images training images in Bird-11K$_{[-\text{NAB}]}$ (Table A3).

Table A5: Stress test results on CUB and NABirds datasets. Despite the ZSL challenge, our method consistently surpasses CLIP and M&V. This underscores the robust generalization of our approach, which leverages descriptors for classification.

Method	CLIP	M&V	PEEB (ours)
CUB	5.95	7.66	**17.90**
NABirds	4.73	6.27	**7.47**

[3] See Terms of Use

E.2 Performance measurement on different noisy levels

In our evaluations, as indicated in Table 2, we discerned a marked performance disparity between the iNaturalist dataset and others. Probing this further, we identified image noise as a principal contributor to these discrepancies.

Experiment A qualitative assessment of the iNaturalist test images revealed a significantly higher noise level than CUB or NABirds. To systematically study this, we utilize the object detector OWL-ViT$_{\text{large}}$ to measure the size of the bird within the images. We formulated two filtered test sets based on the detector's output, categorizing them by the bird's size, specifically, the detected bounding box. Images were filtered out if the bird's size did not exceed predetermined thresholds (areas of 100^2 or 200^2 pixels). Larger birds naturally reduced other content by occupying more image space, thus serving as a proxy for reduced noise. All three test sets, including the original, were evaluated using our pre-trained model PEEB$_{[-\text{test}]}$.

Results The results presented in Table A6 reveal a clear trend: as the image noise level decreases, the classification accuracy consistently improves, with gains ranging from (+6 to +17) points across the various methods. Notably, cleaner images consistently yield better results. At each noise level, our method outperforms the alternatives. While our method exhibits an impressive (+17 points) accuracy boost on the cleanest test set, this substantial gain also indicates that our model is sensitive to image noise.

Table A6: The table showcases the classification accuracies on iNaturalist as we vary the noise levels. The data underscores that the performance disparity on iNaturalist is predominantly due to image noise. While all methods improve with cleaner images, our model exhibits the most substantial gains, particularly in the least noisy sets.

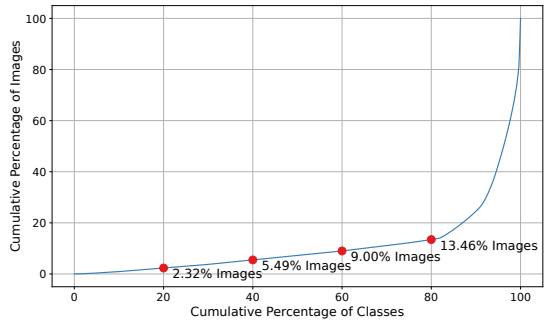
Splits	CLIP	M&V	PEEB (ours)
Original	16.36	17.57	**25.74**
$> 100^2$ pixels	20.18	21.66	**35.32**
$> 200^2$ pixels	22.88	24.90	**42.55**

E.3 Number of training images is the most critical factor towards classification accuracy

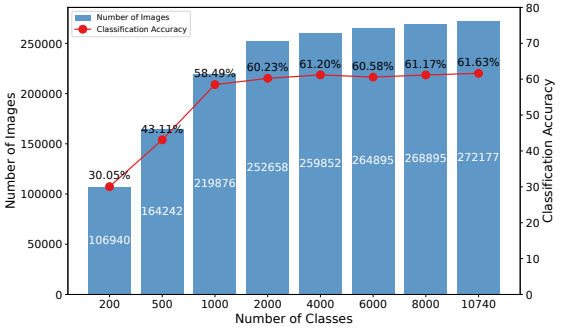
Bird-11K, as shown in Fig. A4a, is a highly imbalanced dataset characterized by a large amount of long-tailed classes. We conduct a comprehensive study to discern how variations in the number of classes and images affect the classification accuracy of our pre-trained models. Predictably, the volume of training images occurred as the most influential factor. However, a noteworthy observation was that the abundance of long-tailed data enhanced the model's accuracy by approximately +1.5 points.

Experiment We curated eight training sets based on varying class counts: 200, 500, 1,000, 2,000, 4,000, 6,000, 8,000, and 10,740. For each set, we maximized the number of training images. It is important to note that a set with a lesser class count is inherently a subset of one with a higher count. For instance, the 500-class set is a subset of the 2,000-class set. For each split, we apply the same training strategy as in Sec. 4.3.1, and choose the checkpoint with the best validation accuracy. We consider the CUB test set as a generic testing benchmark for all variants.

Results As illustrated in Figure Fig. A4b, there is a pronounced correlation between the increase in the number of images and the corresponding surge in accuracy. For instance, an increment from 106K to 164K images led to a rise in classification accuracy from 30.05% to 43.11%. The accuracy appears to stabilize around 60% when the image count approaches 250K. This trend strongly suggests that the volume of training images is the most critical factor for the pre-trained model. We believe that the accuracy of the pre-trained model could be further enhanced if enough data is provided. Interestingly, a substantial amount of long-tailed data bolsters the model's performance, evident from +1.5 points accuracy improvement when comparing models trained on 2,000 classes to those on 10,740 classes. Note that the additional classes in the latter set averaged merely 2.2 images per class.



(a) The Cumulative Distribution Function (CDF) plot for the Bird-11K dataset.



(b) Correlation between the number of training images/-classes and accuracy.

Figure A4: The CDF plot (a), underscores significant imbalance of the Bird-11K dataset. While the dataset has abundant long-tailed classes, e.g., a striking 80% of the classes contribute to only 13.46% of the entire image count. The plot (b) showcases the correlation between the number of training images/classes and the resulting classification accuracy. As the image count grows, there is a noticeable surge in accuracy, which nearly stabilizes upon surpassing 250K images. Additionally, a significant amount of long-tailed data contributes to a +1.5 points boost in accuracy.

E.4 Ablation study on the influence of parts utilized

In this ablation study, we aimed to measure the impact of varying the number of distinct "parts" (back, beak, belly, breast, crown, forehead, eyes, legs, wings, nape, tail, and throat) used in our model. We experiment with a range from a single part to all 12 identifiable parts. Interestingly, even with a solitary part, the model could make correct predictions, though there was an evident decline in performance, approximately -20 points.

Experiment Our testing ground is the pre-trained model PEEB$_{[-\text{test}]}$, evaluated against the CUB test set. We assessed the model's prowess utilizing various subsets of parts: 1, 3, 5, 8, and all 12. These subsets were derived based on the frequency of visibility of the parts within the CUB dataset, enabling us to compare the model's performance when relying on the most frequently visible parts versus the least. For comparison, we also conduct a similar experiment on M&V, where we only use 1, 3, 5, 8, and 12 descriptors (if possible).

Results Relying solely on the most frequent part led to a decline in classification accuracy by around -20 points, registering at 45.44%. In contrast, utilizing the least frequent part resulted in a sharper drop of around -27, with an accuracy of 37.02%. As the model was furnished with increasing parts, its accuracy improved incrementally. The data underscores that optimal performance, an accuracy of 64.33%, is attained when all 12 parts are included. For M&V, the accuracy keeps increasing homogeneously from 5 to 12 descriptors, hinting that accuracy may increase further by increasing the number of descriptors.

E.5 Training is essential for PEEB's classification efficacy

In this ablation study, we highlight the pivotal role of training in the performance of PEEB on bird classification tasks. We demonstrate that without adequate tuning, the results are indistinguishable from random chance.

Experiment We conduct the experiment based on OWL-ViT$_{\text{base32}}$. We retain all components as illustrated in Fig. A1, with one exception: we substitute the part MLP with the MLP layer present in the box prediction head of OWL-ViT because the proposed layers require training. The MLP layers in the box prediction head project the part embeddings to match the dimensionality of the text embeddings. Our focus is on assessing the classification accuracy of the untuned PEEB on two datasets: CUB and NABirds.

Results Table A8 reveals the outcomes of our experiment. Without training, PEEB yields classification accuracies of 0.55% for CUB and 0.31% for NABirds, both of which are proximate to random chance (0.5% for CUB and 0.1% for NABirds). However, with training, the model's performance dramatically transforms: 64.33% for CUB (an increase of +63.78 points) and 69.03% for NABirds (a leap of +68.72 points) for PEEB$_{[-\text{test}]}$. These pronounced disparities underscore the vital role of training in PEEB.

Table A7: Classification accuracy on the CUB test set that uses a different number of parts. Performance dips significantly with just one part, especially for the least visible ones. Maximum accuracy is reached with all 12 parts. The last row of the table also shows the accuracy of (Menon and Vondrick, 2022) method which employs a different number of parts. It is evident that their method is insensitive to the number of parts used, which may not reflect a realistic scenario.

Number of Parts (descriptors)	1	3	5	8	12
Accuracy (most frequent parts)	45.44	56.48	59.89	61.32	**64.33**
Accuracy (least frequent parts)	37.02	55.51	60.04	61.13	**64.33**
Accuracy of (Menon and Vondrick, 2022)	51.93	52.87	52.83	53.33	53.92

Table A8: Impact of Training on Classification Accuracies: Untuned PEEB yields 0.55% on CUB and 0.31% on NABirds, almost mirroring random chance. With training (PEEB$_{[-\text{test}]}$), accuracy surges by +63.78 points on CUB and +68.72 points on NABirds.

	CUB	NABirds
PEEB (no training)	0.55	0.31
PEEB$_{[-\text{test}]}$ pre-trained	64.33	69.03
PEEB$^{\text{CUB}}_{[-\text{test}]}$ finetuned	86.73	-

E.6 Failure analysis

Since PEEB has two branches, box detection, and descriptor matching, we would like to find out, in the failure case, what is the main cause. i.e., is it because of the mismatch in the descriptor to the part embeddings? Or is it because the box detection is wrong? From our ablation study, it turns out that most errors come from the descriptor-part matching.

Experiment We conduct the experiment with PEEB$_{[-\text{test}]}$ on CUB test set. Specifically, we measure the box detection accuracy based on the key point annotation in CUB dataset, i.e., We consider the box prediction as **correct** if the prediction includes the human-annotated key point. We report the box prediction error rate (in %) based on parts.

Results As shown in Table A9, the average error rate difference between success and failure cases is merely 0.38. That is, in terms of box prediction, the accuracy is almost the same, disregarding the correctness of bird identification. It indicates that the prediction error is predominantly due to the mismatch between descriptors and part embeddings. We also noted that some parts, like `Nape` and `Throat`, have a very high average error rate, which may greatly increase the matching difficulties between descriptors and part embeddings.

E.7 Evaluation of predicted boxes from PEEB

Our proposed method primarily aims to facilitate part-based classification. While the core objective is not object detection, retaining the box prediction component is paramount for ensuring model explainability. This section delves into an evaluation of the box prediction performance of our method against the OWL-ViT$_{\text{base32}}$ model.

Experiment Given our focus on part-based classification, we aimed to ascertain the quality of our model's box predictions. To this end, we employed two metrics: mean Intersection over Union (IoU) and precision based on key points. We opted for mean IoU over the conventional mAP because: (1) Ground-truth boxes for bird parts are absent, and (2) our model is constrained to predict a single box per part, ensuring a recall of one. Thus, we treat OWL-ViT$_{\text{large}}$'s boxes as the ground truth and evaluate the box overlap through mean IoU. Furthermore, leveraging human-annotated key points for bird parts, we measure the precision of predicted boxes by determining if they contain the corresponding key points. We evaluate our fine-tuned models on their corresponding test sets. For instance, PEEB$^{\text{Akata}}_{[-\text{cub}]}$, fine-tuned based on the CUB split (Akata et al., 2015), is evaluated on the CUB test set.

Results Our evaluation, as presented in Table A10, shows that PEEB's box predictions do not match those of OWL-ViT$_{\text{base32}}$. Specifically, on average, there is a -5 to -10 points reduction in mean IoU for CUB and NABirds datasets, respectively. The disparity is less distinct when examining precision based on human-annotated key points; our method records about -0.14 points lower precision for CUB and -3.17 points for NABirds compared to those for OWL-ViT$_{\text{base32}}$. These observations reinforce that while PEEB's box predictions might not rival these dedicated object detection models, they

Table A9: Error rate of Box Prediction in Failure and Success Cases. We report the box prediction error rate, depending on whether the prediction box includes ground truth key points. No major difference is found between them, which means the failure is largely due to the part-descriptor mismatch.

Body Part	**Average**	**Back**	**Beak**	**Belly**	**Breast**	**Crown**	**Forehead**	**Eyes**	**Legs**	**Wings**	**Nape**	**Tail**	**Throat**
Failure Cases	16.52	23.38	3.28	8.06	15.96	7.41	24.72	7.29	5.63	3.36	64.79	7.25	27.07
Success Cases	16.14	23.03	2.96	7.44	18.64	7.13	21.53	3.93	6.85	2.68	68.66	6.40	24.38
Difference	**0.38**	0.35	0.33	0.62	-2.68	0.28	3.19	3.36	-1.22	0.68	-3.87	0.85	2.68

consistently highlight the same parts identified by such models as shown in Fig. A5. It is important to note that our approach utilized the same visual embeddings for both classification and box prediction tasks. This alignment emphasizes the part-based nature of our model's predictions.

Table A10: Model evaluation on CUB and NABirds test sets. We evaluate the predicted boxes on two *ground-truth* sets; (1) predicted boxes from OWL-ViT$_{\text{large}}$ as ground-truths, and (2) OWL-ViT$_{\text{large}}$'s boxes that include the human-annotated key points. Our method has slightly lower performance in terms of mean IoU but comparable precision.

	Models	Mean IoU		
		(1) All	(2) w/ Keypoints	Precision
CUB	OWL-ViT$_{\text{large}}$	**100.00**	**100.00**	**83.83**
	OWL-ViT$_{\text{base32}}$	44.41	49.65	83.53
	PEEB (Average)	35.98	40.14	83.39
	PEEB$^{\text{CUB}}_{[-\text{test}]}$	37.45	41.79	81.55
	PEEB$^{\text{Akata}}_{[-\text{cub}]}$	35.11	39.14	82.72
	PEEB$^{\text{SCS}}_{[-\text{cub}]}$	35.77	39.96	84.89
	PEEB$^{\text{SCE}}_{[-\text{cub}]}$	35.58	39.67	84.38
NABirds	OWL-ViT$_{\text{large}}$	**100.00**	**100.00**	**85.01**
	OWL-ViT$_{\text{base32}}$	40.14	47.63	83.89
	PEEB (Average)	36.47	42.01	80.72
	PEEB$^{\text{SCS}}_{[-\text{nab}]}$	36.45	42.03	80.09
	PEEB$^{\text{SCE}}_{[-\text{nab}]}$	36.49	41.99	81.34

F Study on GTP-4 generated descriptors

F.1 Noise measurement in GPT-4 generated descriptors

In this section, we conduct an empirical analysis to quantify the noise in descriptors generated by GPT-4 for 20 different classes within the CUB dataset. To achieve this, we manually inspect each descriptor and tally the instances where at least one factual error is present. Our findings reveal that every one of the 20 classes contains descriptors with errors, and on average, 45% of the descriptors necessitate corrections. This substantial noise level underscores the need for further refinement in our work, particularly in text descriptors.

We observe a notably high error rate in descriptors on the *back* and *wings*, with approximately 60% of these containing inaccurate information (refer to Table A11). This could be attributed to the challenges in distinguishing between the *back* and *wings*, given that the *back* is typically positioned behind the *wings*, yet exhibits considerable variability in size and shape. Addressing all descriptor issues by revising all 11,000 fine-grained descriptors would demand a significant investment of time and resources, which is beyond the scope of the current work. As such, we identify this as an area for future research and development, aiming to enhance the quality of the Bird-11K dataset.

F.2 Revising descriptors improves classification accuracy

As mentioned in the limitation section, the descriptors are generated from GPT-4 and therefore noisy and incorrect. Given that PEEB accepts open vocabulary inputs for classification, a natural way to improve classification accuracy is to improve the correctness of the descriptors.

Experiment We first collect descriptors of 183 CUB classes from AllAboutBirds. We then prompt GPT-4 to revise our original descriptors by providing the collected descriptor. We revise the descriptors with the following prompt:

```
Given the following descriptors of {class name}: {AllAboutBirds descriptors}. Can you revise the incorrect items below (if any) of this bird, return them as a Python dictionary, and use the key as the part name for each item? If a partś descriptor is not specifically described or cannot be inferred from the definition, use your own knowledge. Otherwise, leave as is. Note: please use a double quotation mark for each item such that it works with JSON format.

{Original descriptors}
```

Where `{class name}` the placeholder for the class name, `{AllAboutBirds descriptors}` is the description collected from AllAboutBirds, `{Original descriptors}` is the descriptors we used for training.

Table A11: Summary of manual inspection results for 20 classes, highlighting the need for revision in GPT-4 generated descriptors. An average error rate of 45% indicates substantial room for improvement.

	Back	Beak	Belly	Breast	Crown	Forehead	Eyes	Legs	Wings	Nape	Tail	Throat	Average
Error Rate	60	30	50	40	50	55	50	20	60	50	35	40	45

Due to the errors in the descriptors we used to train PEEB, simply replacing the descriptors with their revised version does not lead to better performance. Because the incorrect descriptors in training change the meaning of some of the phrases. For example, the belly of `Blue bunting` is pure blue, but the descriptors from GPT-4 is *soft, creamy white*. In addition, the GTP-4 uses the exact same descriptor in the belly for other classes, e.g., `Blue breasted quail`, which should be cinnamon. `Blue Fronted Flycatcher`, which should be yellow. Training the same descriptors with different colors confuses the model, and the model will convey the phrase "creamy white" with a different meaning to humans. Therefore, simply changing the descriptors to their' revised version will not work. We empirically inspect the descriptors that PEEB can correctly respond to and replace the class descriptors with the revised version. Specifically, we replace the descriptors of 17 classes in CUB and test the classification accuracy on $\text{PEEB}_{[-\text{test}]}$.

Results As shown in Table A12, the overall accuracy increase +0.8 points. The average improvement of the revised class is around +10.8, hitting that if we have correct descriptors of all classes, we may significantly improve the classification accuracy of the pre-trained model. However, correcting all 11k class descriptors is too expensive and out of the scope of this work. We leave it as a further direction of improving the part-based bird classification.

Table A12: The revised descriptors result in +0.8 for $\text{PEEB}_{[-\text{test}]}$ in CUB. In particular, the average improvement among the 17 revised classes is +10.8, hinting at the large potential of our proposed model.

Descriptors	Original	Partially Revised	Avg. Improvement
$\text{PEEB}_{[-\text{test}]}$	64.33	**65.14**	10.80

G Qualitative Inspections

G.1 Visual comparison of predicted boxes

We provide a visual comparison of the box prediction from $\text{OWL-ViT}_{\text{large}}$, $\text{OWL-ViT}_{\text{base32}}$, and PEEB in Fig. A5. We find that despite the fact that our predicted boxes have lower mean IoU compared to $\text{OWL-ViT}_{\text{large}}$, they are visually similar to the boxes as $\text{OWL-ViT}_{\text{base32}}$.

G.2 Qualitative examples of using randomized descriptors

We visually compare M&V and PEEB based on their utilization of descriptors. (Figs. A6 to A8). Specifically, we randomly swap the descriptors of the classes and then use these randomized descriptors as textual inputs to the tested models to see how they perform. We observe that the scores from M&V tend to cluster closely together. Surprisingly, M&V's prediction remains unchanged despite the inaccurate descriptors. In contrast, PEEB, when presented with randomized descriptors, attempts to identify the best match grounded on the given descriptors.

G.3 Examples of PEEB explanations

Figs. A9 to A11 are examples of how PEEB makes classification based on the descriptors and how it can reject the predictions made by M&V. Since we aggregate all descriptors for the final decision, even if some of them are similar in two classes, our method can still differentiate them from other descriptors. For instance, in Fig. A9, while other descriptors are similar, PEEB can still reject `chesnut-sided warbler` thanks to the distinct features of *forehead*, *throat* and *belly*.

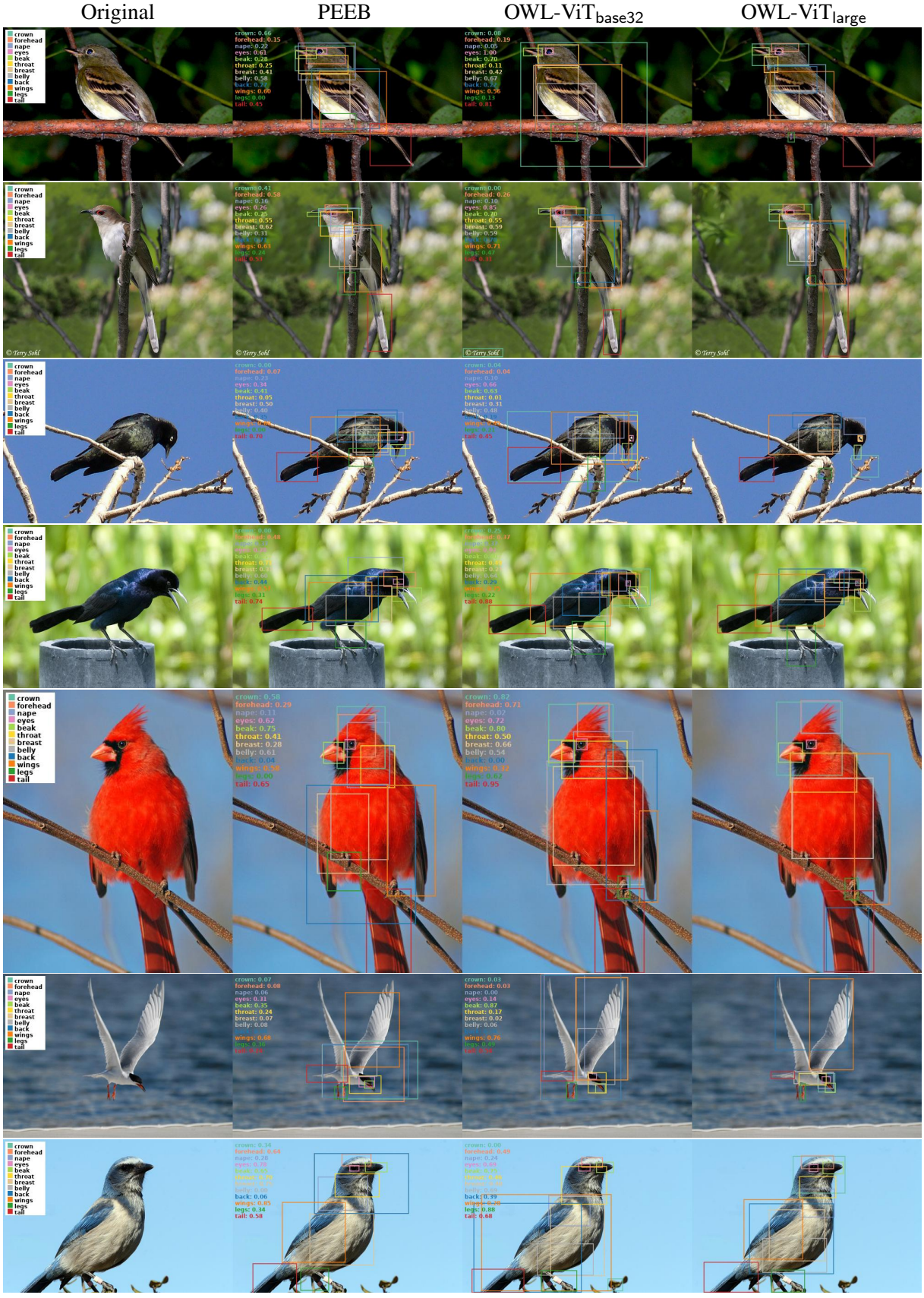


Figure A5: Our predicted boxes (second column) often align closely with those of OWL-ViT$_{base32}$ (third column). However, slight shifts can lead to significant IoU discrepancies. For instance, in the first row, both PEEB and OWL-ViT$_{base32}$ accurately identify the tail. Yet, variations in focus yield a stark IoU contrast of 0.45 versus 0.81.

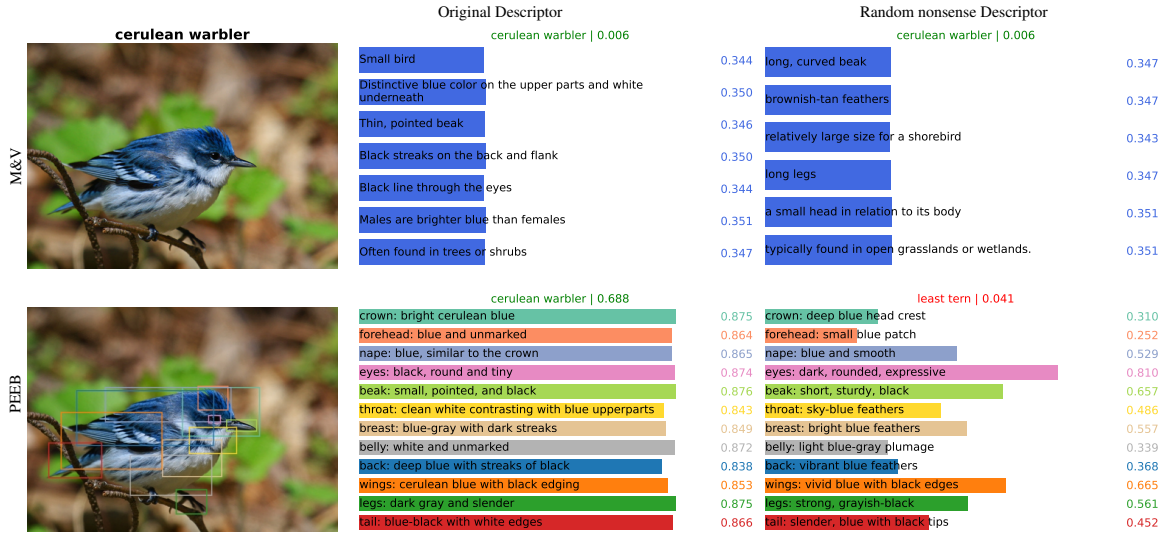


Figure A6: Qualitative example of original descriptors vs. randomized descriptors. Upon swapping descriptors randomly, the prediction outcomes from M&V exhibit minimal variations.

Figure A7: Qualitative example of original descriptors vs. randomized descriptors. Since PEEB's decision is made by the descriptors, the model will try to find the descriptors that best match the image. e.g., in the random descriptors, most parts are blue.

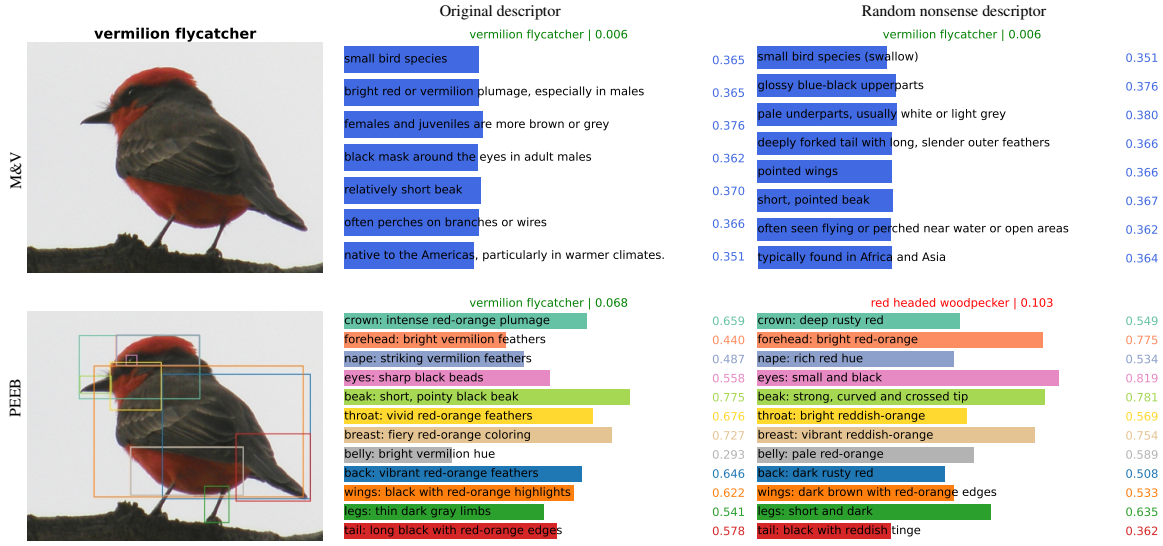


Figure A8: Qualitative example of original descriptors vs. randomized descriptors. M&V maintains similar scores even for mismatched descriptors. For instance, "bright red or vermilion plumage, especially in males" receives a score lower than "glossy blue-black upperparts". Conversely, PEEB leverages the descriptors for classification, consistently relying on the descriptors that most closely align with the image.

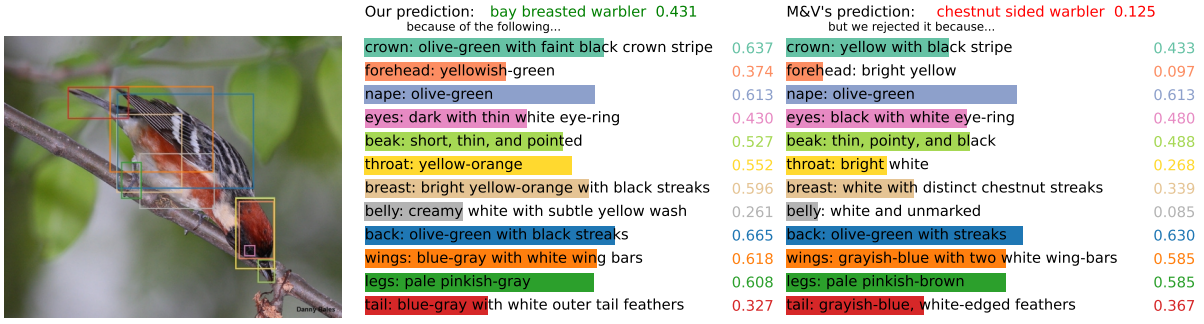


Figure A9: An example of PEEB explanation. We can see that the descriptors of these two classes are largely similar, but PEEB makes the correct prediction based on the distinctive feature of the forehead in the two classes.

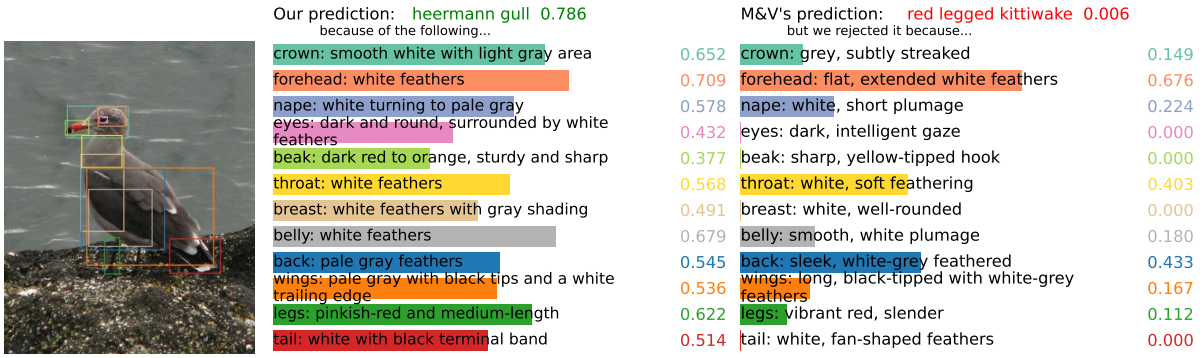


Figure A10: An example of PEEB explanation. M&V incorrectly classifies it as `red-legged kittiwake` where the `heermann gull` does not have red legs but a red beak. This example shows that CLIP is strongly biased towards some particular descriptors.

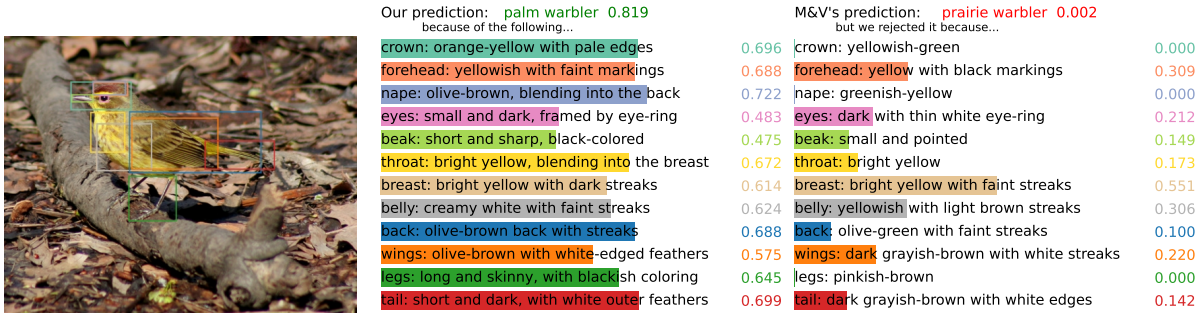


Figure A11: An example of PEEB explanation. We can see that when the descriptor does not match the image, the matching score tends to be zero, e.g., *crown: yellowish-green*. The clear differences in scores provide us transparency of the model's decision.