

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

Thang M. Pham^{*†}
thangpham@auburn.edu

Peijie Chen^{*†}
pzc0018@auburn.edu

Tin Nguyen^{*†}
ttn0011@auburn.edu

Seunghyun Yoon[§]
syoona@adobe.com

Trung Bui[§]
bui@adobe.com

Anh Totti Nguyen[†]
anh.ng8@gmail.com

[†]Auburn University

[§]Adobe Research

Abstract

CLIP-based classifiers rely on the prompt containing a `{class name}` that is known to the text encoder. Therefore, they perform poorly with new classes or the classes whose names rarely appear on the Internet (e.g., scientific names of birds). For fine-grained classification, we propose PEEB—an explainable and editable classifier to (1) express the class name into a set of text descriptors that describe the visual parts of the class; and (2) match the embeddings of the detected parts with 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 significantly outperforms CLIP, achieving a 10-fold increase in top-1 accuracy. Compared to part-based classifiers, PEEB not only achieves state-of-the-art (SOTA) accuracy in the supervised-learning setting—88.80% and 92.20% accuracy on CUB-200 🦜 and Dogs-120 🐕, respectively—but also the first to enable users to *edit* the text descriptors to form a new classifier without any re-training. Compared to concept bottleneck models, PEEB is also the SOTA in both zero-shot and supervised learning settings.

1 Introduction

Fine-grained classification (Wah et al., 2011; Van Horn et al., 2015) is a long-standing computer-vision challenge. Furthermore, it is also important to explain how SOTA classifiers make a decision, e.g., which bird traits make a model think a given bird is Painted Bunting? (Fig. 1)

First, 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 prototypes (Fig. 1b) or natural-language concepts (Fig. 1a). Yet, such prototypes are feature vectors and therefore not editable by users. Textual concepts are often compared against *entire*

^{*}Equal contribution

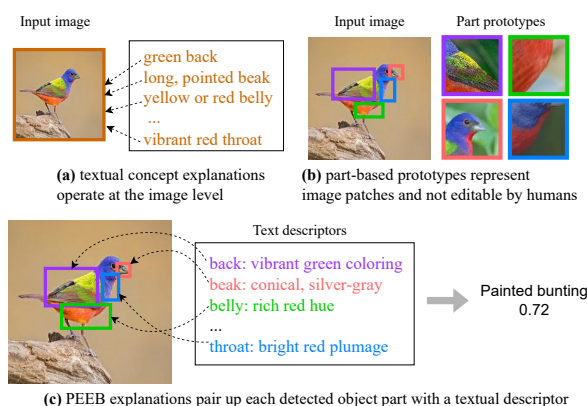


Figure 1: Existing explanations are either (a) textual but at the image level; or (b) part-level but not textual. Combining the best of both worlds, PEEB (c) first matches each detected object part to a text descriptor, then uses the part-level matching scores to classify the image.

image for classification and it is unknown what image details match a given descriptor (Menon and Vondrick, 2023; Yang et al., 2023). Third, most vision-language classifiers need the prompt to have a known `{class name}` (like a special code instead of an expressive, natural description) that matches the input image (Roth 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 species whose photos do not yet exist in the database.

To address the above four problems, we propose PEEB, a Part-based image classifier that is Explainable and Editable via a natural-language Bottleneck. PEEB classifies images by grounding the textual descriptor of object *parts* provided by humans or GPT-4 (no images needed) to detected parts in the image (Fig. 1c). 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 models (Radford

et al., 2021; Pratt et al., 2023; Menon and Vondrick, 2023) rely so heavily on the *known* class names that their accuracy drops significantly when the names are removed or replaced by less-common ones (Sec. 5.1).

For **birds** 🐦, we first define the parts of interest for identifying a bird. We take the 15 parts defined in CUB (Wah et al., 2011) and reduce them to 12 by merging similar parts, e.g. *left wing* and *right wing* are merged into *wings*. Using GPT-4 (OpenAI, 2023), we construct a *textual* descriptor to describe each bird part of every species (see Appendix C). Next, PEEB localizes the 12 bird parts in the image and computes their matching scores with corresponding text descriptors (Fig. 2). The sum of the 12 dot products between the paired visual and textual part embeddings would be the unnormalized distance (logits) between the input image and every class for classification (Fig. 3). For **dogs** 🐕, we use a similar procedure.

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 diversity (less than 1,500 species per dataset), impeding large-scale, vision-language, contrastive learning. Therefore, for our pre-training, we construct Bird-11K, an exceptionally large dataset of bird images, comprising ~290,000 images spanning across ~11,000 species—essentially, *all* known bird species on Earth (Sec. 3). Bird-11K is constructed from seven existing bird datasets and ~55,000 new images that we collect from the [Macaulay Library](#). Similarly, we build Dog-140, a large-scale dataset of 206K dog images. Our main findings are:¹

1. CLIP-based classifiers rely mostly on class names in the prompt: The CUB accuracy of M&V model (Menon and Vondrick, 2023) drops drastically from 53.78% to 5.89% and 5.95% after class names are removed or replaced by scientific names (Sec. 5.1).
2. Our pre-trained PEEB outperforms CLIP-based classifiers by +8 to +29 percentage points (pp) in bird classification across CUB-200, NABirds-555, and iNaturalist-1486 (Sec. 5.2).
3. PEEB allows defining new classes in text at test time (Fig. 2) without any further training. Besides explainability and editability, PEEB

outperforms *text concept-based* methods in both the generalized zero-shot (Sec. 5.3) and zero-shot setting (Sec. 5.4).

4. Compared with explainable CUB classifiers, PEEB scores an 88.80% top-1 accuracy, on par with the best CUB-200 classifiers (81–87% accuracy) that are trained via supervised learning and often *not* editable (Sec. 5.5).
5. PEEB is applicable to multiple domains. On Stanford Dogs-120, PEEB scores 92.20%, substantially outperforming explainable models and on-par with SOTA black-box models (Sec. 5.6).

2 Related Work

Ante- vs. post-hoc explanations It is common to build fine-grained classifiers based on CNNs (He et al., 2016) or ViTs (He et al., 2022a). Although high-performing, these models do not admit an *ante-hoc* 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). PEEB’s textual part-descriptors form an ante-hoc, natural-language explanation bottleneck that enables users to observe and edit the object attributes that contribute to each final prediction. By editing text descriptors, users can re-program the model without any further re-training (Fig. 2).

Prototypical Part Networks Like the explainable classifiers that learn part prototypes (Nauta et al., 2021; Donnelly et al., 2022; Nauta et al., 2022; Chen et al., 2019), PEEB also operates at the object-part level. However, there are two major differences. First, the textual part descriptors in PEEB are human-understandable and editable. In contrast, a part prototype (Chen et al., 2019) is not directly editable or interpretable to users, and often interpreted by showing the nearest training-set image patches. Second, PEEB predicts a *contextualized* embedding for each object part and its spatial information can be viewed by inputting to the Box MLP (see Fig. 3) for bounding-box visualization.

Text-based Concept Bottlenecks Like PEEB, (Chen et al., 2020; Zhu et al., 2018; Rao et al., 2023; Paz-Argaman et al., 2020) also match visual part embeddings to text embeddings. Yet, they (1) do not use CLIP and instead rely on TF-IDF text features; (2) require a trained bird-part detector to

¹Code & data: <https://github.com/anguyen8/peeb>

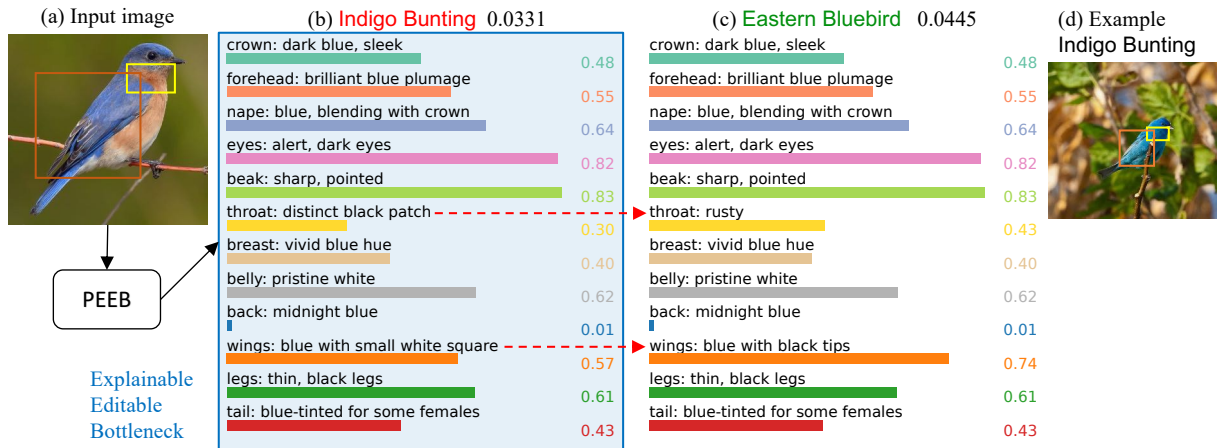


Figure 2: Given an input image (a) from an unseen class of Eastern Bluebird, PEEB misclassifies it into **Indigo Bunting** (b), a visually similar blue bird in CUB-200 (d). To add a new class for Eastern Bluebird to the 200-class list that PEEB considers when classifying, we clone the 12 textual descriptors 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** (softmax: 0.0445) out of 201 classes (c). That is, the dot product between the **wings** text descriptor and the same orange region increases from 0.57 to 0.74.

detect 7 bird parts. In contrast, PEEB relies on CLIP, which admits easy text editability, and OWL-ViT, which serves as an open-vocabulary object-part detector that generalizes to many domains.

Recent vision-language models (VLMs) claim to be interpretable as they use textual concepts in the prompt. Yet, some works that rely on class-wise differential captions (Esfandiarpour and Bach, 2023) or learned concept weights (Yang et al., 2023; Panousis et al., 2023; Oikarinen et al., 2023; Yuksekgonul et al., 2023) do *not* generalize to unseen classes. The most recent, similar work to PEEB might be LaBo (Yang et al., 2023), which; however, operates at the *image* level instead of patch level, and does *not* generalize to unseen classes.

Many CLIP-based classifiers (Han et al., 2023b; Pratt et al., 2023; Menon and Vondrick, 2023) rely heavily on having *seen* class names in the prompt and thus are neither explainable nor editable to users. Unlike CLIP-based models, PEEB reveals what image details are being used for classification by matching descriptors to corresponding visual object parts (e.g. a bird’s **beak** in Fig. 3).

Attribute-based Classifiers Attribute-Label Embedding (ALE) approaches (Akata et al., 2015; Yuksekgonul et al., 2023) employ a fixed set of attributes and train an attribute-to-label weight matrix for zero-shot classification. Several studies (Samuel et al., 2021; 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). Yet, in practice, ALE requires tabular data annotations for every new class in the dataset (e.g., 312 attributes per CUB species), editing the weight matrix, and model re-training. In contrast, to add an unseen class to PEEB, users would only need to describe its 12 bird parts in natural language.

3 Datasets

3.1 Test classification benchmarks

We test PEEB on three 🐦 bird classification datasets: CUB-200 (2011), NABirds-v1 of 555 classes (2015), and iNaturalist (2021) which has 1,486 bird classes. For 🐕 dog images, we test PEEB on Stanford Dogs-120 (2011).

3.2 Bird-11K dataset construction

We combine *labeled* images from 7 distinct datasets and an extra ~55K images (10,534 classes) from Cornell’s Macaulay Library, to form a unified **Bird-11K** dataset² (Appendix D.1) 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 to encompass almost all species on Earth. Since PEEB learns to match visual parts with textual descriptors, it requires that bird images be distinctly visible and

²We do not redistribute the published datasets but release a script for reconstructing Bird-11K on [Github](https://github.com).

sufficiently large for accurate part localization and matching (see Appendix E.3 for ablation studies). However, small and “hard-to-see” bird images in Bird-11K make the dataset noisy and the training complex. Thus, we harness OWL-ViT_{Large} (Minderer et al., 2022) to detect a bird in all images using the prompt “bird” and filter out images where the detected bird’s bounding box is smaller than 100×100 pixels. We find OWL-ViT’s bird detections to be fairly accurate—its mean Intersection over Union (IoU) between the predicted bird boxes and ground-truth boxes on CUB dataset is 0.91.

As class labels from different sources are either general (e.g. Cardinal) or fine-grained (e.g. Yellow vs. Northern Cardinal), we retain only the fine-grained species for more diverse training and exclude all general classes to avoid label ambiguity. Following these filtering 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 (Appendix C). These generated descriptors (see Fig. 4) may not be 100% accurate but discriminative enough to help GPT-4V reach 69.40% accuracy on the CUB-200 test set (Table 3). That is, in the same prompt, we feed each test image x along with the 200 CUB classes’ part-based descriptors and ask GPT-4V to select a matching class label for x (details in Appendix F.1).

3.3 Dataset splits for contrastive pre-training

There are two common settings in the zero-shot learning literature—standard zero-shot (ZSL) and generalized zero-shot (GZSL).

ZSL is a stricter setup where a model is only tested on the *classes* unseen during any prior training. We ensure test-set classes from datasets (e.g., CUB-200 or NABirds-555) are not seen during pre-training. For example, to test on CUB under ZSL, we exclude all 200 CUB classes and their images from our pre-training on Bird-11K.

Following the ZSL literature, we use the 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 finetune only on the corresponding training set given by a ZSL split.

GZSL is closer to the real-world setup where models are tested on both seen & unseen classes. CLIP’s “zero-shot” tests technically fall under

GZSL as its Internet-scale training set might actually have images from the test classes. To test PEEB under GZSL, we exclude the *test* sets of CUB, NABirds, and iNaturalist, and directly evaluate the Bird-11K-pretrained models without further finetuning.

4 Method

4.1 Backbone: OWL-ViT object-part detector

OWL-ViT is an open-vocabulary detector that detects objects and parts in an image given a text prompt, even if the model is not explicitly finetuned to detect those concepts. OWL-ViT consists of four networks (Fig. 3): (1) a ViT-based image encoder, (2) an architecturally identical text encoder, (3) a bounding-box regression head called Box MLP, and (4) a Linear Projection. Box MLP is a three-layer Multilayer Perceptron (MLP) with GELU activations (Hendrycks and Gimpel, 2016) after each of the first two layers. Linear Projection maps the visual and text embeddings to the same space (see Fig. 1 in (Minderer et al., 2022)).

4.2 PEEB classifier

Architecture PEEB (Fig. 3) has five networks: an image encoder, a text encoder, a Linear Projection, a Part MLP, and a Box MLP.

We introduce **Part MLP** to map the visual and textual part embeddings to the same space for computing dot products (logits) for classification (\rightarrow in Fig. 3). This design allows PEEB to easily extend the number of classes without any re-training. Except for Part MLP, all components are adopted from the OWL-ViT framework. Details of all components are in Appendix A.

Inference Given an input image, we first use the 12 generic part names to select the visual part embeddings based on cosine similarity. These selected visual part embeddings are then simultaneously fed into both Part MLP and Box MLP.

Box MLP predicts the bounding box from each part embedding. We compute a dot product to measure the similarity between each embedding output from Part MLP and a corresponding part-descriptor embedding. For classification, a class logit is the sum of the 12 dot products, which essentially computes the similarity between the 12 parts in the image and the 12 text descriptors of each class.

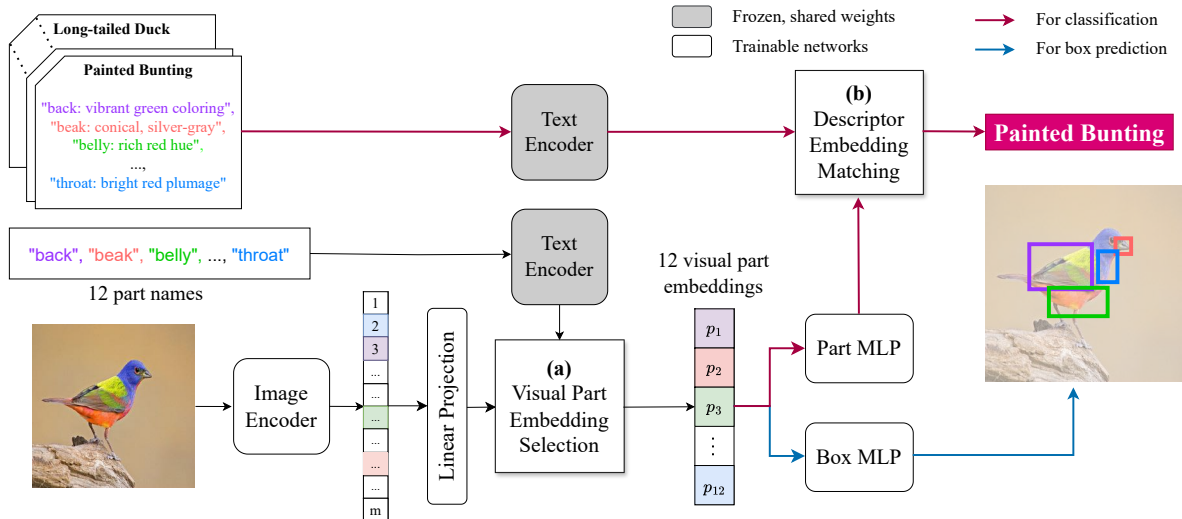


Figure 3: During inference, 12 visual part embeddings with the highest cosine similarity with encoded part names are selected (a). These visual part embeddings are then mapped (\rightarrow) to bounding boxes via Box MLP. Simultaneously, the same embeddings are forwarded to the Part MLP and its outputs are then matched (b) with textual part descriptors to make classification predictions (\rightarrow). Fig. A1 shows a more detailed view of the same process.

4.3 Training strategy

Trainable networks In preliminary experiments, we find training only Part MLP (while keeping all other networks frozen) to result in poor accuracy. Therefore, we train Part MLP from scratch and also finetune the image encoder, Linear Projection, and Box MLP. We finetune all OWL-ViT components from their original weights. In contrast, our proposed Part MLP starts from random weights. Our training has two phases: (a) 2-stage **pre-training** on the large-scale Bird-11K dataset and (b) **finetuning** on downstream tasks. More hyperparameter details are in Appendix A.8.

Objectives We aim to train PEEB to classify images well while maintaining the ability to detect object parts. This translates into **three training objectives**: (1) Train the Part MLP contrastively using a symmetric cross-entropy (SCE) loss (Radford et al., 2021) to maximize the similarity between region-text pairs while minimizing the similarity for negative pairs; (2) Train the Linear Projection using a SCE loss to mimic OWL-ViT’s behaviors (i.e. the similarity matrix) for part selection; and (3) Train Box MLP to predict bounding boxes with DETR losses (Zheng et al., 2021) i.e. a linear combination of ℓ_1 corner-to-corner distance loss and GIoU loss (Rezatofighi et al., 2019).

All three losses are described in Appendix A.10.

A challenge when jointly minimizing all three losses above is that PEEB’s validation loss im-

proves significantly slowly perhaps because of some tension between the two SCE losses and the DETR detection loss. To overcome this challenge, we split the pre-training phase into two stages: (1) first, train the image encoder and Part MLP for classification using the SCE loss; then (2) train the Linear Projection and Box MLP using the 2nd and 3rd loss so they can adapt their weights to the updated image encoder. We always keep the **text encoder** frozen since we want to preserve its generalizability to the descriptors of unseen objects.

4.3.1 2-stage pre-training on Bird-11K

Stage 1: Contrastive learning The image encoder and Part MLP are jointly trained using a SCE loss, which allows PEEB to learn to map the visual parts to corresponding text descriptors. In this stage, we use a pre-trained OWL-ViT_{Large} to *select* 12 part embeddings per input image (i.e., teacher forcing) to ensure the *selection* of part embeddings is meaningful and consistent while the embeddings themselves are updating (see Fig. A2).

Stage 2: Learning to detect from a teacher After the image encoder is modified in Stage 1, we then train the Linear Projection and Box MLP jointly. We use the OWL-ViT_{Large} as the teacher to train both components. Using SCE loss, we train the Linear Projection such that the similarity matrix between the part-names and visual parts matches those of the teacher (Fig. A3, 1a–c, 2a–c). Given the absence of human-annotated boxes for object



parts, we train Box MLP to predict the same boxes as the predictions by OWL-ViT_{Large} using DETR losses (Fig. A3, 2d). In this Stage 2, the image encoder is frozen while Part MLP is not involved.

After 2-stage training, PEEB can perform zero-shot classification and generate explanations.

4.3.2 Finetuning on classification tasks

We finetune the pre-trained PEEB on downstream tasks (CUB, NABirds and iNaturalist for birds and Dogs-120 for dogs) to further improve its accuracy. In this phase, to adapt to a downstream task, all components except the text encoder are trained jointly and the loss for Part MLP is changed from SCE (contrastive) to CE (classification) while the other two losses (DETR) are kept intact.

5 Experiments & Results

We conduct extensive experiments to evaluate PEEB on multiple  bird datasets (CUB, NABirds, iNaturalist) and on GZSL (Secs. 5.1 and 5.3), ZSL (Sec. 5.4) and also supervised learning settings. We also find PEEB to perform well on  dog image classification on Dogs-120 (Sec. 5.6).

5.1 CLIP-based classifiers rely mostly on {class names} (not descriptors)

M&V show that inserting extra GPT-3-generated descriptors into CLIP’s prompts increases its accuracy on many tasks (Menon and Vondrick, 2023). Yet, it is unknown how important these expressive descriptors are compared to the class names. To answer this question, we conduct two experiments on all three models: CLIP, M&V, and our PEEB.

Experiment 1 We evaluate the role of expressive descriptors to CLIP-based models and to PEEB by measuring the drop in CUB-200 accuracy of each model when the descriptors are randomized.

For M&V and PEEB, we randomize the descriptors by swapping each descriptor with another from an arbitrary class (examples in Fig. 4).

Experiment 2 We test the dependence of models on class names by measuring the accuracy drop when they are replaced by scientific names (e.g., Painted Bunting \rightarrow *Passerina ciris*) on CUB, NABirds, and iNaturalist.

Results When random descriptors are used, M&V accuracy drops marginally by -0.9 pp (Table 1; $53.70\% \rightarrow 52.88\%$), showing that descriptors actually play a minimal role in model predictions. Instead, CLIP and M&V mostly rely on class names (e.g., $53.78\% \rightarrow 7.66\%$; Table 2)—their accuracy

drops drastically when class names are replaced by scientific names, which are less common.

In contrast, the expressive part descriptors play a major role in PEEB whose accuracy decreases significantly to near random-chance ($64.33\% \rightarrow 0.88\%$; Table 1) when the descriptors are randomized. Indeed, in PEEB, the textual descriptors serve as editable and interpretable model parameters that can be refined and extended by humans to account for new classes (Fig. 2).

Table 1: Top-1 test accuracy (%) on CUB-200 when using original, correct (a) vs. randomized, wrong descriptors (b). See Fig. 4 for an example of the descriptors.

| | CLIP (2021) | M&V (2023) | PEEB | |
|----------------------------|-------------|------------|------|--------------|
| With class names | ✓ | ✓ | ✗ | |
| (a) Original descriptors | 52.02 | 53.78 | 5.89 | 64.33 |
| (b) Randomized descriptors | n/a | 52.88 | 0.59 | 0.88 |

Table 2: In the **GZSL** setting, PEEB outperforms CLIP and M&V by a large margin, from $+8$ to $+29$ pp in top-1 accuracy (see Sec. 5.3). PEEB is also $\sim 10\times$ better than the other two models when class names are replaced by **scientific names**. As PEEB does not use class names, its accuracy remains unchanged when class names are changed into the scientific ones.

| Acc (%) | CUB-200 | NABirds-555 | iNaturalist-1486 |
|-------------|----------------------|----------------------|----------------------|
| CLIP (2021) | 52.02 (5.95) | 39.35 (4.73) | 16.36 (2.03) |
| M&V (2023) | 53.78 (7.66) | 41.01 (6.27) | 17.57 (2.87) |
| PEEB (ours) | 64.33 (64.33) | 69.03 (69.03) | 25.74 (25.74) |

5.2 Pre-trained PEEB outperforms CLIP-based classifiers in GZSL

The dependence on class names (Sec. 5.1) suggests that CLIP was exposed to these names during training. Thus, for a fair comparison, we compare PEEB with CLIP-based classifiers in the GZSL setting.

Experiment We train PEEB on Bird-11K using the two-stage pre-training (described in Sec. 4.3.1), and then test it on CUB, NABirds, and iNaturalist without any finetuning. That is, PEEB’s contrastive pre-training is at the part level and therefore the model has not seen the species labels of images.

Results PEEB outperforms both CLIP and M&V on all three datasets by huge margins of around $+10$, $+28$, and $+8$ pp on CUB-200, NABirds-555 and iNaturalist-1486, respectively (see Table 2).



Figure 4: With original descriptors, M&V (Menon and Vondrick, 2023) correctly classifies the input image into **Blue Jay** (a). Yet, interestingly, when **randomly** swapping the descriptors of this class with those of other classes (b), M&V’s top-1 prediction remains unchanged, suggesting that the class names in the prompt (e.g., “A photo of {class name}”) have the most influence over the prediction (not the expressive descriptors). In contrast, PEEB changes its top-1 prediction from **Blue Jay** (c) to **Least Tern** (d) when the descriptors are **randomized**.

5.3 PEEB is superior to text descriptor-based classifiers in GZSL on CUB-200

The advent of CLIP (2021) by OpenAI enabled a class of image classifiers that match the input image with pre-defined textual prompts that may include class names or descriptors of the classes. Yet, in contrast to PEEB, these descriptors often describe the entire image and are also matched (via dot product) with the entire image instead of image regions. Here, we compare PEEB with these methods in the GZSL setting on CUB-200.

Experiment We repeat the same experiments in Sec. 5.2. As these bird classifiers (listed in Table 3) were reported on CUB only (not NABirds or iNaturalist), our comparison is on CUB.

Results PEEB exhibits superior GZSL performance, outperforming recent text concept-based approaches by +3 to +10 pp (Table 3b). Compared to prior methods, PEEB is the only one to detect *visual* object parts and match them with text descriptors. Furthermore, attribute-based classifiers, e.g., (Yuksekgonul et al., 2023) require re-training to adapt to new classes or datasets (e.g., NABirds or iNaturalist) in the same domain. In contrast, to apply PEEB to NABirds or a new class, no training is required—it is necessary to only edit its text descriptors (see Fig. 2). Interestingly, PEEB is 2nd-best model, only after GPT-4V (64.33% vs.

69.40%), which is given the same textual part descriptors as PEEB for all 200 CUB classes and asked to select a matching class for each image.

Table 3: PEEB achieves SOTA CUB-200 accuracy among the **text descriptor-based** classifiers in GZSL. * *1-shot learning*. † *k-means with k = 32*.

| Method | Acc (%) | {c} | Textual descriptors |
|--|--------------------|-----|-------------------------------|
| (a) Vision-language models with class names {c} in the prompt | | | |
| CLIP (2021) | 52.02 | ✓ | Image-level |
| M&V (2023) | 53.78 | ✓ | Image-level |
| FuDD (2023) | 54.30 | ✓ | Image-level |
| Han et al. (2023b) | 56.13 | ✓ | Image-level |
| (b) Vision-language models with text bottlenecks and no class names {c} | | | |
| LaBo (2023) | 54.19 [†] | ✗ | Image-level |
| Yan et al. (2023) | 60.27* | ✗ | Image-level, attribute-based |
| PEEB (ours) | 64.33 | ✗ | Part-level |
| GPT-4V (2023) | 69.40 | ✓ | Part-level |
| (c) Concept-Bottleneck Models with attribute-based, non-textual bottlenecks | | | |
| CBM (2020) | 62.90 | ✗ | Attribute-based, tabular data |
| PCBM (2023) | 61.00 | ✗ | Attribute-based, tabular data |

5.4 PEEB generalizes to traditional ZSL

Since PEEB outperforms modern vision-language models in GZSL (Sec. 5.3), we are motivated to further compare PEEB with SOTA approaches in the traditional ZSL setting (where the test classes are excluded from all prior training).

Experiment We evaluate PEEB on **two common ZSL splits**: (a) the CUB split (Akata et al., 2015); and (b) the Super-Category-Similar/Exclusive (SC-S/SCE) splits (Elhoseiny et al., 2017) on CUB and

NABirds. The SCS (Easy) and SCE (Hard) splits are designed to test two generalization levels (generalizing to close vs. distant unseen species).

Aligned with ZSL conventions, we exclude all species that exist in CUB or NABirds from the pre-training and then finetune PEEB using the train/test splits by Akata et al. and Elhoseiny et al.. We randomly take $\sim 10\%$ of the training set as the validation set and choose the checkpoints based on the lowest validation loss.

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 | Mean | Seen | Unseen | Mean |
| (a) Data split by Akata et al. (2015) | | | | | | |
| CLORE _{CLIP} (2023a) | 65.80 | 39.10 | 49.05 | n/a | | |
| PEEB (ours) | 80.78 | 41.74 | 55.04 | | | |
| (b) SCS/SCE splits by Elhoseiny et al. (2017) | | | | | | |
| | SCS (Easy) | SCE (Hard) | Mean | SCS (Easy) | SCE (Hard) | Mean |
| 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 |

Results By a large margin, PEEB outperforms CLORE_{CLIP}, a SOTA CUB method in the (2015) split, on both seen and unseen classes (Table 4a). On the (2017) splits, PEEB is the SOTA in the Hard set on both CUB and NABirds datasets (Table 4b). That is, PEEB is better in generalizing to distant, unseen classes. This may be because PEEB decomposes both the image and the text descriptors into part-level features, which can re-combine to match an arbitrary unseen class (as illustrated in Fig. 2).

Interestingly, on both CUB and NABirds, PEEB is competitive but *not* SOTA on the Easy sets (Table 4b; Easy)—those classes that are close to the training-set classes and thus considered easier to identify. Overall, considering the harmonic mean over both Easy and Hard accuracy scores, PEEB is the SOTA on both CUB and NABirds.

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


After finding that PEEB performs well in both GZSL (Sec. 5.3) and ZSL settings (Sec. 5.4), here we test finetuning the pre-trained PEEB on CUB-

200. That is, we compare PEEB against SOTA explainable classifiers in the supervised learning setting to gain insights into our method’s adaptability to downstream tasks.

Experiment To understand the impact of pre-training and image resolution, we test finetuning three different PEEB variants: (1) PEEB initialized from OWL-ViT_{B/32} **without pre-training** on Bird-11K; (2) PEEB initialized from OWL-ViT_{B/32} with pre-training (described in Sec. 5.2); and (3) PEEB initialized from OWL-ViT_{B/16} with pre-training. We take each PEEB model and finetune *all* components on CUB-200, for 30 epochs with a batch size of 30, a learning rate of 2×10^{-5} . Detailed hyperparameters are in Table A2.

Results Without pre-training, PEEB reaches 77.80% top-1 accuracy on CUB-200. Yet, first pre-training on Bird-11K and then finetuning on CUB yields 86.73%, the best among all *explainable* classifiers (Table 5b–c). Besides, pre-training PEEB from the higher-resolution OWL-ViT_{B/16} results in a further gain of +2.07 (86.73% \rightarrow 88.80%), which is intuitive since fine-grained classification is known to benefit from higher resolutions.

For a complete assessment, we compare and find PEEB to underperform SOTA standard, black-box classifiers by a few points (Table 5a).

Table 5: PEEB is a state-of-the-art, explainable CUB-200  classifiers in the supervised learning.

| Methods | Model size | Backbone | Acc (%) |
|--|------------|-------------------------|--------------|
| (a) SOTA black-box classifiers | | | |
| Base (ViT) (2021) | 22M | DeiT-S (2021) | 84.28 |
| ViT-Net (2022a) | 26M | DeiT-S | 90.10 |
| (b) Concept-bottleneck classifiers | | | |
| CBM (Koh et al., 2020) | 11M | ResNet-18 | 80.10 |
| CPM (Panousis et al., 2023) | 155M | ViT-B/16 | 72.00 |
| CDM (Oikarinen et al., 2023) | 155M | ViT-B/16 | 74.31 |
| LaBo (Yang et al., 2023) | 427M | ViT-L/14 | 81.90 |
| (c) Part-based, explainable classifiers | | | |
| ProtoPNet (2019) | 22M | DeiT-S | 84.04 |
| ProtoTree (2021) | 92M | ResNet-50 | 82.20 |
| TesNet (2021) | 79M | DenseNet-121 | 84.80 |
| Deformable ProtoPNet (2022) | 23M | ResNet-50 | 86.40 |
| ProtoPFormer (2022) | 22M | DeiT-S | 84.85 |
| PEEB (ours) | 155M | | |
| pre-training + finetuning <i>only</i> | 155M | OWL-ViT _{B/32} | 77.80 |
| pre-training + finetuning | 155M | OWL-ViT _{B/32} | 86.73 |
| pre-training + finetuning | 155M | OWL-ViT _{B/16} | 88.80 |

5.6 Applying PEEB to dog identification

We have found that our pre-training dataset construction and PEEB performs well for bird identification. By design, our method is *not* specific to birds but is instead applicable to any fine-grained

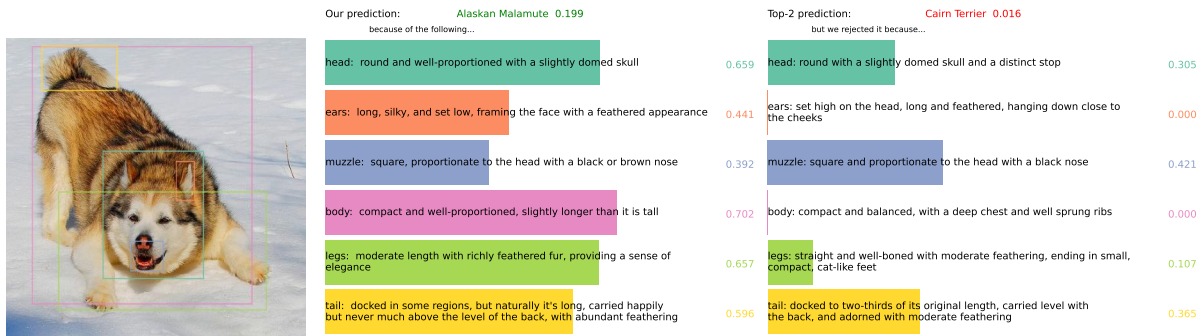


Figure 5: PEEB classifies this Dogs-120 image into **Alaskan Malamute** (softmax: **0.199**) due to the matching between the image regions and associated textual part descriptors. In contrast, the explanation shows that the input image is not classified into **Cairn Terrier** mostly because its ears and body regions do *not* match the text descriptors, i.e., dot products are **0.000** and **0.000**, respectively. See Appendix G for more qualitative examples.

classification domains assuming that the object is decomposable into parts. Here, we show that our method performs well on dog image classification as well.

Pre-training dataset construction First, we define a set of six dog parts that humans use to identify dog species. We use all 4 dog parts defined by PartImageNet (He et al., 2022b)—*head*, *body*, *legs*, and *tail*—and two more parts—*muzzle* and *ears*—based on our manual image examination.

We combine ImageNet-21K and Stanford Dogs-120 into Dog-140, our large-scale pre-training dataset spanning 140 dog species (details in appendix D.2). For each class, we prompt GPT-4 to get the descriptors for 6 parts. For each image in Dog-140, we run OWL-ViT_{Large} to detect the corresponding boxes for 6 pre-defined parts.

Experiment Following the supervised learning experiment in Sec. 5.5, we first pre-train PEEB (initialized from OWL-ViT_{B/32}) on Dog-140 and then further finetune it on Dogs-120.

Results Finetuning PEEB on Dogs-120 from OWL-ViT_{B/32} without pre-training on Dog-140 results in a 74.17% top-1 accuracy on Dogs-120 (Table 6b). In contrast, pre-training on Dog-140 only without finetuning results in much better Dogs-120 accuracy of 87.38%. That is, our contrastive pre-training helps model generalize (in a GZSL setting) while directly finetuning on Dogs-120 perhaps yields an overfitting model. Yet, pre-training and then finetuning reaches the best supervised learning accuracy of 92.20%, which is SOTA among all explainable models reported on Dogs-120.

Besides, PEEB offers novel, editable image-text grounding explanations (see Fig. 5).

Table 6: In the **supervised** learning setting, PEEB is the state-of-the-art explainable, Stanford Dogs-120 🐕 classifiers and competitive w.r.t. SOTA black-box models.

| Methods | Model size Backbone | Acc (%) |
|---------------------------------------|------------------------------|--------------|
| (a) SOTA black-box classifiers | | |
| TransFG (2022a) | 86M ViT-B/16 | 92.30 |
| ViT-Net (2022b) | 86M DeiT-B | 93.60 |
| SR-GNN (2022) | 32M Xception | 97.00 |
| (b) Explainable methods | | |
| FCAN (2016) | 50M ResNet-50 | 84.20 |
| RA-CNN (2017) | 144M VGG-19 | 87.30 |
| ProtoPNet (2019) | 22M DeiT-S | 77.30 |
| Deformable ProtoPNet (2022) | 23M ResNet-50 | 86.50 |
| PEEB (ours) | 155M | |
| pre-training + finetuning <i>only</i> | 155M OWL-ViT _{B/32} | 74.17 |
| pre-training + finetuning | 155M OWL-ViT _{B/32} | 87.37 |
| pre-training + finetuning | 155M OWL-ViT _{B/16} | 92.20 |

6 Discussion and Conclusion

We introduce PEEB, a unique, novel explainable classifier due to its editability (Fig. 2) and operation at the part level on both image and text sides. The part-level operation makes PEEB applicable to fine-grained classification. Yet, it is also interesting to extend PEEB into an object-level model for multi-domain tasks like ImageNet or VQA.

Besides enabling users to edit PEEB’s text descriptors to re-program PEEB, it might also be promising to let users edit the bounding boxes while working with PEEB to improve the human-AI team accuracy (Nguyen et al., 2024). On object detection, PEEB’s Box MLP performs on-par with OWL-ViT_{B/32} based on quantitative (Appendix E.7) and qualitative results (Appendix G).

Finally, we contribute to the broader research community by curating the Bird-11K and Dog-140 datasets and showing that it is possible to leverage them for large-scale training.

7 Limitations

Text encoder may not fully comprehend the bird descriptors Our CLIP text encoder, pre-trained on an Internet-scale image-text dataset (Radford et al., 2021), may not fully capture the intricate details specific to birds. Furthermore, the CLIP text encoder is known to suffer from the *binding* problem and do not understand some logical operators such as “and”, “or”, or negation. PEEB accuracy depends heavily on the quality of the text encoder.

Assumption that object parts mostly visible PEEB operates based on the assumption that most if not all of the object parts are visible in the image. In cases where a part is missing or occluded, the model may still assign a non-zero similarity score (i.e. a non-zero dot product between the image-part embedding and its associated text descriptor), which makes it harder to separate classes. It might be beneficial to incorporate extra training samples and specifically encourages PEEB to assign *zero* image-text similarity score to the missing or occluded parts.

Hallucinations in GPT-4 descriptors The accuracy of PEEB is directly governed by the accuracy of descriptors, which are currently generated by GPT-4. Yet, our manual assessment over 20 bird classes reveals that, on average, 45% of these descriptors do not accurately reflect the birds’ features (Appendix F.2). Also, we observe that revising certain descriptors in the CUB dataset led to a **significant** improvement of +10 points in classification accuracy for those classes (Appendix F.3). This primitive observation suggests that PEEB can be further improved if trained with more accurate, human-labeled descriptors.

Acknowledgement

We are grateful to Cornell’s Macaulay Library for providing us with ~55K images categorized into 10,534 bird species for the large-scale pre-training. We thank Pooyan Rahmzadehgervi, Giang Nguyen, Tin Nguyen, and Hung Huy Nguyen from Auburn University for their helpful feedback on our early results. We also thank Phat Nguyen for her valuable support and feedback. AN is supported by NaphCare Foundations, Adobe gifts, and NSF grant no. 2145767.

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 hyper-parameters. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8673–8683.
- Asish Bera, Zachary Wharton, Yonghuai Liu, Nik Bessis, and Ardhendu Behera. 2022. Sr-gnn: Spatial relation-aware graph neural network for fine-grained image categorization. *IEEE Transactions on Image Processing*, 31:6017–6031.
- 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.
- 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.
- 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 Esfandiarpour and Stephen H Bach. 2023. Follow-up differential descriptions: Language models resolve ambiguities for image classification. *arXiv preprint arXiv:2311.07593*.
- Jianlong Fu, Heliang Zheng, and Tao Mei. 2017. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Fédération Cynologique Internationale (FCI). 2023. [Nomenclature of the breeds recognised by the fci](#). Accessed: 2014-02-25.
- 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. 2023a. Zero-shot classification by logical reasoning on natural language explanations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8967–8981, Toronto, Canada. Association for Computational Linguistics.
- Songhao Han, Le Zhuo, Yue Liao, and Si Liu. 2023b. 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. 2022a. Transfg: A transformer architecture for fine-grained recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 852–860.
- Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, Qihang Yu, and Alan Yuille. 2022b. Partimagenet: A large, high-quality dataset of parts. In *European Conference on Computer Vision*, pages 128–145. Springer.
- 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*.
- 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.
- Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2. Citeseer.

- Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. 2022a. Vit-net: Interpretable vision transformers with neural tree decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR.
- Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. 2022b. ViT-NeT: Interpretable vision transformers with neural tree decoder. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 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*.
- Xiao Liu, Tian Xia, Jiang Wang, Yi Yang, Feng Zhou, and Yuanqing Lin. 2016. Fully convolutional attention networks for fine-grained recognition. *arXiv: Computer Vision and Pattern Recognition*.
- Sachit Menon and Carl Vondrick. 2023. Visual classification via description from large language models. In *The Eleventh International Conference on Learning Representations*.
- Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xi-aohua 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, Sunnie SY Kim, and Anh Nguyen. 2024. Allowing humans to interactively guide machines where to look does not always improve a human-ai team’s classification accuracy. *arXiv preprint arXiv:2404.05238*.
- 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, Reut Tsarfaty, Gal Chechik, and Yuval Atzmon. 2020. ZEST: Zero-shot learning from text descriptions using textual similarity and visual summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 569–579, Online. Association for Computational Linguistics.
- Gerald Piosenka. 2022. [Birds 525 - species image classification](#).
- Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. 2023. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15691–15701.
- 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 Reza Tofighi, 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)*.
- Karsten Roth, Jae Myung Kim, A Koepke, Oriol Vinyals, Cordelia Schmid, and Zeynep Akata. 2023. Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15746–15757.
- 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](https://media.ebird.org/). 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 Oisín 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](https://www.eecr.org/projects/2011-2012/10-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](https://arxiv.org/abs/2010.11929). 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](https://arxiv.org/abs/1905.07421). *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](https://arxiv.org/abs/1011.4280). 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. Prototformer: 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 Yuksekogonul, Maggie Wang, and James Zou. 2023. [Post-hoc concept bottleneck models](https://arxiv.org/abs/2305.18276). In *The Eleventh International Conference on Learning Representations*.
- Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Xiaogang Wang, Hongsheng Li, and Hao Dong. 2021. End-to-end object detection with adaptive clustering transformer.
- 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.

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

A Architecture details

A.1 Image encoder and text encoder

We employ the image encoder and text encoder from 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 (see [detail implementation here](#)), which is composed of a learnable logit scale, a learnable logit shift, and an Exponential Linear Unit (ELU) activation function. These processed image embeddings then have the same dimension as the text embeddings. For OWL-ViT_{B/32}, the image embeddings are projected from 768 to 512. 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 **Part MLP** to enable part-based classification (see [implementation detail here](#)). It comprises a three-layer MLP with GELU activations (Hendrycks and Gimpel, 2016). **Part MLP** takes in the selected part embeddings (i.e. output of step 1 in Fig. A1) and outputs a vector of size \mathbb{R}^d for each part, where d is the dimension of descriptor embeddings (for OWL-ViT_{B/32}, the input dimension is 768, and $d = 512$). Part MLP is trained to map the selected part embeddings to the same dimensional space with descriptor embeddings to compute final logits for classification.

A.4 Box MLP

The **Box MLP** retained from OWL-ViT consists of a three-layer MLP (see [here for implementation detail](#)). 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.

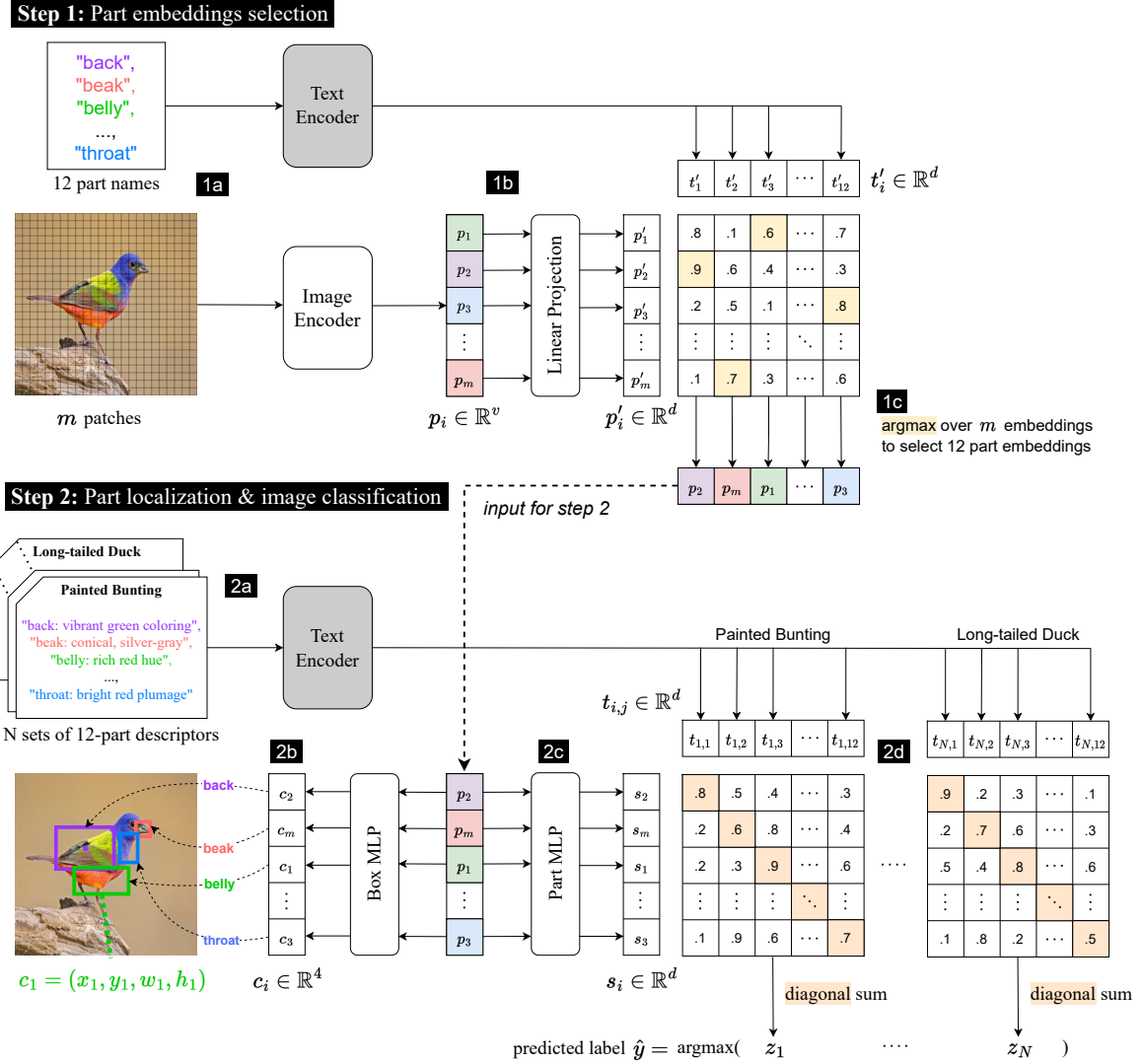


Figure A1: During the test time using PEEB, we perform 2 steps.

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 a 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} .

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 implemen-

tation (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 finetuned 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 finetuning takes 2 to 4 hours with one single GPU.

A.10 Pre-training and Finetuning objectives

As discussed in Sec. 4.3, we have three objectives during the **Pre-training** phase:

1. Pre-training **Stage 1:** (Fig. A2) During the pre-training stage one, we contrastively pre-train the model to maximize the similarity between related part-descriptor pairs while minimizing the unrelated pairs using *symmetric cross-entropy (SCE)* loss (Radford et al., 2021).
2. Pre-training **Stage 2:** (Fig. A3) We try to remove the dependence on the OWL-ViT_{Large} teacher model by training PEEB to mimic OWL-ViT_{Large}'s box predictions using the SCE loss.
3. Pre-training **Stage 2:** (Fig. A3) We simultaneously train PEEB to improve box prediction with DERT losses (Zheng et al., 2021).

During the **Finetuning** phase where we finetune on a downstream task (e.g. Dogs-120 or CUB-200), we also employ the same three losses. However, we change the first loss from SCE into CE since on the downstream classification task, the classifier is tasked with selecting one class that matches the single input image from a set of classes.

A.10.1 Pre-training stage one: Symmetric cross-entropy loss for contrastive pre-training

We first define the embeddings derived from the image and text encoders:

$$I'_f = \text{image_encoder}(I) \quad (1)$$

where I is the input image, and $I'_f \in \mathbb{R}^{n \times d_i}$ is output image embeddings. Here, d_i is the feature dimension of the image encoder. The text embedding T_f is given by

$$T_f = \text{text_encoder}(T) \quad (2)$$

where T represents the text input, and $T_f \in \mathbb{R}^{m \times d_t}$. In this case, d_t is the feature dimension of the text encoder. The image embedding I'_f is then transformed by *Part MLP* layer (Fig. A1, 1b) to align its dimensions with the text embedding. This transformation is denoted as

$$I_f = \text{Part MLP}(I'_f) \quad (3)$$

where $I_f \in \mathbb{R}^{n \times d_t}$. The similarity matrix S between the image and text embeddings is computed as the dot product of I_f and the transpose of T_f , expressed as

$$S = I_f \cdot T_f^\top \quad (4)$$

where $S \in \mathbb{R}^{n \times m}$. The image logits (S^i) and text logits (S^t) are then defined as

$$S^i = \text{softmax}(S, \text{axis}=0) \quad (5)$$

and

$$S^t = \text{softmax}(S, \text{axis}=1) \quad (6)$$

Next, we define the symmetric cross-entropy loss for the multi-modal embeddings.

$$L_{sce} = - \frac{(\sum_i y_i^i \log(S_i^i) + \sum_m y_m^t \log(S_m^t))}{2} \quad (7)$$

where $y^i \in \mathbb{R}^n$ is the label for image and $y^t \in \mathbb{R}^m$ is the label for text.

A.10.2 Pre-training stage 2: Symmetric cross-entropy loss to mimic the teacher OWL-ViT_{Large} detector

To mimic the object detection capability of the OWL-ViT_{Large} teacher, we train PEEB to mimic the image-text similarity prediction between image embedding and textual part-name embeddings (as shown in Fig. A1, 1c). We first binary the teacher logits and consider it as the ground truth label. Then, apply the same symmetric cross-entropy loss as described in eq. (7) with two minor differences: (1) The text input is part names rather than descriptions. (2) The *Part MLP* is replaced by *Linear Projection* (Fig. A1, 2c).

A.10.3 Pre-training stage 2: DETR losses to mimic the teacher OWL-ViT_{Large} detector

DETR losses are designed to optimize the box detection performance. We employ partial losses in our training for box predictions. Specifically, we employ ℓ_1 corner-to-corner distance loss and GIoU loss. For the selected embeddings, we predict the boxes with *Box MLP* (Fig. A1, 2b)

$$B = \text{Box MLP}(I'_f) \quad (8)$$

where I'_f is the image selected image embeddings from eq. (1), $B \in \mathbb{R}^{n \times 4}$ is the predicted bounding boxes. Let $Y^{GT} \in \mathbb{R}^{n \times 4}$ be the ground truth boxes. The ℓ_1 corner-to-corner distance loss is defined as

$$L_{\ell_1} = \sum_i \|Y_i^{GT} - B_i\| \quad (9)$$

The GIoU loss L_{GIoU} is defined in Appendix A.10.3, and the total box loss is defined as

$$L_{Box} = \frac{L_{\ell_1} + L_{GIoU}}{2} \quad (10)$$

Algorithm 1 Generalized Intersection over Union

Require: Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

Ensure: $GIoU$

- 1: For A and B , find the smallest enclosing convex object C , where $C \subseteq \mathbb{S} \in \mathbb{R}^n$
 - 2: $IoU = \frac{|A \cap B|}{|A \cup B|}$
 - 3: $GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$
-

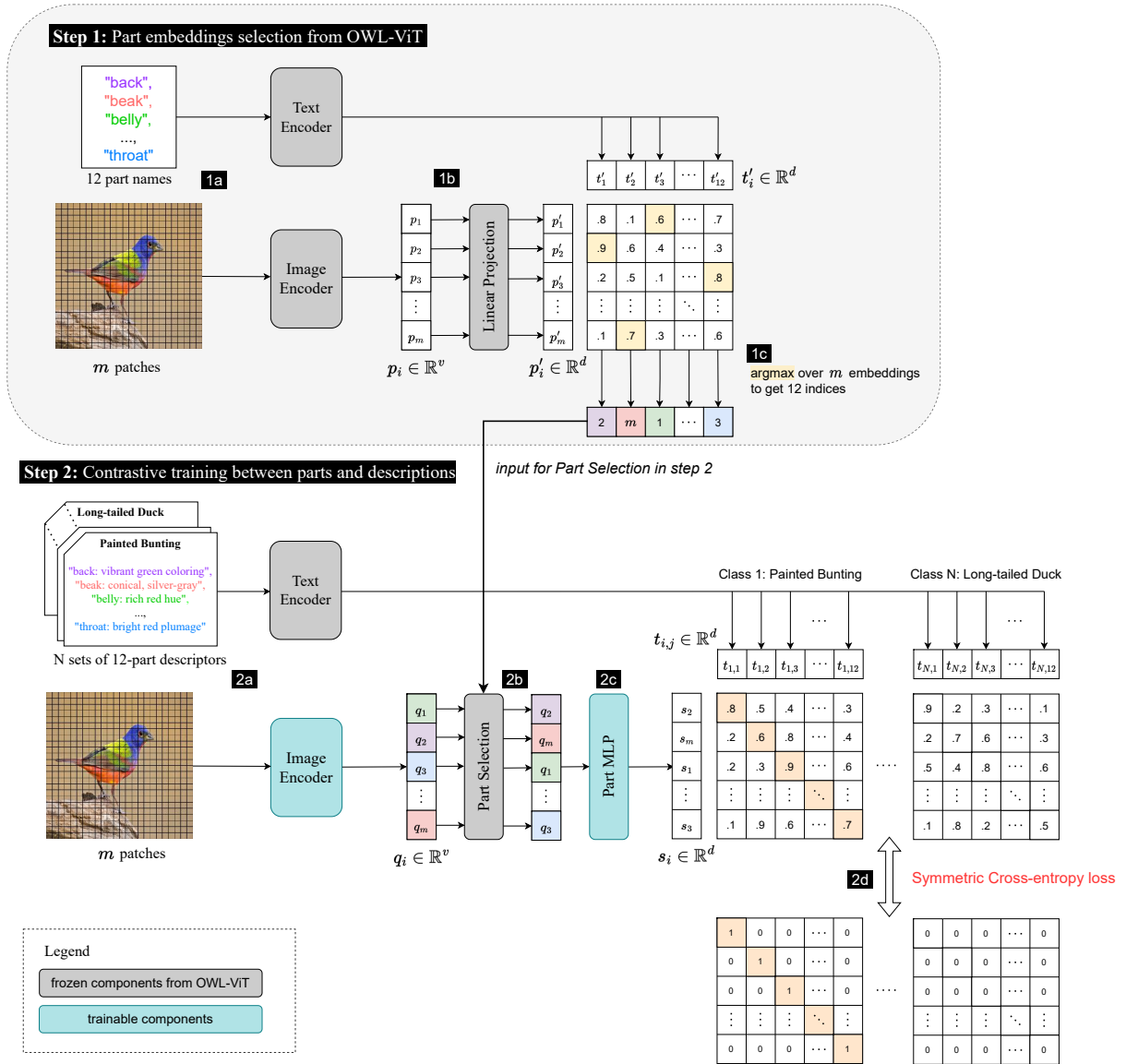


Figure A2: In pre-training stage 1, the objective is to let the Image Encoder learn the general representation of different parts of the birds. Therefore, in pre-training stage 1, we train the *Image Encoder* and *Part MLP* contrastively. During the training, the **Step 1** utilizes a teacher model (OWL-ViT_{B/32}) 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 t'_i , and the patch embedding p_i . (1b) Then the patch embeddings p is forwarded to Linear Projection to obtain p' , matching the dimension of t' . (1c) We compute the dot product between p and t' and apply argmax over 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 t and patch embeddings 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 s , which has the same dimension as t . Then, we compute the similarity matrix for the patch embedding and the descriptor embedding by computing the dot product between s and t . (2d), we construct a one-hot encoded matrix based on the descriptors and minimize the Symmetric Cross-Entropy loss between the similarity matrix in (2c) and the ground truth label.

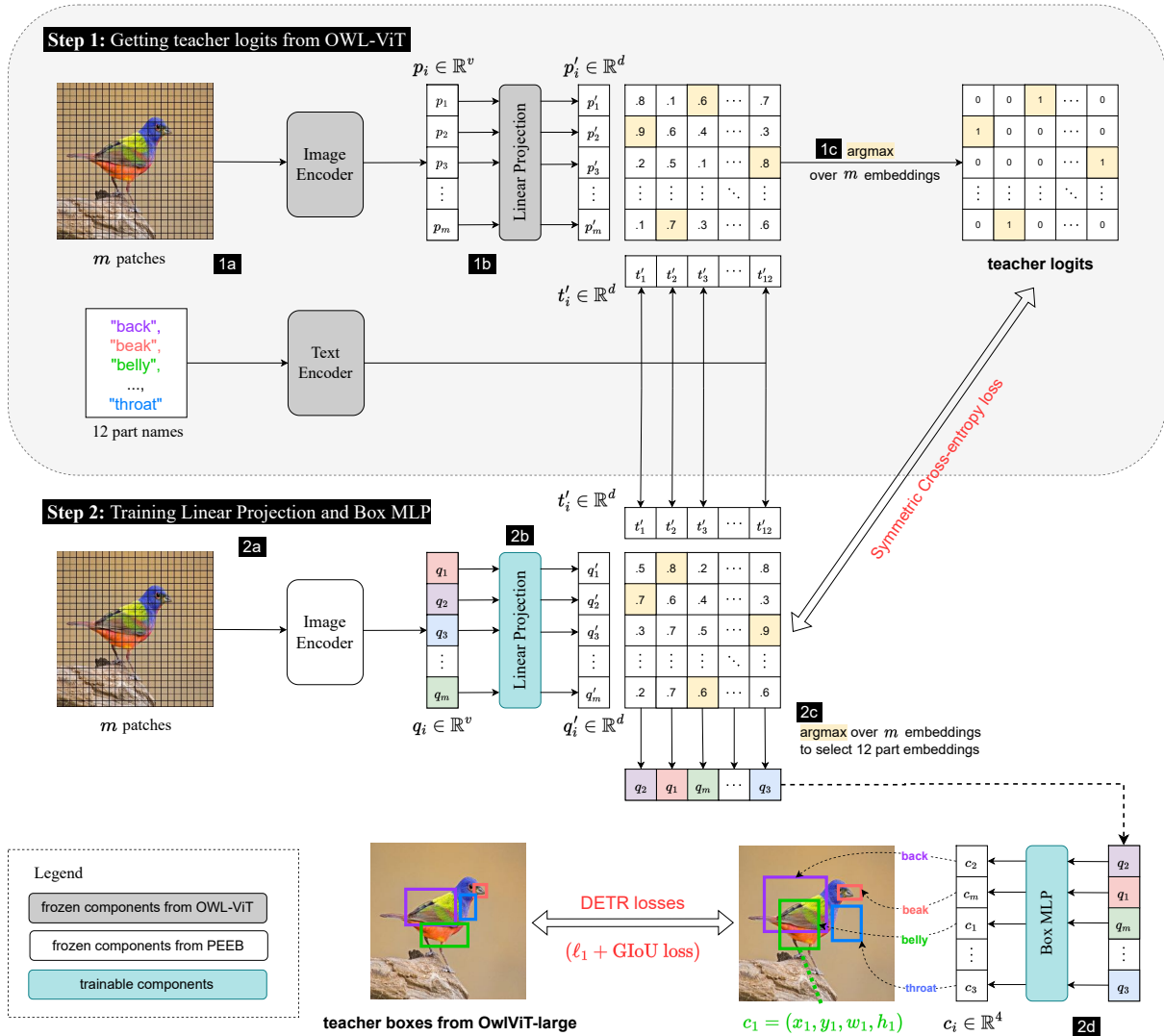


Figure A3: In pre-training stage 2, the goal is to eliminate the teacher model to obtain a standalone classifier. Therefore, the targeted components are **Linear Projection** and **Box MLP**. Since these two components are taking care of different functionalities for patch embedding selection and box 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 t'_i and patch embedding p_i . (1b) Then the patch embeddings p is projected by Linear Projection to obtain p' . (1c) We then compute dot product between p' and one-hot encode the matrix via the dimension of p' to obtain the “teacher logits”. In **Step 2**: (2a), We encode the image with *Image Encoder* to obtain patch embedding q_i . (2b) The patch embeddings are then being projected by **Linear Projection** to derive q' . (2c), We compute the dot product between projected patch embeddings q' and part name embeddings 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 12 part embeddings by taking argmax over 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 _[-test] | 32 | 32 | 50 | $2e^{-4}$ | 0.01 | 48 | 50 | 5 | Bird-11K _[-test] |
| PEEB _[-CUB] | 32 | 32 | 50 | $2e^{-4}$ | 0.001 | 48 | 50 | 10 | Bird-11K _[-CUB] |
| PEEB _[-NAB] | 32 | 32 | 50 | $2e^{-4}$ | 0.001 | 48 | 50 | 10 | Bird-11K _[-NAB] |
| Pre-training stage 2 | | | | | | | | | |
| PEEB _[-test] | 32 | 32 | 50 | $2e^{-5}$ | 0.01 | 48 | 50 | 5 | Bird-11K _[-test] |
| PEEB _[-CUB] | 32 | 32 | 50 | $2e^{-5}$ | 0.001 | 48 | 50 | 5 | Bird-11K _[-CUB] |
| PEEB _[-NAB] | 32 | 32 | 50 | $2e^{-5}$ | 0.001 | 48 | 50 | 5 | Bird-11K _[-NAB] |

Table A2: Details of our finetuned models.

| Model | Fine-tune from | Epoch | Batch size | LR | Weight decay | Early stop | Training set |
|---|-------------------------|-------|------------|-----------|--------------|------------|----------------|
| PEEB _[-test] ^{CUB} | PEEB _[-test] | 30 | 32 | $2e^{-5}$ | 0.001 | 5 | CUB |
| PEEB _[-cub] ^{Akata} | PEEB _[-CUB] | 5 | 32 | $2e^{-5}$ | 0.001 | 5 | CUB ZSL (2015) |
| PEEB _[-cub] ^{SCS} | PEEB _[-CUB] | 5 | 32 | $2e^{-5}$ | 0.001 | 5 | CUB-SCS |
| PEEB _[-cub] ^{SCE} | PEEB _[-CUB] | 5 | 32 | $2e^{-5}$ | 0.001 | 5 | CUB-SCE |
| PEEB _[-nab] ^{SCS} | PEEB _[-NAB] | 5 | 32 | $2e^{-5}$ | 0.001 | 5 | NABirds-SCS |
| PEEB _[-nab] ^{SCE} | PEEB _[-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_[-CUB] and Bird-11K_[-NAB] exclude all CUB and NABirds classes, respectively. For GZSL, we exclude all test sets in CUB, NABirds, and iNaturalist, denoted as Bird-11K_[-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 _[-test] | 234,693 | 29,234 | 10,740 | 9,746 |
| Bird-11K _[-CUB] | 244,182 | 28,824 | 10,602 | 9,608 |
| Bird-11K _[-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_[-test] is pre-trained model using Bird-11K_[-test] dataset.
- PEEB_[-CUB] is pre-trained model using the Bird-11K_[-CUB] dataset.
- PEEB_[-NAB] is pre-trained model using the Bird-11K_[-NAB] dataset.

We named finetuned models after the pre-trained model and the finetuned training set. For example, PEEB_[-test]^{CUB} is finetuned from PEEB_[-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 (e.g., Painted Bunting).

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 Datasets

D.1 Bird-11K

We provide a brief statistic of Bird-11K in Table A4. Bird-11K is a diverse and long-tailed bird-image dataset. The descriptors generated by GPT-4 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 | # of Images | # of 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 | 1,320 |
| 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 restricted³ 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. We use the `train_mini` subset on [Github](#), which has 1,486 classes. After filtering out images (as described in Sec. 3.2), we end up with 1,320 classes and 74,300 images for including in Bird-11K.
- 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):

³See [Terms of Use](#)

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...

D.2 🐕 Dog-140

To pre-train PEEB on dogs, we construct Dog-140 by combining dog images from ImageNet-21K and Stanford Dogs-120. Specifically, we selected 189 dog classes from ImageNet-21K, and based on Fédération Cynologique Internationale (FCI) ([Fédération Cynologique Internationale \(FCI\), 2023](#)), we merged them with 120 classes from Stanford Dogs, ending up with 140 classes. After merging, Dog-140 has 206,076 images in total. We provide a class distribution analysis in Fig. A4, where we can find that Dog-140 is roughly class-balanced.

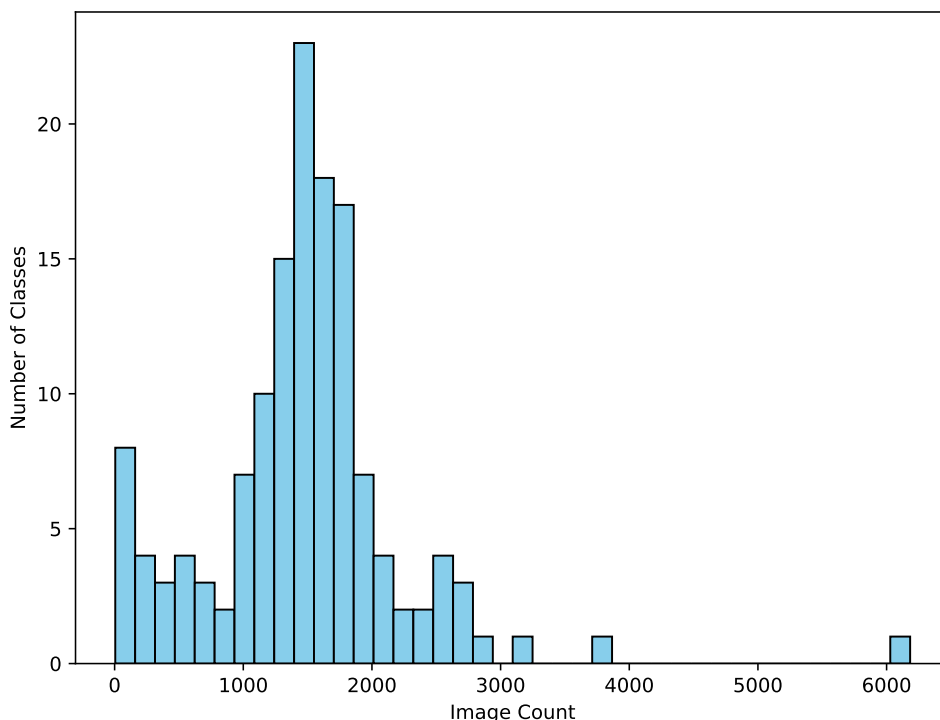


Figure A4: The class distribution of Dog-140 dataset. The histogram indicates that most classes in Dog-140 have around 1,000 to 2,000 images.

Data splits Similar to Bird-11K, we provide data splits and metadata, e.g., file names, image size, and bounding boxes, along with the instruction of Dog-140 construction in our repository.

License and terms

- Stanford Dogs ([Khosla et al., 2011](#)): The dataset was constructed using images and annotations from ImageNet. Therefore, all the images (including those presented in the paper) follow the ImageNet license.
- ImageNet-21K ([Deng et al., 2009](#)): BSD-3-Clause license, non-commercial.

E Additional results

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_[-CUB] and Bird-11K_[-NAB], which deliberately exclude images bearing the same class label as the target dataset. Specifically, we test on our pre-train model PEEB_[-CUB] and PEEB_[-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 substantially higher (+10) than M&V, highlights the advantages of our part-based classification. On NABirds, our method surpasses 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_[-NAB] (the variants of Bird-11K with classes excluded for ZSL testing are described in 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 |

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_{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 100² or 200² 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_[-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.

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

Bird-11K, as shown in Fig. A5a, 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

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 |

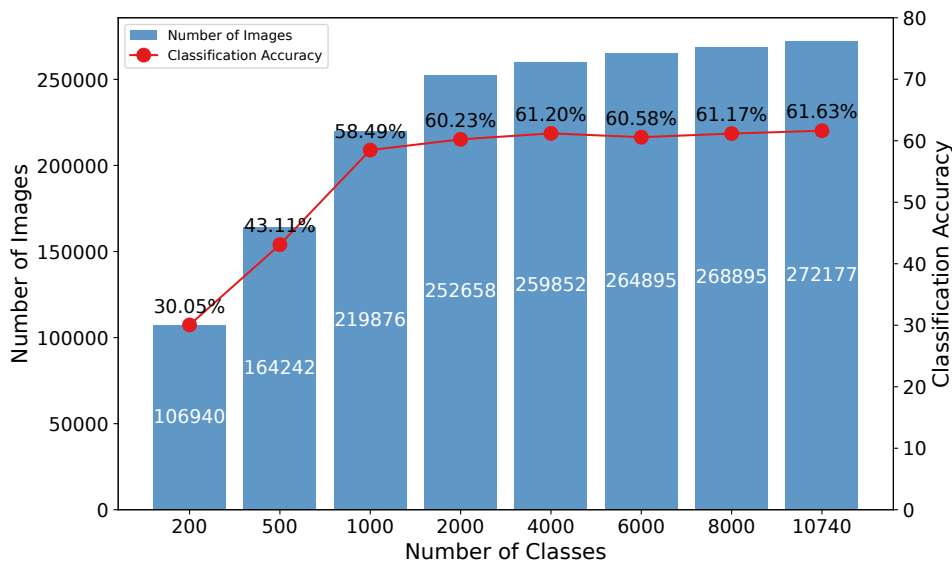
images occurred as the most influential factor. However, a noteworthy observation was that the abundance of long-tailed data enhanced the model’s accuracy by approximately +1.5 points.

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. A5b, 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 A5: 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_[-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% (Table A7). 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.

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, 2023) 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, 2023) | 51.93 | 52.87 | 52.83 | 53.33 | 53.92 |

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_{B/32}. 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_[-test]. These pronounced disparities underscore the vital role of training in PEEB.

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_[-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 _[-test] pre-trained | 64.33 | 69.03 |
| PEEB _[-test] ^{CUB} 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 $\text{PEEB}_{[-\text{test}]}$ 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.

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 |

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_{B/32} 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_{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 finetuned models on their corresponding test sets. For instance, $\text{PEEB}_{[-\text{cub}]}^{\text{Akata}}$, finetuned 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_{B/32}. 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_{B/32}. These observations reinforce that while PEEB’s box predictions might not rival these dedicated object detection models, they consistently highlight the same parts identified by such models as shown in Fig. A6. 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_{Large} as ground-truths, and (2) OWL-ViT_{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 _{Large} | 100.00 | 100.00 | 83.83 |
| | OWL-ViT _{B/32} | 44.41 | 49.65 | 83.53 |
| | PEEB (Average) | 35.98 | 40.14 | 83.39 |
| | PEEB _[-test] ^{CUB} | 37.45 | 41.79 | 81.55 |
| | PEEB _[-cub] ^{Akata} | 35.11 | 39.14 | 82.72 |
| | PEEB _[-cub] ^{SCS} | 35.77 | 39.96 | 84.89 |
| | PEEB _[-cub] ^{SCE} | 35.58 | 39.67 | 84.38 |
| | NABirds | OWL-ViT _{Large} | 100.00 | 100.00 |
| OWL-ViT _{B/32} | | 40.14 | 47.63 | 83.89 |
| PEEB (Average) | | 36.47 | 42.01 | 80.72 |
| PEEB _[-nab] ^{SCS} | | 36.45 | 42.03 | 80.09 |
| PEEB _[-nab] ^{SCE} | | 36.49 | 41.99 | 81.34 |

F Study on GPT-4 generated descriptors

F.1 Assessment of the generated part-based descriptors

We test GPT-4V on the CUB test set using the generated descriptors of 200 classes to assess their usability. Specifically, we feed GPT-4V with each test image encoded in the payload and 200 sets of part-based descriptors through a carefully designed prompt (Table A11). GPT-4V is asked to output one of 200 provided class names to compute the classification accuracy. As a result, GPT-4V achieves 69.4% accuracy which is slightly higher than PEEB’s generalized zero-shot accuracy (64.33%) and significantly lower than PEEB results after finetuning (86-88%).

Table A11: Prompt for GPT-4V evaluation on CUB where {list_of_200_classes} is the placeholder for the actual 200 CUB classes while {descriptors} (see an example in appendix C) is the placeholder for the actual descriptors associated with a given bird image from the CUB test set.

You are an image classifier which can tell what type of a bird is from the given image and its associated part descriptors describing 12 parts of the bird. Your answer should be strictly formatted as {"prediction": "bird_class"}.

where "bird_class" is one of the following 200 bird classes: {list_of_200_classes}

Given the bird image and the following descriptors: {descriptors}

What kind of bird is this? Let’s think step by step.

F.2 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 A12). 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.

Table A12: 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 |

F.3 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 part's 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.

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 GPT-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 PEEB_[-test].

Results As shown in Table A13, the overall accuracy increases by +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 A13: The revised descriptors result in +0.8 for PEEB_[-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 |
|-------------------------|----------|-------------------|------------------|
| PEEB _[-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 OWL-ViT_{Large}, OWL-ViT_{B/32}, and PEEB in Fig. A6. We find that despite the fact that our predicted boxes have lower mean IoU compared to OWL-ViT_{Large}, they are visually similar to the boxes as OWL-ViT_{B/32}.

G.2 Qualitative examples of using randomized descriptors

We visually compare M&V and PEEB based on their utilization of descriptors. (Figs. A7 to A9). 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 for birds

Figs. A10 to A12 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. A10, while other descriptors are similar, PEEB can still reject chesnut-sided warbler thanks to the distinct features of *forehead*, *throat* and *belly*.

G.4 Examples of PEEB explanations for dogs

Figs. A13 to A15 are examples of how PEEB makes classification based on the descriptors in Stanford Dogs dataset. We demonstrate that our model works well on dogs, which indicates that our proposed method is transferable to other domains while maintaining high-quality explainability as in birds.

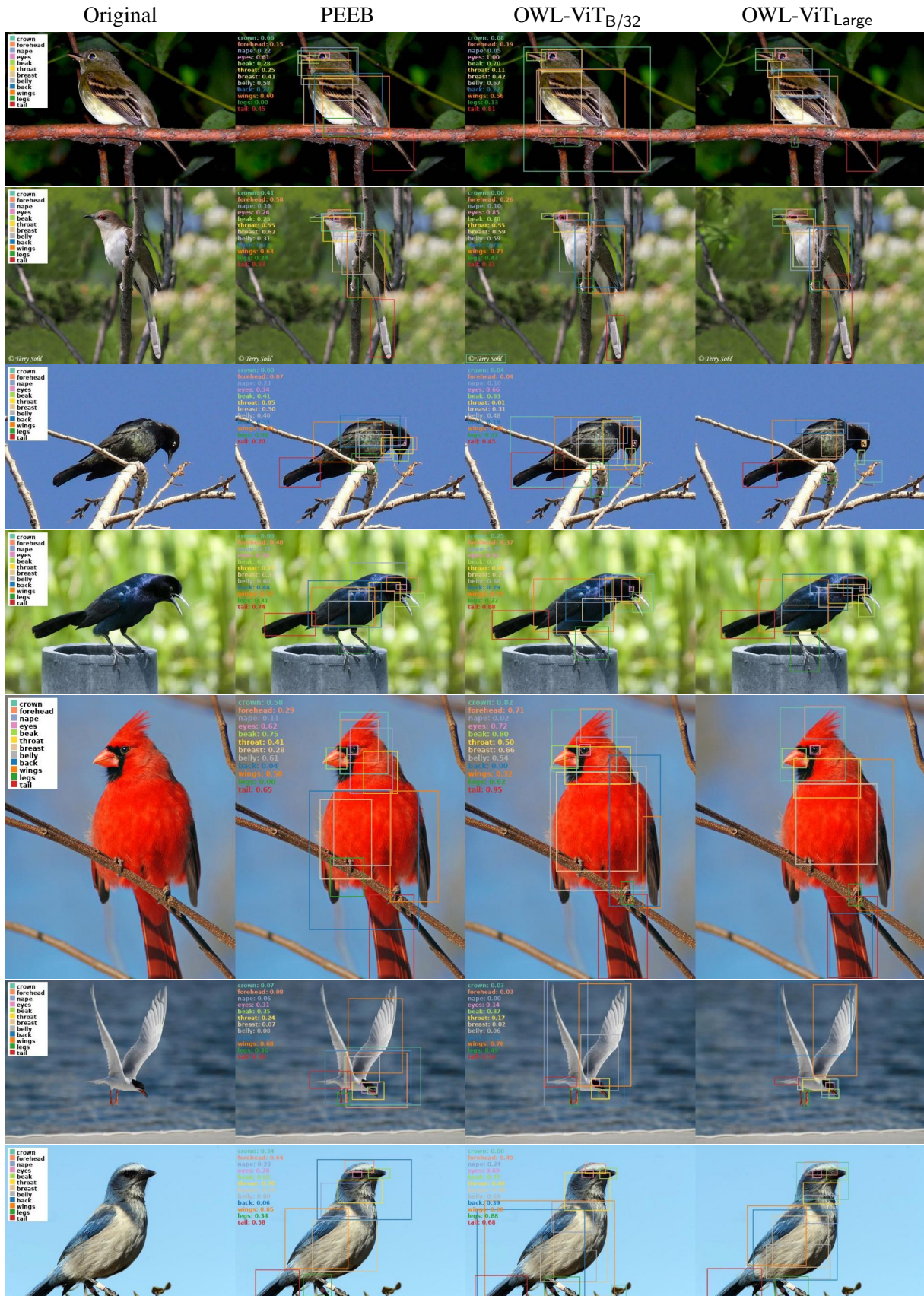


Figure A6: Our predicted boxes (second column) often align closely with those of OWL-ViT_{B/32} (third column). However, slight shifts can lead to significant IoU discrepancies. For instance, in the first row, both PEEB and OWL-ViT_{B/32} accurately identify the tail. Yet, variations in focus yield a stark IoU contrast of 0.45 versus 0.81.

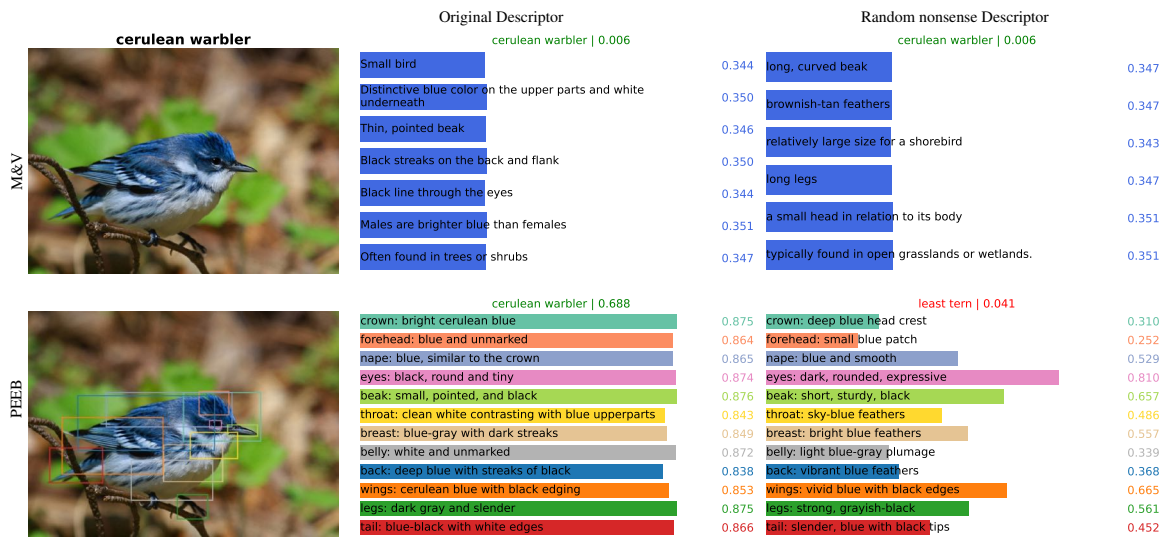


Figure A7: Qualitative example of original descriptors vs. randomized descriptors. Upon swapping descriptors randomly, the prediction outcomes from M&V exhibit minimal variations.



Figure A8: 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.

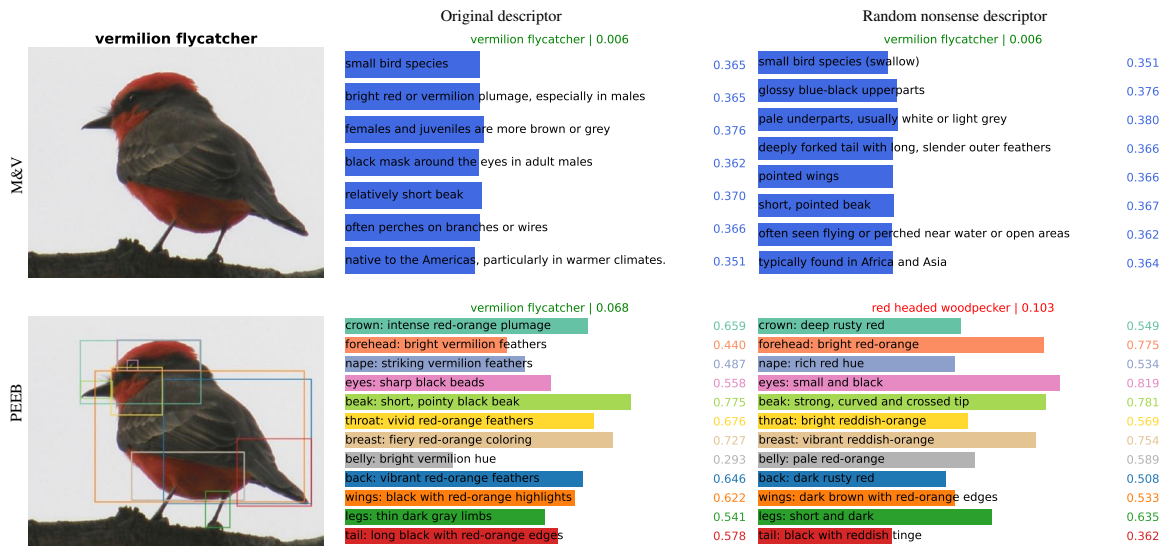


Figure A9: Qualitative example of original descriptors vs. randomized descriptors. M&V maintains similar scores even for mismatched descriptors. For instance, “bright red or vermillion plumage, especially in males” receives a score lower than “glossy blue-black upperparts”. Conversely, PEEB leverages the descriptors for classification, consistently relying on the descriptors that most closely align with the image.



Figure A10: 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.

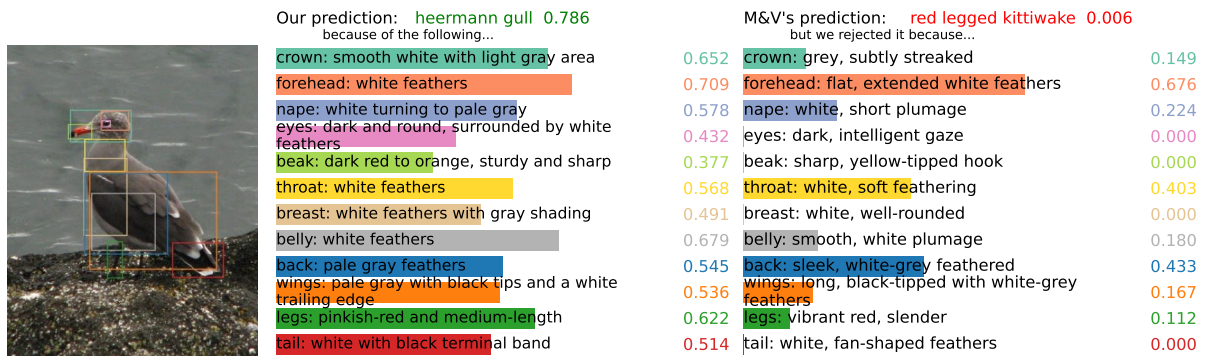


Figure A11: An example of PEEB explanation. M&V incorrectly classifies it as red-legged kittiwake where the heermann gull does not have red legs but a red beak. This example shows that CLIP is strongly biased towards some particular descriptors.



Figure A12: 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.

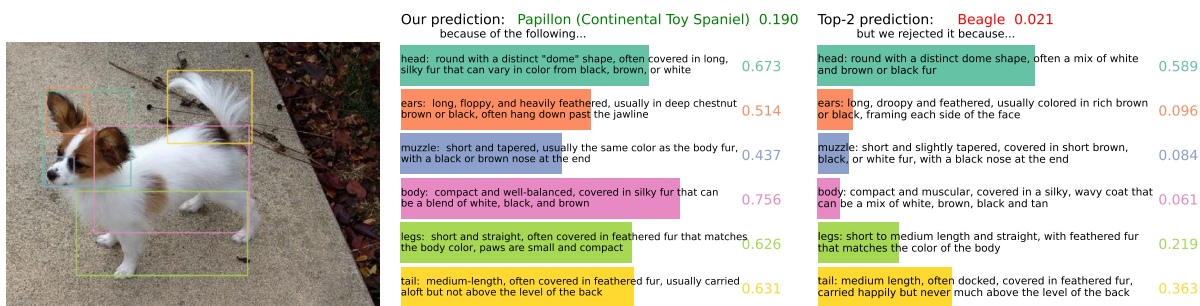


Figure A13: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.

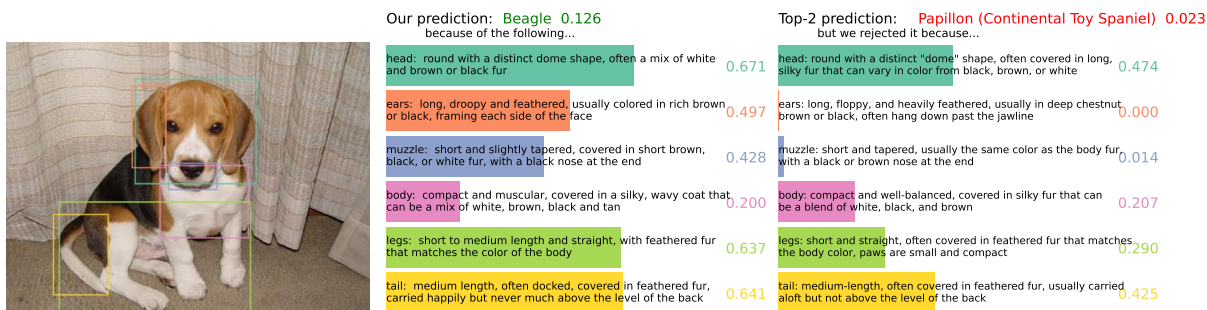


Figure A14: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.

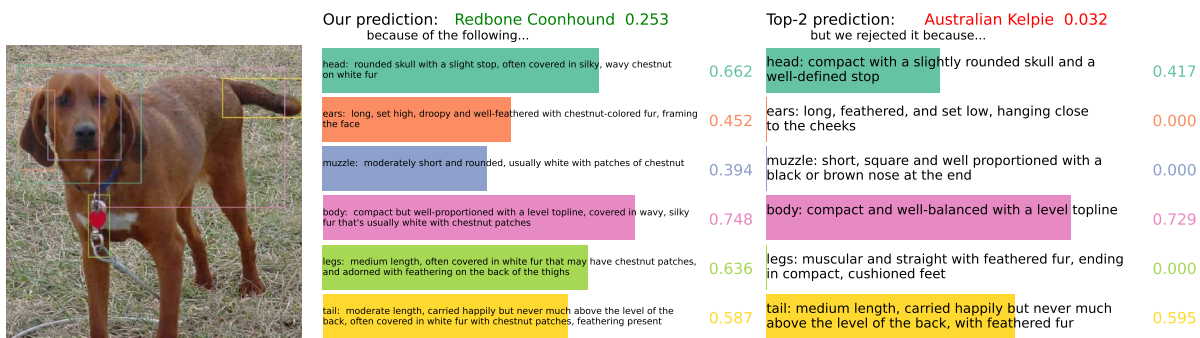


Figure A15: An example of PEEB explanation for dogs. Like birds, PEEB first identifies the predefined parts and then matches them to the descriptions.