Interpretable Image Classification with Adaptive Prototype-based Vision Transformers

Chiyu Ma
Dartmouth College
chiyu.ma.gr@dartmouth.edu

Jon Donnelly
Duke University
jon.donnelly@duke.edu

Wenjun Liu
Dartmouth College
wenjun.liu.gr@dartmouth.edu

Soroush Vosoughi
Dartmouth College
soroush.vosoughi@dartmouth.edu

Cynthia Rudin
Duke University
cynthia@cs.duke.edu

Chaofan Chen
University of Maine
chaofan.chen@maine.edu

Abstract

We present ProtoViT, a method for interpretable image classification combining deep learning and case-based reasoning. This method classifies an image by comparing it to a set of learned prototypes, providing explanations of the form "this looks like that." In our model, a prototype consists of *parts*, which can deform over irregular geometries to create a better comparison between images. Unlike existing models that rely on Convolutional Neural Network (CNN) backbones and spatially rigid prototypes, our model integrates Vision Transformer (ViT) backbones into prototype based models, while offering spatially deformed prototypes that not only accommodate geometric variations of objects but also provide coherent and clear prototypical feature representations with an adaptive number of prototypical parts. Our experiments show that our model can generally achieve higher performance than the existing prototype based models. Our comprehensive analyses ensure that the prototypes are consistent and the interpretations are faithful. Our code is available at https://github.com/Henrymachiyu/ProtoViT.

1 Introduction

With the expanding applications of machine learning models in critical and high-stakes domains like healthcare [4, 14, 44, 52, 51], autonomous vehicles [32], finance [47] and criminal justice [5], it has become crucial to develop models that are not only effective but also interpretable by humans. This need for clarity and accountability has led to the emergence of prototype networks. These networks combine the capabilities of deep learning with the clarity of case-based reasoning, providing understandable outcomes in fine-grained image classification tasks. Prototype networks operate by dissecting an image into informative image patches and comparing these with prototypical features established during their training phase. The model then aggregates evidence of similarities to these prototypical features to draw a final decision on classification.

Existing prototype-based models [10, 4, 13, 26, 35, 36, 46, 27, 45, 8, 7], which are **inherently interpretable** [34] (i.e., the learned prototypes are directly interpretable, and all calculations can be visibly checked), are mainly developed with convolutional neural networks (CNNs). As vision transformers (ViTs) [15, 40, 41] gain popularity and inspire extensive applications, it becomes crucial to investigate how prototype-based architectures can be integrated with vision transformer backbones. Though a few attempts to develop a prototype-based vision transformer have been made [50, 49, 22], these methods do not provide inherently interpretable explanations of the models' reasoning because they do not project the learned prototypical features to examples that actually exist in the dataset.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1312

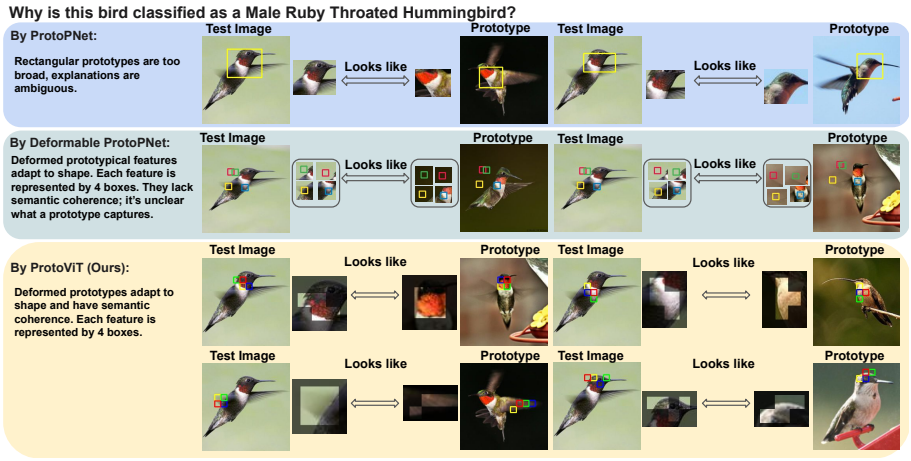

Figure 1: Reasoning process of how prototype based models identify a test image of a male Ruby-Throated Hummingbird as the correct class. Prototypes are shown in the bounding boxes.

Table 1: Table of contributions of ProtoVit (Ours) compared to existing works.

Model	Support ViT Backbone?	Deformable Prototypes?	Coherent Prototypes?	Adaptive Sizes?	Inherently Interpretable?
ProtoPNet [10]	Yes	No	Maybe	No	Yes
Deformable ProtoPNet [13]	No	Yes	No	No	Yes
ProtoPformer [50]	Yes	No	Maybe	No	No
ProtoViT (Ours)	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Thus, the **cases** that these models use in their reasoning have no well defined visual representation, making it impossible to know what exactly each prototype represents.

The coherence and clarity of the learned prototypes are important. Most prototype-based models, such as ProtoPNet [10], TesNet [46] and ProtoConcept [26], use spatially rigid prototypical features (such as a rectangle), which cannot account for geometric variations of an object. Such rigid prototypical features can be ambiguous as they may contain multiple features in one bounding box (see top row of Fig. 1 for an example). Deformable ProtoPNet [13] deforms the prototypes into multiple pieces to adjust for the geometric transformation. However, Deformable ProtoPNet utilizes deformable convolution layers that rely on a continuous latent space to derive fractional offsets for prototypical parts to move around, and thus are not suitable for models like ViTs which output discrete tokens as latent representations. Additionally, the prototypes learned by Deformable ProtoPNet tend to be incoherent (see Fig. 1, middle row).

Addressing the gaps in current prototype-based methods (see Table 1), we propose the prototype-based vision transformer (ProtoViT), a novel architecture that incorporates a **ViT backbone** and can **adaptively** learn **inherently interpretable** and **geometrically variable** prototypes of **different sizes**, without requiring explicit information about the shape or size of the prototypes. We do so using a novel greedy matching algorithm that incorporates an adjacency mask and an adaptive slots mechanism. We provide global and local analysis, shown in Fig. 4, that empirically confirm the **faithfulness** and **coherence** of the prototype representations from ProtoViT. We show through empirical evaluation that ProtoViT achieves state-of-the-art accuracy as well as excellent clarity and coherence.

2 Related Work

Posthoc explanations for CNNs like activation maximization [16, 28], image perturbation [17, 20], and saliency visualizations [3, 38, 39] fall short in explaining the reasoning process of deep neural networks because their explanations are not necessarily faithful [33, 2]. In addition, numerous efforts have been made to enhance the interpretability of Vision Transformers (ViTs) by analyzing attention weights [1, 42], and other studies focus on understanding the decision-making process through gradients [18, 19], attributions [9, 53], and redundancy reduction techniques [30]. However, because of their posthoc nature, the outcomes of these techniques can be uncertain and unreliable [2, 25].

In contrast, prototype-based approaches offer a transparent prediction process through **case-based reasoning**. These models compare a small set of learned latent feature representations called prototypes (the "cases") with the latent representations of a test image to perform classification. These models are inherently interpretable because they leverage comparisons to only well-defined cases in reasoning. The original Prototypical Part Network (ProtoPNet) [10] employs class-specific prototypes, allocating a fixed number of prototypes to each class. Each prototype is trained to be similar to feature patches from images of its own class and dissimilar to patches from images of other classes. Each prototype is also a latent patch from a training image, which ensures that "cases" are well-defined in the reasoning process. The similarity score from the test image to each prototype is added as positive evidence for each class in a "scoresheet," and the class with the highest score is the predicted class. The Transparent Embedding Space Network (TesNet) [46] refines the original ProtoPNet by utilizing a cosine similarity metric to compute similarities between image patches and prototypes in a latent space. It further introduces new loss terms to encourage the orthogonality among prototype vectors of the same class and diversity between the subspaces associated with each class. Deformable ProtoPNet [13] aims to decompose the prototypical parts into smaller sub-patches and use deformable convolutions [12, 54] to capture pose variations. Other works [35, 27, 36] move away from class-specific prototypes, which reduces the number of prototypes needed. This allows prototypes to represent a similar visual concept shared in different classes. Our work is similar to these works that define "cases" as latent patch representations by projecting the trained class-specific prototypes to the closest latent patches.

As an alternative to the closest training patches, ProtoConcept [26] defines the cases as a group of visualizations in the latent spaces bounded by a prototypical ball. Although they are not projected, the visualizations of ProtoConcept are fixed to the set of training patches falling in each prototypical ball, which establishes well defined cases for each prototype. On the other hand, works such as ProtoPFormer [50] do not project learned prototypes to the closest training patches because they observed a performance degradation after projection. The degradation is likely caused by the fact that the prototypes are not sufficiently close to any of the latent patches. The design of a "global branch" that aims to learn prototypical features from class tokens also raises concerns about interpretability, as visualizing an arbitrary trained vector (class token) does not offer any semantic connection to the input image. On the other hand, work such as ViTNet [22] also lack details about how the "cases" are established, yielding concerns about the interpretability of the model. Without a mechanism like *prototype projection* [10] or *prototypical concepts* [26] to enable visualizations, prototypes are just arbitrary learned tensors in the latent space of the network, with no clear interpretations. Visualizing nearby examples alone cannot explain the model's reasoning, since the closest patch to a prototype can be arbitrarily far away without the above mechanisms. **Simply adding a prototype layer to an architecture without well defined "cases" in the reasoning process does not make the new architecture more interpretable.**

Our work is also related to INTR [31], which trains a class specific attention query and inputs it to a decoder to localize the patterns in an image with cross-attention. In contrast, our encoder-only model, through a different reasoning approach, learns patch-wise deformed features that are more explicit, semantically coherent, and provides more detail about the reasoning process than attention heatmaps [21, 48, 6] – it shows how the important pixels are used in reasoning, not just where they are.

3 Methods

We begin with a general introduction of the architecture followed by an in-depth exploration of its components. We then delve into our novel training methodology that encourages prototypical features to capture semantically coherent attributes from the training images. Detailed implementations on specific datasets are shown in the experiment sections.

3.1 Architecture Overview

Fig. 2 shows an overview of the ProtoVit architecture. Our model consists of a feature encoder layer f, which is a pretrained ViT backbone that computes a latent representation of an image; a greedy matching layer g, which compares the latent representation to learned prototypes to compute prototype similarity scores; and an evidence layer h, which aggregates prototype similarity scores into a classification using a fully connected layer. We explain each of these in detail below. Let

https://doi.org/10.52202/079017-1312

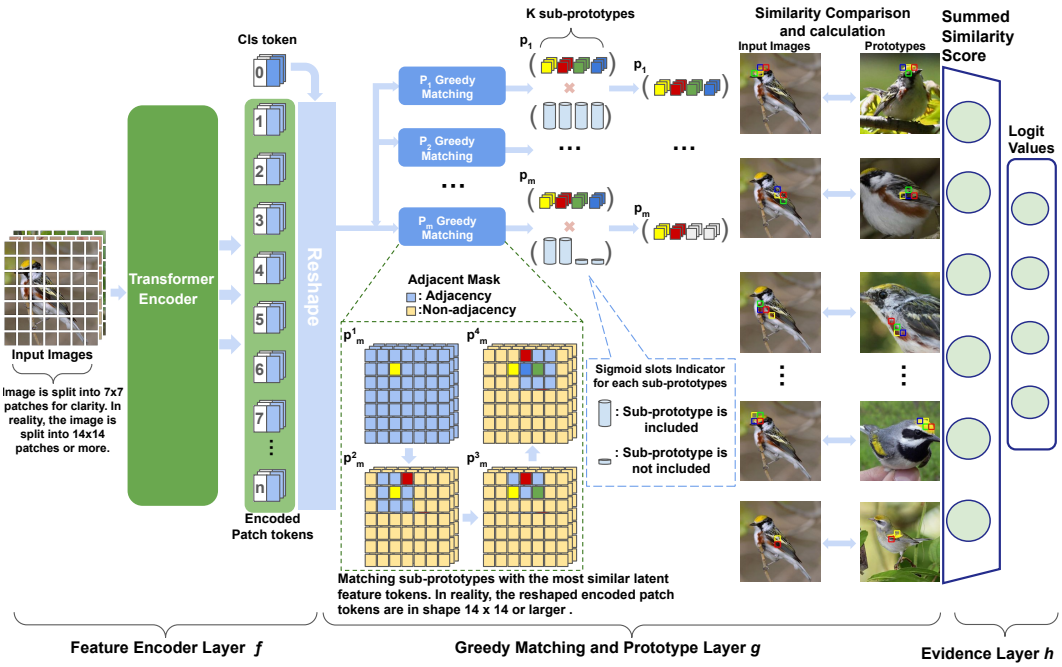

Figure 2: Architecture of ProtoViT. The feature encoder layer f can be any kind of vision transformer encoders such as DeiT and CaiT. The greedy matching and prototype layer g deforms the prototypes into sub-prototypes and finds the closest non-overlapping feature patches from the test image. Our adaptive slots mechanism filters out some number of sub-prototypes, and the sum of the remaining similarity scores (with a correction to avoid down-weighting prototypes with more sub-prototypes filtered out) is returned. The evidence layer h is a fully connected layer that computes the logit predictions based on the summed similarity scores.

$H \times W \times C$ be the shape of an input image $\mathbf{x}$, where H, W, C are the height, width and number of channels of the image respectively. The feature encoder layer f is a ViT backbone, which first splits the input images into N units of $L \times L \times C$ sized patches (such as the Chestnut Sided Warbler in Fig. 2), where L is the height and width of the image patch. The feature encoder layer f then flattens and encodes the image patches into a set of latent feature tokens $\mathbf{z}_f$ defined later in Eq. 1, where each latent feature token $\mathbf{z}_f^i \in \mathbb{R}^d$ and d is the hidden dimension of the model. The network aims to learn m prototypes $\mathbf{P} = \{\mathbf{p}_j\}_{j=1}^m$, where each $\mathbf{p}_j \in \mathbb{R}^{K \times d}$ is composed of K sub-prototype vectors $\mathbf{p}_j^k \in \mathbb{R}^d$. The greedy matching layer g then finds the most similar latent feature token $\mathbf{z}_f^i$, measured by cosine similarity, of the input image to each sub-prototype $\mathbf{p}_j^k$ without replacement (i.e., if $\mathbf{p}_j^1$ matches $\mathbf{z}_f^1$, $\mathbf{p}_j^2$ cannot match with $\mathbf{z}_f^1$). We further introduce an adjacency mask A to ensure that, for each prototype $\mathbf{p}_j$, its sub-prototypes $\mathbf{p}_j^k$ are geometrically contiguous. We then introduce our adaptive slots mechanism, which decides whether each prototype $\mathbf{p}_j$ should include each sub-prototype $\mathbf{p}_j^k$ based on its coherence to the other sub-prototypes and its influence on model performance, allowing the model to learn prototypes with a dynamic number of sub-prototypes. Finally, the evidence layer h computes the logit values for each class based on the summed cosine similarity score $g_{\mathbf{p}_j}^{\text{greedy}}$ for each prototype $\mathbf{p}_j$. The logit values are then normalized by a softmax function to make predictions. Intuitively, the prototypes, which are later projected to the closest latent feature tokens, can be viewed as the most representative features of each class that the model found among the training examples. The model performs classification based on the input's similarity to these key features.

3.2 Feature Encoder Layer

Given an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, the feature encoder layer first splits the image into N unit patches $\mathbf{x}_{\text{patch}}$ each of shape $L \times L \times C$. It then flattens the unit patches and projects them

to the embedding space as $\mathbf{x}_{\text{Embed}} \in \mathbb{R}^{N \times d}$, which is then prepended with a trainable class token $\mathbf{x}_{\text{class}} \in \mathbb{R}^d$, and passed into the Multi-Head Attention layers along with a learnable position embedding $E_{\text{pos}} \in \mathbb{R}^{(N+1) \times d}$. The ViT backbone then outputs the encoded tokens as $\mathbf{z} := [\mathbf{z}_{\text{class}}; \mathbf{z}^1_{\text{patch}}; \mathbf{z}^2_{\text{patch}}; \cdots; \mathbf{z}^N_{\text{patch}}]$, where $\mathbf{z} \in \mathbb{R}^{(N+1) \times d}$. In this sense, each of the output patch tokens $\mathbf{z}^i_{\text{patch}}$ is the latent representation of the corresponding unit patch $\mathbf{x}^i_{\text{patch}}$. The class token can be viewed as a way to approximate the weighted sum over patch tokens, enabling an image-level representation. It is, thus, difficult to visualize the class token and therefore unsuitable for comparisons to prototypes. Drawing inspiration from the idea of focal similarity [36], which involves calculating the maximum prototype activations minus the mean activations to achieve more focused representations, we take the difference between patch-wise features and the image-level representation. In doing so, we aim to similarly produce more salient visualizations. Specifically, we define the feature token $\mathbf{z}_f$ by taking the difference between each patch token and the image-level representation. Thus, latent feature tokens can be written as:

$$\mathbf{z}_f := [\mathbf{z}^1_f; \mathbf{z}^2_f; \cdots; \mathbf{z}^N_f], \text{ where } \mathbf{z}^i_f = \mathbf{z}^i_{\text{patch}} - \mathbf{z}_{\text{class}}, \text{ and } \mathbf{z}^i_f \in \mathbb{R}^d. \tag{1}$$

By this design, we not only encode richer semantic information within the latent feature tokens, but also enable the application of prototype layers to various ViT backbones that contains a class token. An ablation study shows that model performance drops substantially when removing the class token from the feature encoding (see Appendix Sec. E.1) .

3.3 Greedy Matching Layer

The greedy matching layer g integrates three key components: a greedy matching algorithm, adjacency masking, and an adaptive slots mechanism, as illustrated in Fig. 2.

Similarity Metric: The greedy matching layer contains m prototypes $\mathbf{P} = \{\mathbf{p}_j\}_{j=1}^m$, where each $\mathbf{p}_j$ consists of K sub-prototypes $\mathbf{p}^k_j$ that have the same dimension as each latent feature token $\mathbf{z}^i_f$. Each sub-prototype $\mathbf{p}^k_j$ is trained to be semantically close to at least one latent feature token. To measure the closeness of the sub-prototype $\mathbf{p}^k_j$ and latent feature token $\mathbf{z}^i_f$, we use cosine similarity $\cos(\mathbf{z}^i_f, \mathbf{p}^k_j) = \frac{\mathbf{z}^{i\,T}_f \mathbf{p}^k_j}{\|\mathbf{z}^i_f\|_2 \|\mathbf{p}^k_j\|_2}$, which has range -1 to 1. The overall similarity between prototype $\mathbf{p}_j$ and the latent feature tokens $\mathbf{z}_f$ is then computed as the sum across the sub-prototypes of the cosine similarities, which has a range from $-K$ to K.

Greedy Matching Algorithm: We have not yet described how the token $\mathbf{z}^i_f$ to which $\mathbf{p}^k_j$ is compared is selected. In contrast to prior work, we do not restrict that sub-prototypes have fixed relative locations. Rather, we perform a greedy matching algorithm to identify and compare each of the K sub-prototypes $\mathbf{p}^k_j$ of $\mathbf{p}_j$ to *any* K non-overlapping latent feature tokens. To be exact, for a given prototype $\mathbf{p}_j$ with K sub-prototypes, we iteratively identify the sub-prototype $\mathbf{p}^k_j$ and latent feature token $\mathbf{z}^i_f$ that are closest in cosine similarity, "match" $\mathbf{z}^i_f$ with $\mathbf{p}^k_j$, and remove $\mathbf{z}^i_f$ and $\mathbf{p}^k_j$ from the next iteration, until all K pairs are found. This is illustrated in Fig. 2, and more details can be found in Appendix Sec. C.

Adjacency Masking: Without any restrictions, this greedy matching algorithm can lead to many sub-prototypes that are geometrically distant from each other. Assuming that the image patches representing the same feature are within $r \in \mathbb{R}$ spatial units of one another in horizontal, vertical and diagonal directions, we introduce the Adjacency Mask A to temporarily mask out latent feature tokens that are more than r positions away from a selected sub-prototype/latent feature token pair in all directions. Within each iteration k of the greedy matching algorithm, the next pair can only be selected from the latent feature tokens, $\mathbf{z}_{A^k_j} = A(\{\mathbf{p}^{k-1}_j, \mathbf{z}^{i\,k-1}_f\}; \mathbf{z}_f; r)$, that are within r positions of the last selected pair. An example of how the adjacency mask with $r = 1$ works is illustrated in Fig. 2. Incorporating adjacency masking with the greedy matching algorithm, we thus find K non-overlapping and adjacent sub-prototype-latent patch pairs.

Adaptive Slots Mechanism: Because not all concepts require K sub-prototypes to represent, we introduce the the adaptive slots mechanism S. The adaptive slots mechanism consists of learnable

https://doi.org/10.52202/079017-1312

vectors $\mathbf{v} \in \mathbb{R}^{m \times K}$ and a sigmoid function with a hyperparameter temperature τ. We chose a sufficiently large value for τ to ensure that the sigmoid function has a steep slope. The vectors $\mathbf{v}$ are sent to the sigmoid function to approximate the indicator function as $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}} = \textbf{Sigmoid}(\mathbf{v}_j^k, \tau)$. Each $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}}$ is an approximation of the indicator for whether the k-th sub-prototype will be included in the j-th prototype $\mathbf{p}_j$. More details can be found in Appendix Sec. D.

Computation of Similarity: As described above, we measure the similarity of a selected latent-patch–sub-prototype pair using cosine similarity. We use the summed similarity across sub-prototypes to measure the overall similarity for prototype $\mathbf{p}_j$ with K selected non-overlapping latent feature tokens. As pruning out some sub-prototypes for a prototype $\mathbf{p}_j$ reduces the range of the summed cosine similarity for $\mathbf{p}_j$, we rescale the summed similarities back to their original range by $K/\sum_k \tilde{\mathbb{1}}\{\text{include } \mathbf{p}_j^k\}$. We formally define the reweighted summed similarity function obtained from the greedy matching layer g as:

$$g_{\mathbf{p}_j}^{\text{greedy}}(\mathbf{z}_f) = \frac{K}{\sum_{k=1}^{K} \tilde{\mathbb{1}}_{\{\text{include } \mathbf{p}_j^k\}}} \sum_{k=1}^{K} \left\{ \max_{\mathbf{z}_f^i \in \mathbf{z}_{A_j^k}} \cos(\mathbf{z}_f^i, \mathbf{p}_j^k) \right\} \cdot \tilde{\mathbb{1}}_{\{\text{include } \mathbf{p}_j^k\}}. \tag{2}$$

3.4 Training Algorithm

Training of ProtoViT has four stages: (1) optimizing layers before the last layer by stochastic gradient descent (SGD); (2) prototype slots pruning; (3) projecting the trained prototype vectors to the closest latent patches; (4) optimizing the last layer h. Note that a well-trained first stage is crucial to achieve minimal performance degradation after prototype projection. A procedure plot is illustrated in Appendix Fig. 5.

Optimization of layers before last layer: The first training stage aims to learn a latent space that clusters feature patches from the training set that are important for a class near semantically similar prototypes of that class. This involves solving a joint optimization problem over the network parameters via stochastic gradient descent (SGD). We initialize every slot indicator $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}}$ as 1 to allow all sub-prototypes to learn during SGD. We initialize the last layer weight similarly to ProtoPNet [10]. The slot indicators and the final layer are frozen during this training stage. The slot indicator functions are not involved in computing cluster loss or separation loss because each sub-prototype should remain semantically close to certain latent feature tokens, regardless of its inclusion in the final computation. Since we deform the prototypes, we propose modifications to the original cluster and separation loss defined in [10] to:

$$\begin{aligned} \mathcal{L}_{Clst} &= -\frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{p}_j \in \mathbf{P}_{y_i}} \max_{\mathbf{p}_j^k \in \mathbf{p}_j} \max_{\mathbf{z}_f^i \in \mathbf{z}_{A_j^k}} \cos(\mathbf{z}_f^i, \mathbf{p}_j^k); \\ \mathcal{L}_{Sep} &= \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{p}_j \notin \mathbf{P}_{y_i}} \max_{\mathbf{p}_j^k \in \mathbf{p}_j} \max_{\mathbf{z}_f^i \in \mathbf{z}_{A_j^k}} \cos(\mathbf{z}_f^i, \mathbf{p}_j^k). \end{aligned} \tag{3}$$

The minimization of this new cluster loss encourages each training image to have some unit latent feature token $\mathbf{z}_f^i$ that is close to at least one sub-prototypes $\mathbf{p}_j^k$ of its own class. Similarly, by minimizing the new separation loss, we encourage every latent feature token of a training image to stay away from the nearest sub-prototype from any incorrect class. This is similar to the traditional cluster and separation loss from [10] if we define the prototypes to consist of only one sub-prototype. We further introduce a novel loss term, called coherence loss, based on the intuition that, if the sub-prototypes collectively represent the same feature (e.g., the feet of a bird), these sub-prototypes should be similar under cosine similarity. The coherence loss is defined as:

$$\mathcal{L}_{Coh} = \frac{1}{m} \sum_{j=1}^{m} \max_{\mathbf{p}_j^k, \mathbf{p}_j^s \in \mathbf{p}_j; \mathbf{p}_j^s \neq \mathbf{p}_j^k} (1 - \cos(\mathbf{p}_j^k, \mathbf{p}_j^s)) \cdot \tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}} \tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^s\}}. \tag{4}$$

Intuitively, the coherence loss penalizes the sub-prototypes that are the most dissimilar to the other sub-prototypes out of the K slots for prototype $\mathbf{p}_j$. The slots indicator function is added to the

coherence loss term to prune sub-prototypes that are semantically distant from others in a prototype. Moreover, we use the orthogonality loss, introduced in previous work [46, 13], to encourage each prototype $\mathbf{p}_j$ to learn distinctive features. The orthogonality loss is defined as:

$$\mathcal{L}_{Orth} = \sum_{l=1}^{C} \|\mathbf{P}^{(l)}\mathbf{P}^{(l)^T} - \mathbf{I}_\rho\|_{\mathbf{F}}^2 \tag{5}$$

where C is the number of classes. For each class l with ρ assigned prototypes, $\mathbf{P}^{(l)} \in \mathbb{R}^{\rho \times Kd}$ is a matrix obtained by flattened the prototypes from class l, and ρ is the number of prototypes $\mathbf{p}_j$ for each class. $\mathbf{I}_\rho$ is an identity matrix in the shape of $\rho \times \rho$. Overall, this training stage aims to minimize the total loss as:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda_1\mathcal{L}_{Clst} + \lambda_2\mathcal{L}_{Sep} + \lambda_3\mathcal{L}_{Coh} + \lambda_4\mathcal{L}_{Orth} \tag{6}$$

where $\mathcal{L}_{CE}$ is the cross entropy loss for classification and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyper-parameters.

Slots pruning: In this stage, our goal is to prune the sub-prototypes $\mathbf{p}_j^k$ that are dissimilar to the other sub-prototypes for each prototype. Intuitively, we aim to remove these sub-prototypes because sub-prototypes that are not similar will lead to prototypes with an inconsistent, unintuitive semantic meaning. We freeze all of the parameters in the model, except the slot indicator vectors $\mathbf{v}$. During this stage, we jointly optimize the coherence loss defined in Eq. 4 along with the cross entropy loss. We lower the coherence loss weight to avoid removing all the slots. Since the slot indicators are approximations of step functions using sigmoid functions with a sufficiently high temperature parameter τ, the indicator values during this phase are fractional but approach binary values close to 0 or 1. The loss in the stage is defined as:

$$\mathcal{L}_{\text{prune}} = \mathcal{L}_{CE} + \lambda_5\mathcal{L}_{Coh}. \tag{7}$$

Projection: In this stage, we first round the fractional indicator values to the nearest integer (either 1 or 0), and freeze the slot indicators' values. Then, we project each prototype $\mathbf{p}_j$ to the closest training image patch measured by the summed cosine similarity defined in Eq. 2, as in [10]. Because the latent feature tokens from the ViT encoder correspond to image patches, we do not need to use up-sampling techniques for visualizations, and the prototypes visualized in the bounding boxes represent the exact corresponding latent feature tokens that the prototypes are projected to.

As demonstrated in Theorem 2.1 from ProtoPNet [10], if the prototype projection results in minimal movement, the model performance is unlikely to change because the decision boundary remains largely unaffected. This is ensured by minimizing cluster and separation loss, as defined in Eq. 3. In practice, when prototypes are sufficiently well-trained and closely clustered to certain latent patches, the change in model performance after the projection step should be minimal. Additionally, this step ensures that the prototypes are inherently interpretable, as they are projected to the closest latent feature tokens, which have corresponding visualizations and provide a pixel space explanation of the "cases" in the model's reasoning process.

Optimization of last layers: Similar to other existing works [10, 46], we performed a convex optimization on the last evidence layer h after performing projection, while freezing all other parameters. This stage aims to introduce sparsity to the last layer weights W by penalizing the $1-$norm of layer weights $\mathbf{W_{b,l}}$ (initially fixed as -0.5) associated with the class b for the l-th class prototype $\mathbf{p}^l$, where $l \neq b$. By minimizing this loss term, the model is encouraged to use only positive evidence for final classifications. The loss term is defined as:

$$\mathcal{L}_h = \mathcal{L}_{CE} + \lambda_6 \sum_{l}^{C} \sum_{b \in \{0,\ldots,C\}: b \neq l} \|\mathbf{W_{b,l}}\|_1. \tag{8}$$

4 Experiments

4.1 Case Study 1: Bird Species Identification

To demonstrate the effectiveness of ProtoVit, we applied it to the cropped Caltech-UCSD Birds-200-2011 (CUB 200-2011) dataset [43]. This dataset contains 5,994/5,794 images for training and testing

https://doi.org/10.52202/079017-1312

Table 2: Comparison of ProtoVit implemented with DeiT and CaiT backbones to other existing works. Our model is not only inherently interpretable but also superior in performance compared to other methods using the same backbone. We also include models with a CNN backbone (Densenet-161) for reference in the top section. The reported accuracies are the final results after all training stages.

Arch.	Model	CUB Acc.[%]	Car Acc.[%]
Densenet-161 ~28.68M params	ProtoPNet (given in [10])	80.1 ± 0.3	89.5 ± 0.2
	Def. ProtoPNet(2x2) (given in [13])	80.9 ±0.22	88.7 ± 0.3
	ProtoPool (given in [36])	80.3 ± 0.3	90.0 ± 0.3
	TesNet (given in [46])	**81.5 ± 0.3**	**92.6 ± 0.3**
DeiT-Tiny ~5M params	Base (given in [50])	80.57	86.21
	ViT-Net (given in [50])	81.98	88.41
	ProtoPFormer (given in [50])	82.26	88.48
	ProtoViT(K=4, r=1) (ours)	**82.92 ± 0.5**	**89.02 ± 0.1**
DeiT-Small ~22M params	Baseline (given in [50])	84.28	90.06
	ViT-Net (given in [50])	84.26	91.34
	ProtoPFormer (given in [50])	84.85	90.86
	ProtoViT(K=4, r=1) (ours)	**85.37 ± 0.13**	**91.84 ± 0.3**
CaiT-XXS 24 ~11.9M params	Baseline (given in [50])	83.95	90.19
	ViT-Net (given in [50])	84.51	91.54
	ProtoPFormer (given in [50])	84.79	91.04
	ProtoViT(K=4, r=1) (ours)	**85.82 ± 0.15**	**92.40 ± 0.1**

from 200 different bird species. We performed similar offline image augmentation to previous work [10, 46, 26] which used random rotation, skew, shear, and left-right flipping. After augmentation, the training set had roughly 1,200 images per class. We performed prototype projections on only the non-augmented training data. Additionally, we performed ablation studies on the class token (See Appendix Sec. E.1), coherence loss (See Appendix Sec. E.3), and adjacency mask (See Appendix Sec. E.2). The quantitative results for the ablation can be found in Appendix Table. 6. We assigned the algorithm to choose 10 class-specific prototypes for each of the 200 classes. Each of the prototypes is composed of 4 sub-prototypes. Each set of sub-prototypes was encouraged to learn the 'key' features for its corresponding class through only the last layer weighting, and the 4 sub-prototypes were designed to collectively represent one key feature for the class. More discussion on different choices of K can be found in Appendix F. For more details about hyperparameter settings and training schedules, please refer to Appendix Sec. A and Appendix Sec. B respectively. Details and results of user studies regarding the interpretability of ProtoViT can be found in Appendix. Sec. K.

4.1.1 Predictive Performance Results

The performance of our ProtoViT with CaiT and DeiT backbones is compared to that of existing work in Table 2, including results from prototype-based models using DenseNet161 (CNN) backbones. Integrating ViT (Vision Transformer) backbones with a smaller number of parameters into prototype-based algorithms significantly enhances performance compared to those using CNN (Convolutional Neural Network) backbones, particularly on more challenging datasets. Compared with other prototype-based models that utilize ViT backbones, our model produces the **highest accuracy**, **outperforming** even the black box ViTs used as backbones, while offering **interpretability**. We provide accuracy using an ImageNet pretrained Densenet-161 to match the ImageNet pretraining used in all transformer models.

4.1.2 Reasoning Process and Analysis

Fig. 3 shows the reasoning process of ProtoViT for a test image of a Barn Swallow. Example visualizations for the classes with the top two highest logit scores are provided in the figure. Given this test image $\mathbf{x}$, our model compares its latent feature tokens against the learned sub-prototypes $\mathbf{p}_j^k$ through the greedy matching layer g. In the decision process, our model uses the patches that have the most similar latent feature tokens to the learned prototypes as evidences. In the example, our model correctly classifies a test image of a Barn Swallow and thinks the Tree Swallow is the second most likely class based on prototypical features. In addition to this example reasoning process, we conducted local and global analyses (shown in Fig. 4) to confirm the semantic consistency of prototypical features across all training and testing images, where local and global analyses are

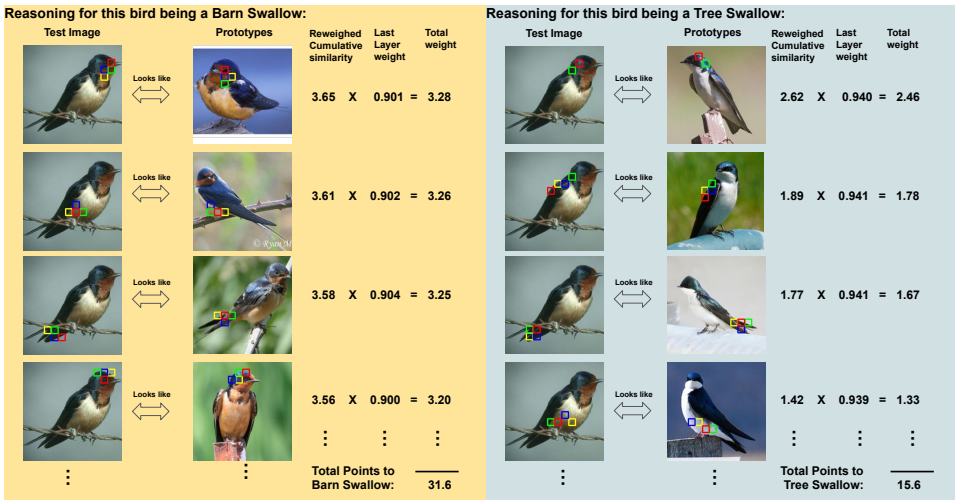

Figure 3: Reasoning process of how ProtoViT classifies a test image of Barn Swallow using the learned prototypes. Examples for the top two predicted classes are provided. We use the DeiT-Small backbone with r=1 and k=4 for the adjacency mask.

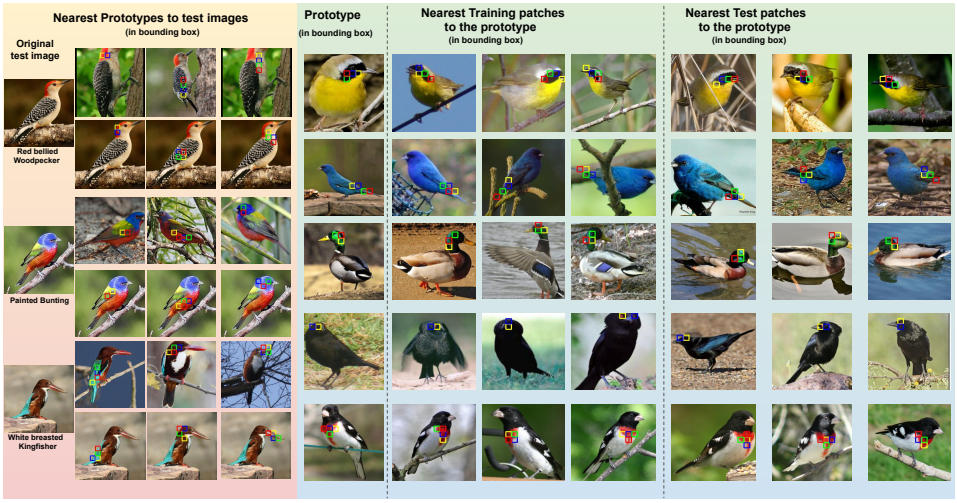

Figure 4: Nearest prototypes to test images (left), and the nearest image patches to prototypes (right). We exclude the nearest training patch, which is the prototype itself by projection.

defined as in [10]. The left side of Fig. 4 displays local analysis examples, visualizing the most semantically similar prototypes to each test image. The right side of Fig. 4 shows global analysis examples, presenting the top three nearest training and testing images to the prototypes. Our local analysis confirms that, across distinct classes and prototypes, the comparisons made are reasonable. For example, the first prototype compared to the white breasted kingfisher seems to identify the bird's blue tail, and is compared to the tail of the bird in the test image. Further, our global analysis shows that each prototype consistently activates on a single, meaningful concept. For example, the prototype in the first row of the global analysis consistently highlights the black face of the common yellowthroat at a variety of scales and poses. Taken together, these analyses show that **the prototypes of ProtoViT have strong, consistent semantic meanings**. More examples of the reasoning process and analysis can be found in Appendix Sec. H and Sec. J respectively.

4.1.3 Location Misalignment Analysis

Vision Transformers (ViTs) use an attention mechanism that blends information from all image patches, which may prevent the latent token at a position from corresponding to the input token at that

https://doi.org/10.52202/079017-1312

Table 3: Experimental results on the Location Misalignment Benchmark. We compare our ProtoViT (Deit-Small) with other prototype-based models with CNN backbones (ResNet34). We found that our model performs similarly to or better than the existing models with CNN backbones in terms of the misalignment metrics and test accuracy.

Method	PLC	PRC	PAC	Acc. Before	Acc. After	AC.
ProtoPNet	24.0 ± 1.7	13.5 ± 3.1	23.7 ± 2.8	76.4 ± 0.2	68.2 ± 0.9	8.2 ± 1.1
TesNet	16.0 ± 0.0	2.9 ± 0.3	3.4 ± 0.3	81.6 ± 0.2	75.8 ± 0.5	5.8 ± 0.6
ProtoPool	31.8 ± 0.8	4.5 ± 0.9	11.2 ± 1.3	80.8 ± 0.2	76.0 ± 0.3	4.8 ± 0.1
ProtoTree	27.7 ± 0.3	13.5 ± 3.1	23.7 ± 2.8	76.4 ± 0.2	68.2 ± 0.9	8.2 ± 1.1
ProtoViT(ours)	$\mathbf{21.68 \pm 3.1}$	$\mathbf{1.28 \pm 0.1}$	$\mathbf{2.92 \pm 0.1}$	$\mathbf{85.4 \pm 0.1}$	$\mathbf{82.8 \pm 0.3}$	$\mathbf{2.6 \pm 0.4}$

position. To assess whether ViT backbones can be interpreted as reliably as Convolutional Neural Networks (CNNs) in ProtoPNets, we conducted experiments using the gradient-based adversarial attacks described in the Location Misalignment Benchmark [37]. As summarized in Table 3, our method, which incorporates a ViT backbone, consistently matched or outperformed leading CNN-based prototype models such as ProtoPNet [10], ProtoPool [36], and ProtoTree [27] across key metrics: Percentage Change in Location (PLC), Percentage Change in Activation (PAC), and Percentage Change in Ranking (PRC), as defined in the benchmark. Lower values for these metrics indicate better performance. This shows that ProtoViT is at least as robust as CNN-based models. Moreover, as observed in the Location Misalignment Benchmark[37], the potential for information leakage also exists in deep CNNs, where the large receptive fields of deeper layers would encompass the entire image same as attention mechanisms. **While our model is not entirely immune to location misalignment, empirical results indicate that its performance is on par with other state-of-the-art CNN-based models.** Qualitative comparisons that further support this point can be found in Appendix. Sec. G, and more results and discussions on PLC for ablated models can be found in Appendix. Sec. E

4.2 Case Study 2: Cars Identification

In this case study, we apply our model to car identification. We trained our model on the Stanford Car dataset[24] of 196 car models. More details about the implementation can be found in Appendix. Sec. L. The performance with baseline models can be found in Table. 2. Example visualizations and analysis can be found in Appendix L. We find that **ProtoViT again produces superior accuracy and strong, semantically meaningful prototypes on the Cars dataset.**

5 Limitations

While we find our method can offer coherent and consistent visual explanations, it is not yet able to provide explicit textual justification to explain its reasoning process. With recent progress in large vision-language hybrid models [23, 11, 29], a technique offering explicit justification for visual explanations could be explored in future work. It is important to note that in some visual domains such as mammography or other radiology applications, features may not have natural textual descriptions – thus, explicit semantics are not yet possible, and a larger domain-specific vocabulary would first need to be developed. Moreover, as discussed in Sec. 4.1.3, our model is not completely immune to location misalignment, meaning the learned prototypical features may be difficult to visualize in local patches. However, this issue is not unique to our approach; CNN-based models face the same challenge. As layers deepen, the receptive field often expands to cover the entire image, leading to the same problem encountered with attention mechanisms in vision transformers.

6 Conclusion

In this work, we presented an interpretable method for image classification that incorprates ViT backbones with deformed prototypes to explain its predictions (*this* looks like *that*). Unlike previous works in prototype-based classification, our method offers spatially deformed prototypes that not only account for geometric variations of objects but also provide coherent prototypical feature representations with an adaptive number of prototypical parts. While offering inherent interpretability, our model empirically outperform the previous prototype based methods in accuracy.

7 Acknowledgement

We would like to acknowledge funding from the National Science Foundation under grants HRD-2222336 and OIA-2218063 and the Department of Energy under DE-SC0021358.

References

[1] Samira Abnar and Willem Zuidema. Quantifying Attention Flow in Transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity Checks for Saliency Maps. *Advances in Neural Information Processing Systems*, 31, 2018.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PloS one*, 10(7):e0130140, 2015.

[4] Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yinhao Ren, Joseph Y Lo, and Cynthia Rudin. A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nature Machine Intelligence*, 3(12):1061–1070, 2021.

[5] Richard Berk, Drougas Berk, and D Drougas. *Machine Learning Risk Assessments in Criminal Justice Settings*. Springer, 2019.

[6] Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is Attention Explanation? An Introduction to the Debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, 2022.

[7] Andrea Bontempelli, Stefano Teso, Katya Tentori, Fausto Giunchiglia, and Andrea Passerini. Concept-level debugging of part-prototype networks. *arXiv preprint arXiv:2205.15769*, 2022.

[8] Zachariah Carmichael, Suhas Lohit, Anoop Cherian, Michael J Jones, and Walter J Scheirer. Pixel-grounded prototypical part networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4768–4779, 2024.

[9] Hila Chefer, Shir Gur, and Lior Wolf. Transformer Interpretability Beyond Attention Visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791, 2021.

[10] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This Looks Like That: Deep Learning for Interpretable Image Recognition. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: Universal Image-TExt Representation Learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings*, pages 104–120. Springer, 2020.

[12] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[13] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. 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, 2022.

[14] Jon Donnelly, Luke Moffett, Alina Jade Barnett, Hari Trivedi, Fides Schwartz, Joseph Lo, and Cynthia Rudin. AsymMirai: Interpretable Mammography-based Deep Learning Model for 1–5-year Breast Cancer Risk Prediction. *Radiology*, 310(3):e232780, 2024.

 https://doi.org/10.52202/079017-1312

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing Higher-Layer Features of a Deep Network. *University of Montreal*, 1341(3):1, 2009.

[17] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding Deep Networks via Extremal Perturbations and Smooth Masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2950–2958, 2019.

[18] Wei Gao, Fang Wan, Xingjia Pan, Zhiliang Peng, Qi Tian, Zhenjun Han, Bolei Zhou, and Qixiang Ye. TS-CAM: Token Semantic oupled Attention Map for Weakly Supervised Object Localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2886–2895, 2021.

[19] Saurav Gupta, Sourav Lakhotia, Abhay Rawat, and Rahul Tallamraju. ViTOL: Vision Transformer for Weakly Supervised Object Localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4101–4110, 2022.

[20] Maksims Ivanovs, Roberts Kadikis, and Kaspars Ozols. Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters*, 150:228–234, 2021.

[21] Sarthak Jain and Byron C Wallace. Attention is not Explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[22] Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. ViT-NeT: Interpretable Vision Transformers with Neural Tree Decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR, 2022.

[23] Wonjae Kim, Bokyung Son, and Ildoo Kim. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

[24] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[25] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2801–2807, 2019.

[26] Chiyu Ma, Brandon Zhao, Chaofan Chen, and Cynthia Rudin. This Looks Like Those: Illuminating Prototypical Concepts Using Multiple Visualizations. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Meike Nauta, Ron Van Bree, and Christin Seifert. 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, 2021.

[28] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[29] Giang Nguyen, Mohammad Reza Taesiri, and Anh Nguyen. Visual correspondence-based explanations improve AI robustness and human-AI team accuracy. *Neural Information Processing Systems (NeurIPS)*, 2022.

[30] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-Aware Redundancy Reduction for Vision Transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911, 2021.

[31] DIPANJYOTI Paul, Arpita Chowdhury, Xinqi Xiong, Feng-Ju Chang, David Edward Carlyn, Samuel Stevens, Kaiya L Provost, Anuj Karpatne, Bryan Carstens, Daniel Rubenstein, et al. A Simple Interpretable Transformer for Fine-Grained Image Classification and Analysis. In *The Twelfth International Conference on Learning Representations*, 2023.

[32] Sebastian Ramos, Stefan Gehrig, Peter Pinggera, Uwe Franke, and Carsten Rother. Detecting Unexpected Obstacles for Self-Driving Cars: Fusing Deep Learning and Geometric Modeling. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1025–1032. IEEE, 2017.

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

[34] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16:1–85, 2022.

[35] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. ProtoPShare: Prototypical Parts Sharing for Similarity Discovery in Interpretable Image Classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1420–1430, 2021.

[36] Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable Image Classification with Differentiable Prototypes Assignment. In *European Conference on Computer Vision*, pages 351–368. Springer, 2022.

[37] Mikołaj Sacha, Bartosz Jura, Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Interpretability benchmark for evaluating spatial misalignment of prototypical parts explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 21563–21573, 2024.

[38] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv:1312.6034*, 2013.

[39] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[41] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going Deeper with Image Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[44] Chong Wang, Yuanhong Chen, Yuyuan Liu, Yu Tian, Fengbei Liu, Davis J McCarthy, Michael Elliott, Helen Frazer, and Gustavo Carneiro. Knowledge distillation to ensemble global and interpretable prototype-based mammogram classification models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 14–24. Springer, 2022.

[45] Chong Wang, Yuyuan Liu, Yuanhong Chen, Fengbei Liu, Yu Tian, Davis McCarthy, Helen Frazer, and Gustavo Carneiro. Learning support and trivial prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2062–2072, 2023.

 https://doi.org/10.52202/079017-1312

[46] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable Image Recognition by Constructing Transparent Embedding Space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 895–904, 2021.

[47] Thierry Warin and Aleksandar Stojkov. Machine learning in finance: a metadata-based systematic review of the literature. *Journal of Risk and Financial Management*, 14(7):302, 2021.

[48] Sarah Wiegreffe and Yuval Pinter. Attention is not not Explanation. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pages 11–20. Association for Computational Linguistics, 2019.

[49] Yang Xu and Zuqiang Meng. Interpretable vision transformer based on prototype parts for COVID-19 detection. *IET Image Processing*, 2024.

[50] Mengqi Xue, Qihan Huang, Haofei Zhang, Lechao Cheng, Jie Song, Minghui Wu, and Mingli Song. ProtoPFormer: Concentrating on Prototypical Parts in Vision Transformers for Interpretable Image Recognition. *arXiv preprint arXiv:2208.10431*, 2022.

[51] Haoming Yang, Steven Winter, Zhengwu Zhang, and David Dunson. Interpretable ai for relating brain structural and functional connectomes. *arXiv preprint arXiv:2210.05672*, 2022.

[52] Haoming Yang, Pramod KC, Panyu Chen, Hong Lei, Simon Sponberg, Vahid Tarokh, and Jeffrey Riffell. Neuron synchronization analyzed through spatial-temporal attention. *bioRxiv*, pages 2024–07, 2024.

[53] Tingyi Yuan, Xuhong Li, Haoyi Xiong, Hui Cao, and Dejing Dou. Explaining Information Flow Inside Vision Transformers uUing Markov Chain. In *eXplainable AI approaches for debugging and diagnosis.*, 2021.

[54] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable ConvNets v2: More Deformable, Better Results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019.

Appendix Table of Contents

https://doi.org/10.52202/079017-1312

A Training Hyperparameters

This section documents the specific hyper-parameters used to train ProtoViT, shown in Table 4. We used identical parameter settings across all the backbones. We used the values suggested in prior work for terms that already existed, and used a small grid search to select the other values. No dedicated tuning procedure is necessary for these coefficients.

B Training schedule

This section documents the specific training schedule used to train ProtoViT. As discussed in Sec. 3.4, the training stages are generally divided into four steps as shown in Appendix Fig. 5. Training schedule settings can be found in Appendix Table 5. Warm-up optimization is performed by training the feature encoder with a extremely small learning rate while training the prototype vectors for 5 epochs. Then, we increase the learning rate for the feature encoder layer for the following 10 epochs. As mentioned in Appendix Sec. A, the number of sub-prototypes being pruned during the slots pruning stage depend on the learning rate of the slots parameters and the weight of the coherence loss in the slots pruning stage.

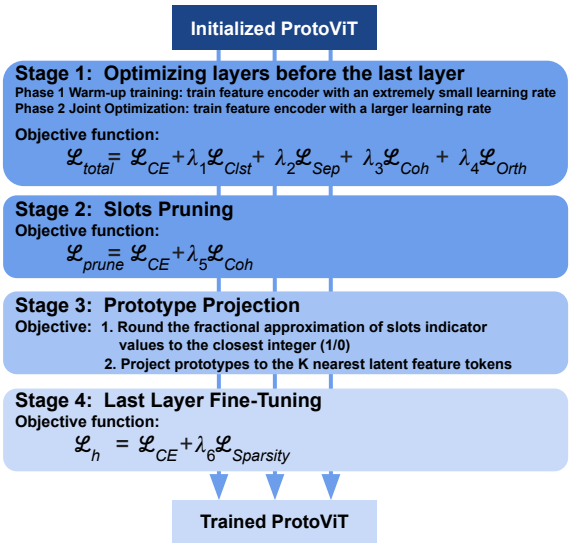

Figure 5: The procedure for training ProtoViT. The objectives for each stage are shown.

Table 4: Parameter Settings for ProtoViT

Parameter	Weight
Cross Entropy Weights	1.0
Cluster Loss	-0.8
Separation Loss	0.09
L_1 Norm Loss	1×10^{-2}
Orthogonality Loss	1×10^{-3}
Coherence Loss	3×10^{-3}
Coherence Loss for Slots Pruning	5×10^{-5}
Sigmoid Temperature τ	100

https://doi.org/10.52202/079017-1312

Table 5: Training Schedule for ProtoViT

Training Stage	Model Layers	Learning Rate	Duration
Optimization Warm-up Stage	feature encoder f	1×10^{-7}	
	prototype layer p	3×10^{-3}	5 epochs
Optimization Joint Stage	feature encoder f	5×10^{-5}	
	prototype layer p	3×10^{-3}	10 epochs
Slots Pruning Stage	slots parameters	8×10^{-5}	10 epochs
Fine-tuning Stage	last layer	1×10^{-4}	15 epochs

C More discussion on the greedy matching layer

Intuitively, a non-deformed prototype $\mathbf{p}_j$ consists of K non-overlapping sub-prototypes $\mathbf{p}_j^k$ with rigid adjacency (i.e., a rectangular shape). These non-deformed prototypes are compared with K non-overlapping latent feature tokens $\mathbf{z}_f^i$ in the same geometric shape (i.e., in a rectangle). To deform such a prototype, we could treat each sub-prototype vector $\mathbf{p}_j^k$ as an independent "prototype" to train. Then, the prototype $\mathbf{p}_j$ is naturally deformed into K independent sub-prototypes $\mathbf{p}_j^k$ that move freely in the latent space. However, this naïve approach could lead to significant overlap among the sub-prototypes. Although this could be mitigated by incorporating a unit-wise orthogonality loss during the training phase to encourage dissimilarity among the sub-prototypes, similar to the orthogonality loss outlined in Eq. 5, such a unit-wise orthogonality loss is in conflict with the objectives of the coherence loss defined in Eq. 4, which encourages the sub-prototypes $\mathbf{p}_j^k$ to remain neighboring in cosine distance to preserve the semantic coherence of each prototype $\mathbf{p}_j$. Thus, an algorithm that could provide unit-wise and non-overlapped matching is ideal, and the greedy matching algorithm with adjacency masking meets the needs. An summary of the algorithm for greedy matching with adjacency masking is shown in Alg. 1.

Algorithm 1 Greedy Distance Matching Algorithm

Require: Input latent feature tokens $\mathbf{z}$, and prototypes $\mathbf{p}$,
1: Compute cosine similarity between latent feature tokens and k sub-prototypes.
2: Initialize masks in shape of latent feature tokens, and sub-prototypes to include all possible pairs
3: **for** k in K slots of sub-prototypes **do**
4: Mask out non-adjacent and selected patches by replacing their similarity scores with a high negative value
5: Identify the closest latent feature token for each remaining sub-prototype
6: Select the pair that has the highest cosine similarity out of all identified pairs
7: Update all masks based on the selected pair
8: Track the sequence of selected sub-prototypes for reordering
9: **end for**
10: Sort the pairs and the corresponding cosine similarity based on its original sub-prototype order

D More discussions on adaptive slots mechanism

As discussed in the Sec. 3.3, The adaptive slots mechanism consists of learnable vectors $\mathbf{v} \in \mathbb{R}^{m \times K}$ and a sigmoid function with a hyper-parameter temperature τ. The vectors $\mathbf{v}$ are sent to the sigmoid function to approximate the indicator function as $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}} = \textbf{Sigmoid}(\mathbf{v}_j^k, \tau)$. Each $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}}$ is an approximation of the indicator for whether the k-th sub-prototype will be included in the j-th prototype $\mathbf{p}_j$. To be specific, we define the approximated indication function $\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}}$ with temperature τ as:

$$\tilde{\mathbb{1}}_{\{\text{Include } \mathbf{p}_j^k\}} = \frac{1}{1 + e^{-\mathbf{v}_j^k \tau}} \tag{9}$$

By the range of the sigmoid function, we are able to approximate the indicator function with a high temperature τ. Fig. 6 shows an example of different choices of τ. As the temperature value goes up, the sigmoid function more closely approximates the behavior of the indicator function. Thus, we

https://doi.org/10.52202/079017-1312

picked a sufficiently large τ for approximation, and later rounded the values to the closest integer (i.e 0 or 1) to serve as the actual slot indicator for each sub-prototype.

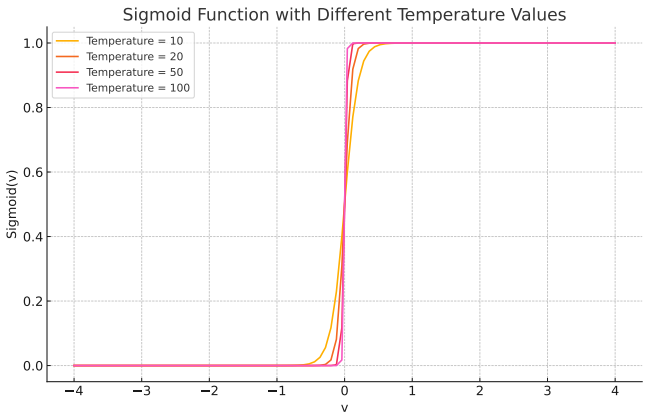

Figure 6: Plot of sigmoid function with different choices of temperature value τ

E Ablation studies

This section details ablation studies on the class token, coherence loss, and adjacency mask. We performed the ablation studies with backbone DeiT-Small with $r = 1$ and $K = 4$ for the adjacency mask and number of sub-prototypes respectively.

Table 6: Ablation study on ProtoViT. Model is compared with $K = 4$, and $r = 1$ for slots and adjacency mask settings. We use Deit-Small as the backbone, and the model is trained on CUB-200-2011 dataset. For the current setting of ProtoViT, we included class token, coherence loss, and adjacency mask along with the greedy matching algorithm.

greedy matching	class token	coherence loss	adjacency mask	Accuracy [%]	PLC [%]
Yes	No	No	No	84.32 ± 0.10	31.02 ± 0.1
Yes	Yes	No	No	85.54 ± 0.20	35.88 ± 0.8
Yes	Yes	Yes	No	85.13 ± 0.17	32.98 ± 0.6
Yes	Yes	No	Yes	85.45 ± 0.30	31.67 ± 0.2
Yes	No	Yes	Yes	84.12 ± 0.05	17.83 ± 1.5
Yes	**Yes**	**Yes**	**Yes**	$\mathbf{85.37 \pm 0.13}$	$\mathbf{21.68 \pm 3.1}$

E.1 Class-token

In this section, we performed an ablation study on the class token. In the ablated model ProtoViT (patch tokens only), the latent feature tokens $\mathbf{z}_f$ are defined as the latent patch tokens $\mathbf{z}_{patch}$. The class token $\mathbf{x}_{class}$ and its corresponding component in position embedding $\mathbf{E}_{\text{pos}}$ are excluded in training. As shown in Appendix Table. 6, by incorporating the class token into the latent feature tokens, the performance of the model generally increased by 1% with and without the adjacency mask and coherence loss. Though we observe a drop in performance by excluding the class token and the corresponding component in the position embedding, the ablated model still achieves comparable performance to the other prototype-based ViTs and the black-box backbone.

Saliency test: Subtracting out the class token may have interesting implications for the class specificity of our prototypes. We expect subtracting the class token out operates as a kind of focal similarity, where each token (after taking the difference) represents the unique information available in that position which is relevant to the target class. Prototypes will, then, learn to represent unique positional information that is relevant to the predicted class. As such, we might expect prototypes

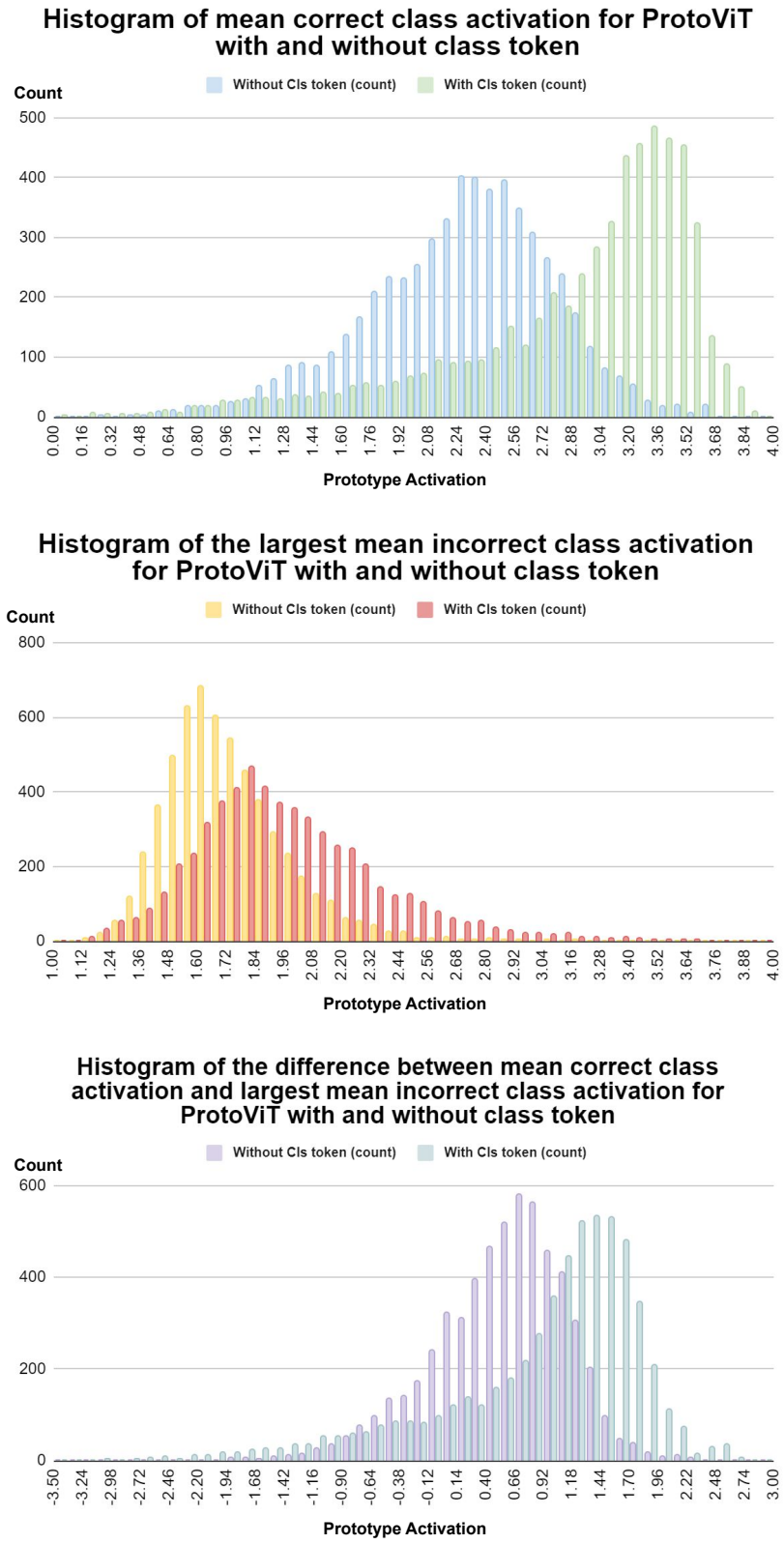

Figure 7: Histogram analysis of mean activation for ProtoViT with and without class token.

https://doi.org/10.52202/079017-1312

to be more tightly tied to their class than they would be if they were learning with simple arbitrary tokens. We thus expect an improvement in class-specific saliency of the prototypes.

To test if class token improved the saliency of prototype to its assigned class, we performed the analysis on correct and incorrect class similarity scores. If prototypes are more specific to their assigned class when including the class token, we would expectthe gap between the mean activation for prototypes of the correct class (summed cosine similarity scores as defined in Eq. 2) and the largest mean activation for prototypes of an incorrect class to be larger. To test this, we randomly pick a model with the class token and one without the class token and evaluate each model on the test set of CUB200-2011[43]. For each test image, we average over the correct class prototypes activations, and denote this average as $\mathbf{g}^{\text{abl}}_{\text{cor}}$ for the model trained with patch-tokens only, and $\mathbf{g}^{\text{cls}}_{\text{cor}}$ for the model trained with class tokens. Similarly, we denote the largest mean activations from the incorrect class prototypes as $\mathbf{g}^{\text{cls}}_{\text{incor}}$, and $\mathbf{g}^{\text{abl}}_{\text{incor}}$ for the models with and without class token respectively. We use δ^{cls} and δ^{abl} to denote the difference between the two measures for the two models. As shown in Appendix Fig. 7, there is a clear shift in distribution between the two models for all three measures. When we include the class token, prototypes tend to activate more highly on the correct class, and the gap between the mean correct class activation and the highest mean incorrect class activation tends to be larger.

We further perform one sided t-tests to test whether $\mathbf{g}^{\text{cls}}_{\text{cor}} - \mathbf{g}^{\text{abl}}_{\text{cor}}$ and $\delta^{\text{cls}} - \delta^{\text{abl}}$ are significantly greater than 0. As shown in Appendix Table. 7, including the class token statistically significantly increases each measure relative to a model trained with patch token only. That is, including the class token improves the saliency of encoded features.

E.2 Adjacency mask

As introduced in Sec. 3.3, the adjacency mask is designed to ensure that sub-prototypes are geometrically adjacent. Without the adjacency mask, sub-prototypes are able to match with latent feature tokens from anywhere in the image.As indicated in Appendix Table 6, removing the adjacency mask typically results in a slight improvement in performance. This outcome is expected since prototypes are learned for the maximum possible performance with fewer constraints. However, removing the adjacency mask tends to damage the semantics of the model's prototypes. For instance, as demonstrated in Appendix Fig. 8, without the adjacency mask, the model might correctly identify the beak of a least tern in one image patch but erroneously relate other prototypical sub-parts to the bird's feet in another patch. Although the slim feet of the least tern indeed looks similar to the slim beak on the scale of the sub-prototype, such comparisons are not meaningful. In contrast, implementing the adjacency mask in ProtoViT effectively prevents such issues by enforcing geometric adjacency among the sub-prototypes. Furthermore, without the adjacency mask, the model may learn sub-prototypes that, though visually similar to other sub-prototypes, are fundamentally different, introducing noisy representations and disrupting the coherence of prototypical features. For instance, as demonstrated in Appendix Fig. 8, both models misclassified a test image of a black tern as a pigeon guillemot which also has red feet. Without the adjacency mask, the prototypical feature capturing the red feet of the pigeon guillemot also mistakenly includes reflections of the red feet in the water. Although the water reflection looks similar to the actual red feet captured by the other sub-prototypes, this misrepresentation undermines the coherence of the feature representation. On the other hand, ProtoViT compares the feet of the black tern to the red feet of the pigeon guillemot. Thus, the use of an adjacency mask is essential for preserving the coherence of prototypical feature representations.

Table 7: Results of the t-test over correct class activation with and without class token, and the difference between the correct and incorrect class activation with and without class token. The p-values for both tests are ~ 0.

Target	Mean $\pm$1.96 Std.	T-stats against 0
Correct Class Activation ($\mathbf{g}^{\text{cls}}_{\text{cor}}$ - $\mathbf{g}^{\text{abl}}_{\text{cor}}$)	0.625 ± 0.011	116.3
Correct and Incorrect Class Activation Difference (δ^{cls} -δ^{abl})	0.375 ± 0.008	47.1

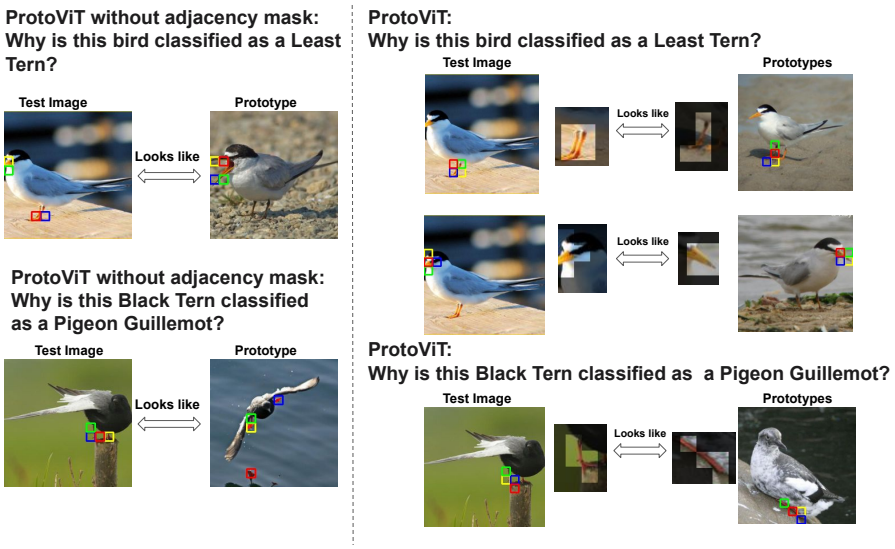

Figure 8: How ProtoViT without adjacency mask makes predictions (left) vs. how ProtoViT makes predictions. The test image of a least tern is correctly classified by the two models, and the test image of a black tern is classified as a pigeon guillemot by both of the models.

E.3 Coherence loss

As explained in Eq. 4, coherence loss is used to ensure that the sub-prototypes within each prototypical feature are semantically similar to each other. By design, this loss term helps sub-prototypes to collectively represent a coherent feature. By removing the coherence loss, the prototypical features would still contain diverse parts of features in one prototypical feature, a similar problem that the non-deformed prototypes have, even though the sub-prototypes remain geometrically adjacent. For example, as shown in Appendix Fig. 9, a model trained without coherence loss tends to mix the wing and belly of the Myrtle Warbler into one prototypical feature, while it mixes the head and wing for the prototypical features for Painted Bunting. In comparison, training ProtoViT with coherence loss ensures that all the sub-prototypes collectively represent one prototypical feature, in this case containing the striped belly of the Myrtle Warbler, or the wing and the red lower part of the Painted Bunting.

E.4 Ablated Misalignment

We computed the Prototype Location Change (PLC) for each ablated model using greedy matching algorithms. Following the PLC metric defined in Sacha et al. (2024) [37], we measured the shift in the 90th percentile of prototype activations across test images. As shown in Table 6, the combination of coherence loss and adjacent masks effectively reduces changes in prototype locations after adversarial attacks. This indicates that these mechanisms promote greater stability in the learned prototypes. Interestingly, incorporating the class token into the latent representations increases PLC. This is expected, as the class token contains more global information, which, while improving overall model performance, leads to larger shifts in prototype locations.

F Choices of K sub-prototypes

Why K=4 : By the design of the vanilla ProtoPNet[10] and the other non-deformed CNN based prototype models[46, 15, 26], the input images are encoded into latent features z by CNN backbones, where $z \in \mathbf{R}^{7\times7\times d}$ and d denotes the latent dimension varying by different choices of backbones. These models learn prototypical features p of shape $1 \times 1 \times d$ from the latent features. On the other hand, ViT backbones such as CaiT and DeiT encode the input images into $14 \times 14 \times d$ latent feature tokens. To be consistent with existing models, we design the model to learn prototypes of shape $2 \times 2 \times d$ from the $\mathbf{R}^{14\times14\times d}$ latent features, so that each prototype corresponds to the same proportion of the input image. Using the greedy matching algorithm, we can move away from

https://doi.org/10.52202/079017-1312

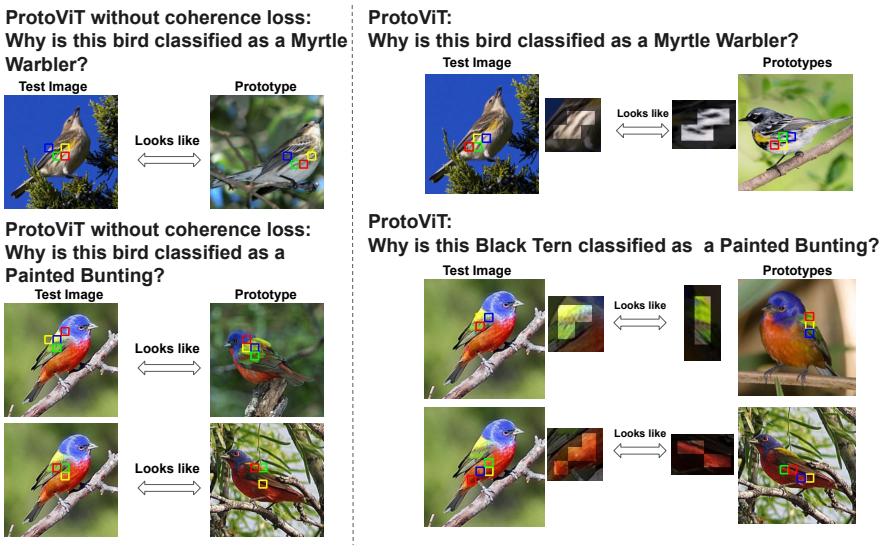

Figure 9: How ProtoViT without coherence loss make predictions (left) vs. how ProtoViT makes predictions. The given test image of Myrtle Warbler and Painted Bunting are correctly classified by both models.

learning fixed 2×2 rectangular prototypes and instead have **adaptively shaped prototypes** consisting of four sub-prototypes. It is worth noting that prototype-based models with CNN backbones heavily rely on up-sampling for prototype visualizations. This process can introduce errors, leading to the use of the top 5% most similar regions for visualization, which results in irregularly sized bounding boxes, as shown in the top row of Fig. 1. In contrast, with ViT backbones, we are able to visualize the exact image patches that the prototype projected to, and thus produce more precise and accurate prototype visualizations.

Other choices of K : Although we selected K= 4 to maintain consistency with other existing prototype-based models, alternative values of K are also viable. The model performance for $K = 5$ and $K = 6$ settings can be found in Appendix Table. 8. We observed that the model performs best with $r = 2$ when $K = 5$ is used, and with $r = 3$ when $K = 6$. As shown in the table, the models with different choices of K perform similarly to the setting with $K = 4$. Appendix Fig. 10 and Fig. 11 show examples of reasoning process for ProtoViT($K = 5, r = 2$) and ProtoViT($K = 6, r = 3$) respectively. As shown in the figures, though having more sub-prototypes is helpful to capture larger featuires such as the tail of the Lincoln Sparrow and Forster Tern in Fig. 10 and Fig. 11 respectively, because of variations in scale, many features such as the beak and the head in 11 do not always need that many sub-prototypes. Local and global analysis for ProtoViT($K = 5, r = 2$) and ProtoViT($K = 6, r = 3$) are shown in Fig. 13 and Fig. 13 respectively. As shown in the figures, having more sub-prototypes is advantageous in representing features that are large in scale such as the white belly of Sayornis shown in the second row of global analysis in Fig. 12, and brown pattern of Eastern Towhee in the bottom row of Fig. 13. Both the local and global analysis again show that **the prototypes of ProtoViT have strong, consistent semantic meanings.**

It is worth noting that prior prototype-based models could not easily support prototypes with 5 or 6 sub-prototypes. Since methods like Deformable ProtoPNet treat prototypes as convolutional features, the only way to handle prototypes with 5 sub-prototypes would be to form each prototype as a 1 by 5 dimensional convolutional filter. This would allow the model to have 5 sub-prototypes, but would enforce a very strange shape on prototypes (a horizontal line).

G Qualitative examples of robustness against perturbations

We provide several instances of the perturbation examples, in which we mask out the region selected by each prototype, as shown in Fig.14. In each row, we mask out all matched locations for a prototype (middle-left column) using a white mask on the black wings and a black mask on the red parts,

Table 8: ProtoViT performance with $K = 5$ and $K = 6$ using DeiT-Small backbone

# of sub-prototypes	Adjacency mask range R	Accuracy [%]
4	1	85.37 ± 0.13
5	2	85.33 ± 0.20
6	3	85.26 ± 0.15

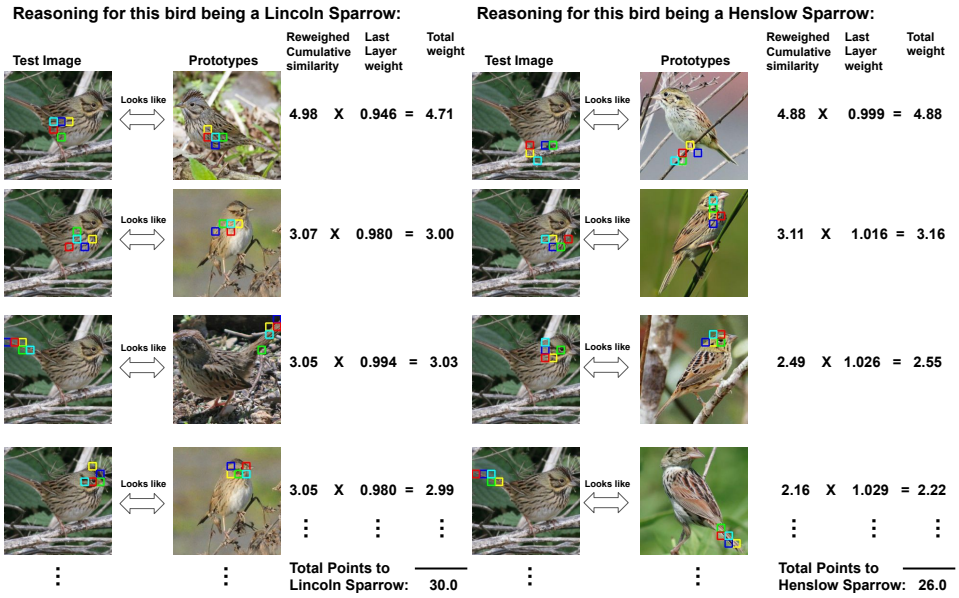

Figure 10: How ProtoViT with $K = 5$ and $r = 2$ using DeiT-Small backbone classifies a test image of a Lincoln Sparrow to the correct class (left), and the second most likely class Henslow Sparrow (right).

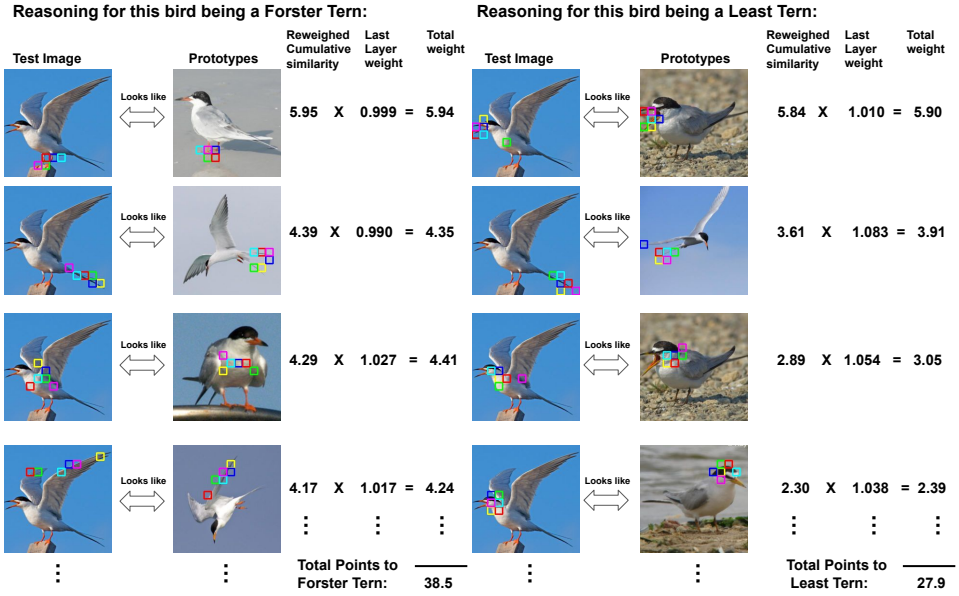

Figure 11: How ProtoViT with $K = 6$ and $r = 3$ using DeiT-Small backbone classifies a test image of a Forster Tern to the correct class (left), and the second most likely class Least Tern (right).

and check where that prototype activates after masking (shown in the leftmost column). We then confirm that the activated region for other prototypes remains reasonable when the mask from another

https://doi.org/10.52202/079017-1312

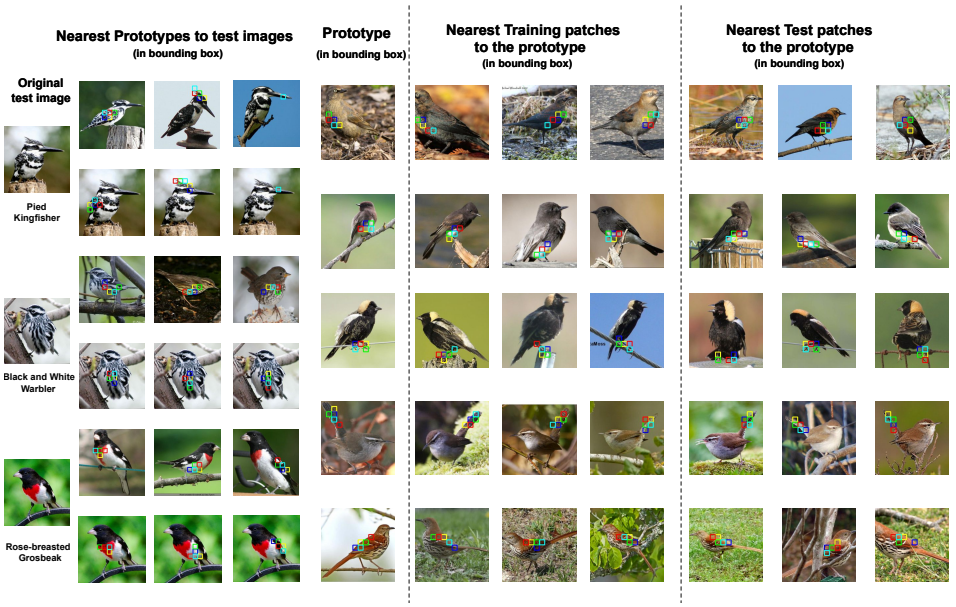

Figure 12: Examples of local analysis (left) and global analysis (right) of ProtoViT with DeiT-Small backbone with $K = 5$ and $r = 2$.

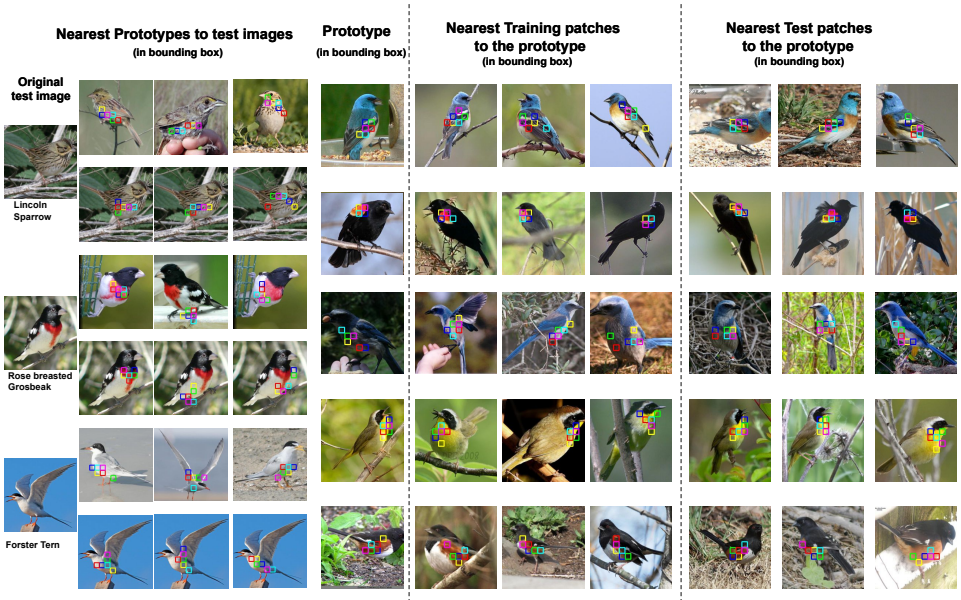

Figure 13: Examples of local analysis (left) and global analysis (right) of ProtoViT with DeiT-Small backbone with $K = 6$ and $r = 3$.

https://doi.org/10.52202/079017-1312

prototype is applied (right two columns). We observed that after removing the preferred region by each prototype, it activates on another reasonable alternative (e.g., a red belly prototype might activate on a red back as a second choice).

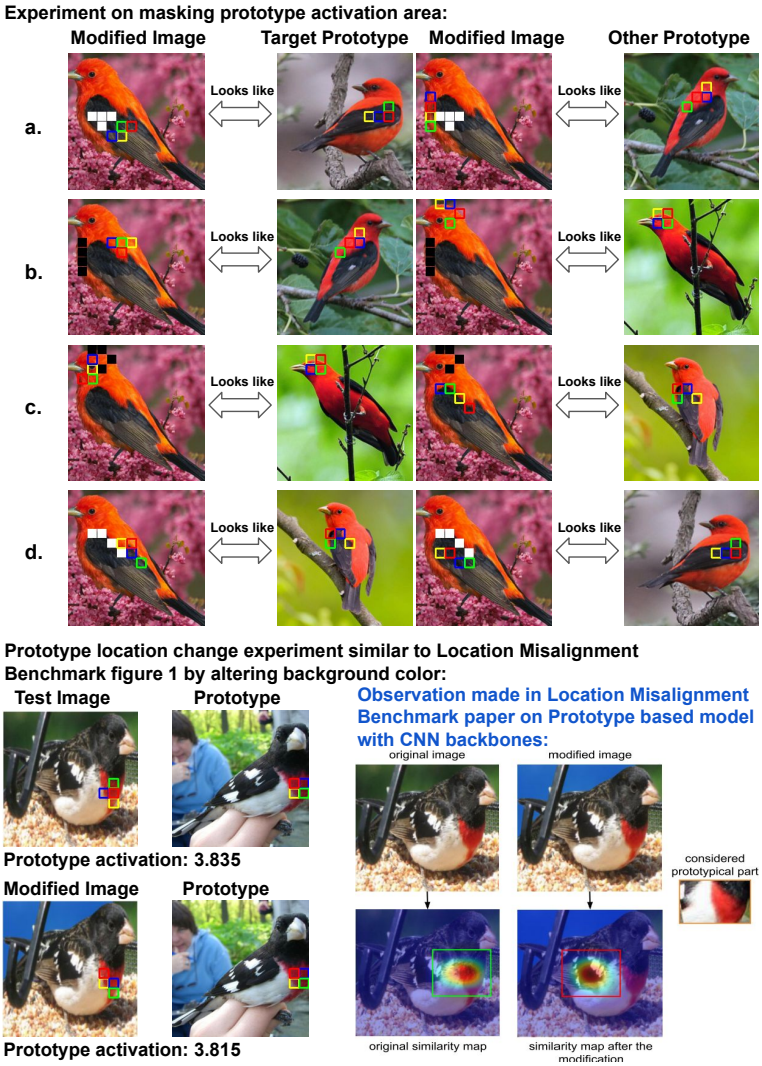

Figure 14: Perturbations (top) and background change (bottom). As shown in the top, our prototypes identify reasonable alternative regions on which to activate after masking the originally most activated regions. As shown in the bottom, altering the background does not substantially impact the activation location of our ViT-based prototype. This shows that our ProtoViT has a better location alignment than the CNN-based prototype models.

H More examples of reasoning process

This section provides more examples for the model reasoning process. Fig. 15, Fig. 16, and Fig. 17 demonstrate more examples of the reasoning process of ProtoViT with DeiT-Small backbone. Fig. 18, Fig. 19 and 20 are the examples of reasoning process of ProtoViT with CaiT-backones. Fig. 21, Fig. 22 and Fig. 23 are the examples of reasoning process of ProtoViT with Deit-Tiny backbone. In each case, we again see intuitive reasoning from ProtoViT.

https://doi.org/10.52202/079017-1312

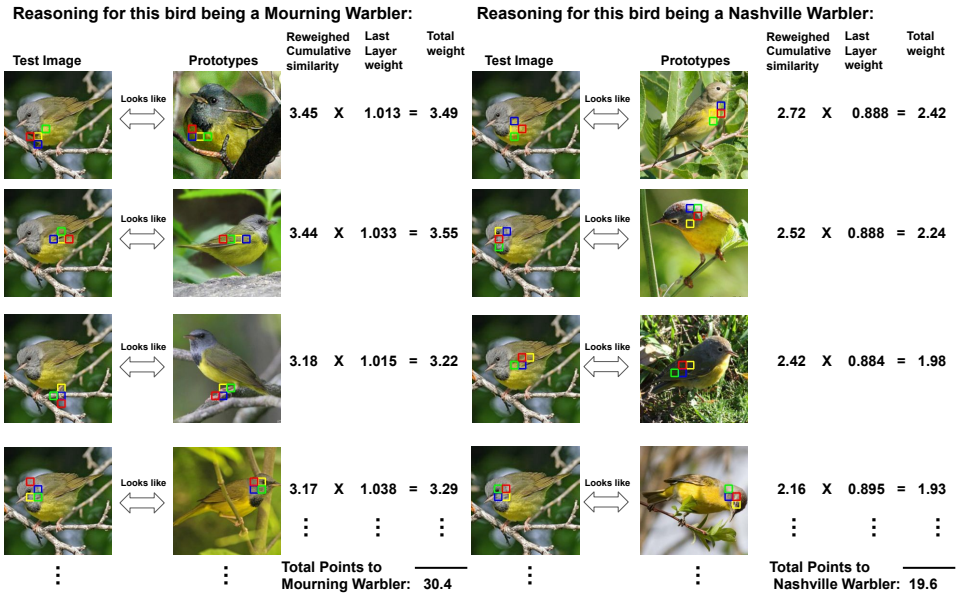

Figure 15: How ProtoViT with DeiT-Small backbone classifies a test image of a Mourning Warbler to the correct class (left), and the second most likely class Nashville Warbler (right).

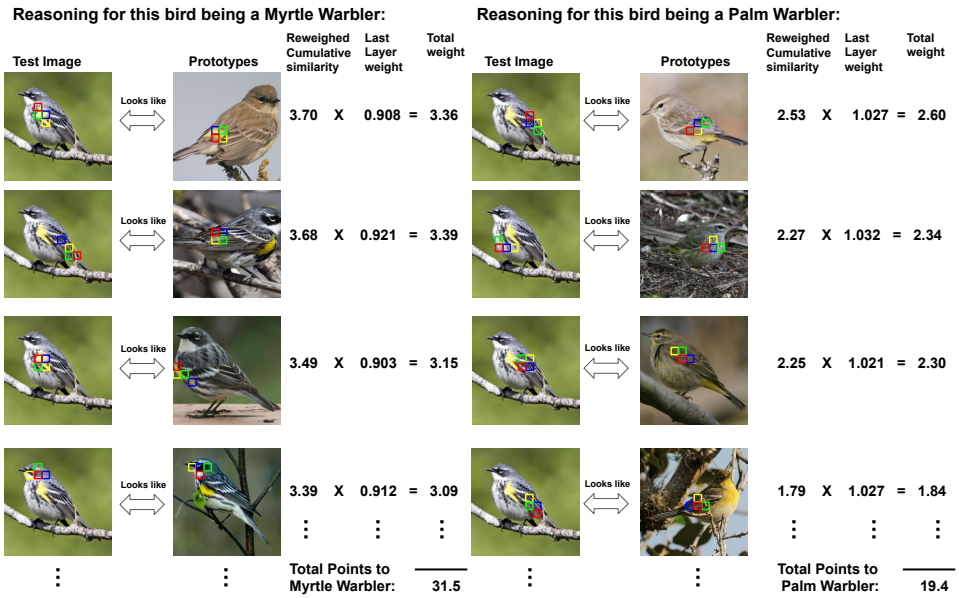

Figure 16: How ProtoViT with DeiT-Small backbone classifies a test image of a Myrtle Warbler to the correct class(left), and the second most likely class Palm Warbler (right).

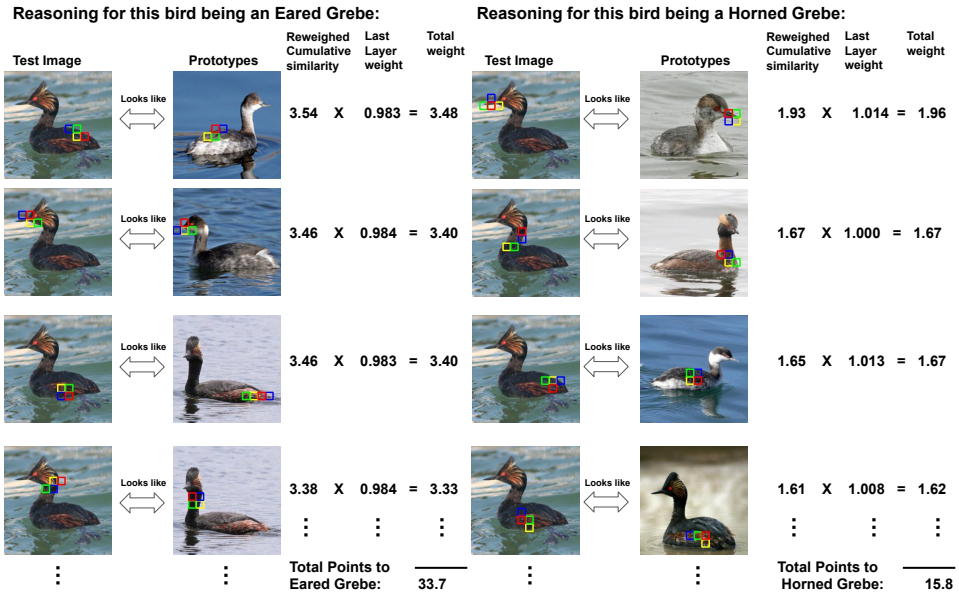

Figure 17: How ProtoViT with DeiT-Small backbone classifies a test image of an Eared Grebe to the correct class (left), and the second most likely class Horned Grebe (right).

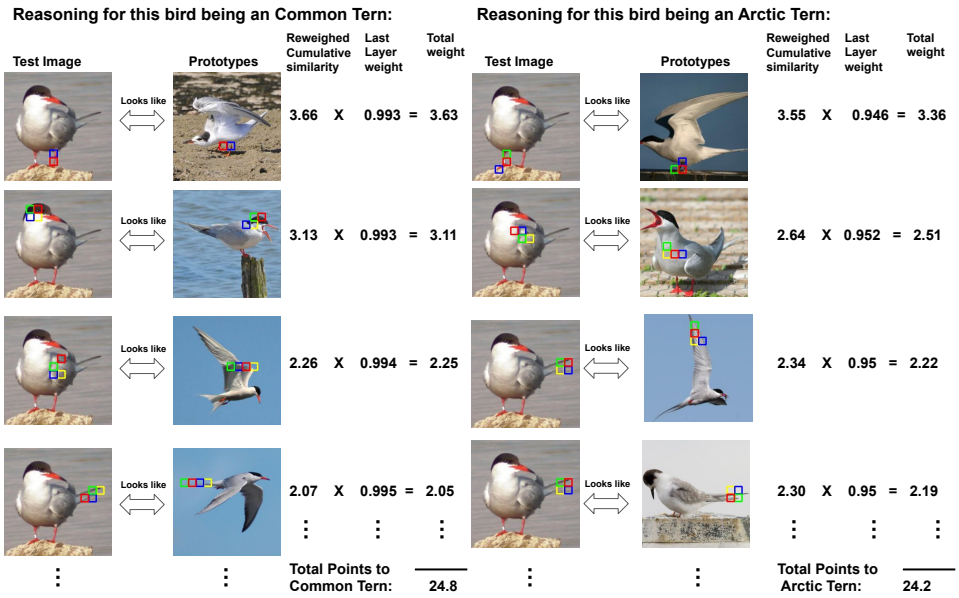

Figure 18: How ProtoViT with CaiT-xxs backbone classifies a test image of a Common Tern to the correct class (left), and the second most likely class Arctic Tern (right).

https://doi.org/10.52202/079017-1312

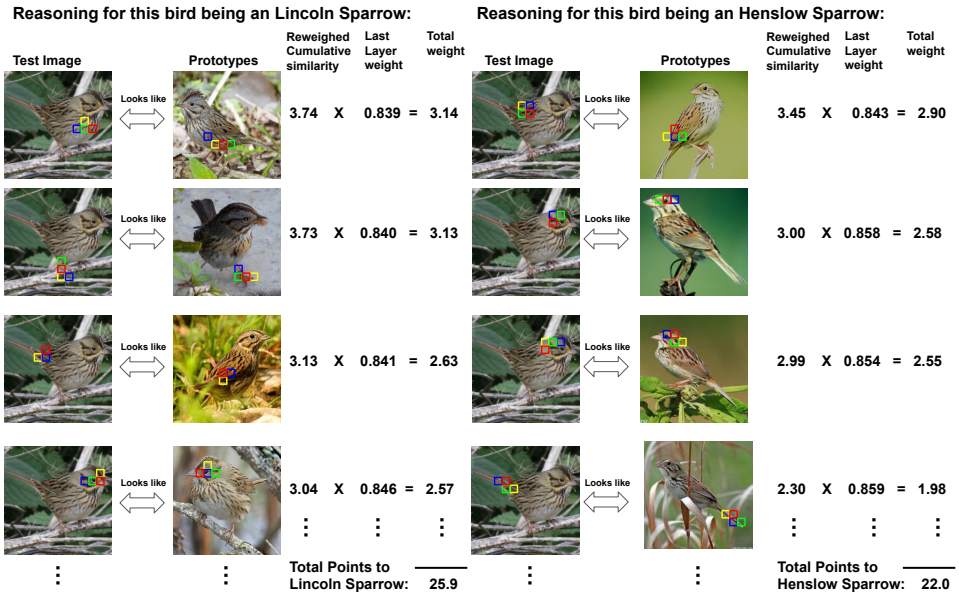

Figure 19: How ProtoViT with CaiT-xxs backbone classifies a test image of a Lincoln Sparrow to the correct class (left), and the second most likely class Henslow Sparrow (right).

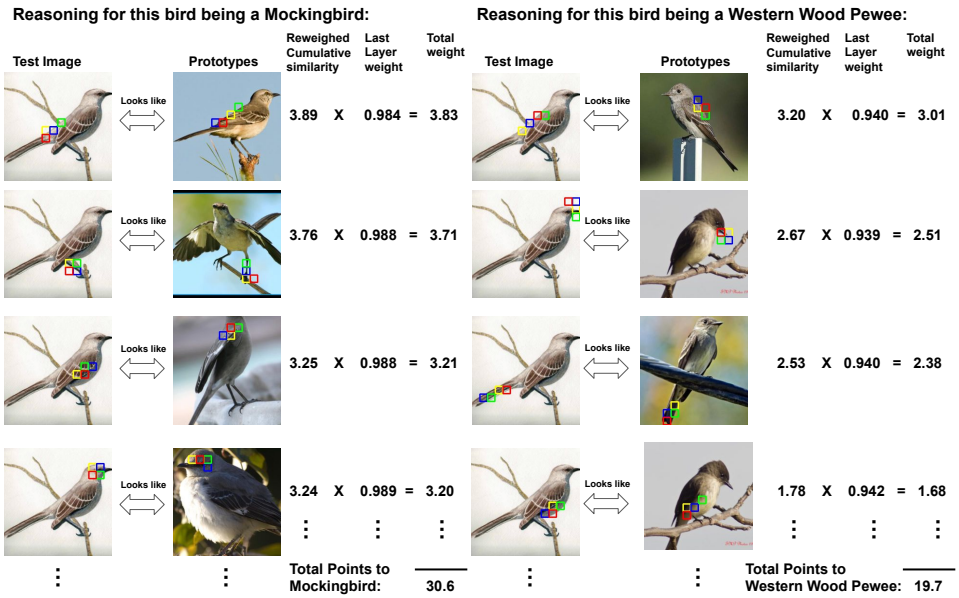

Figure 20: How ProtoViT with CaiT-xxs backbone classifies a test image of a Mockingbird to the correct class (left), and the second most likely class Western Wood Pewee (right).

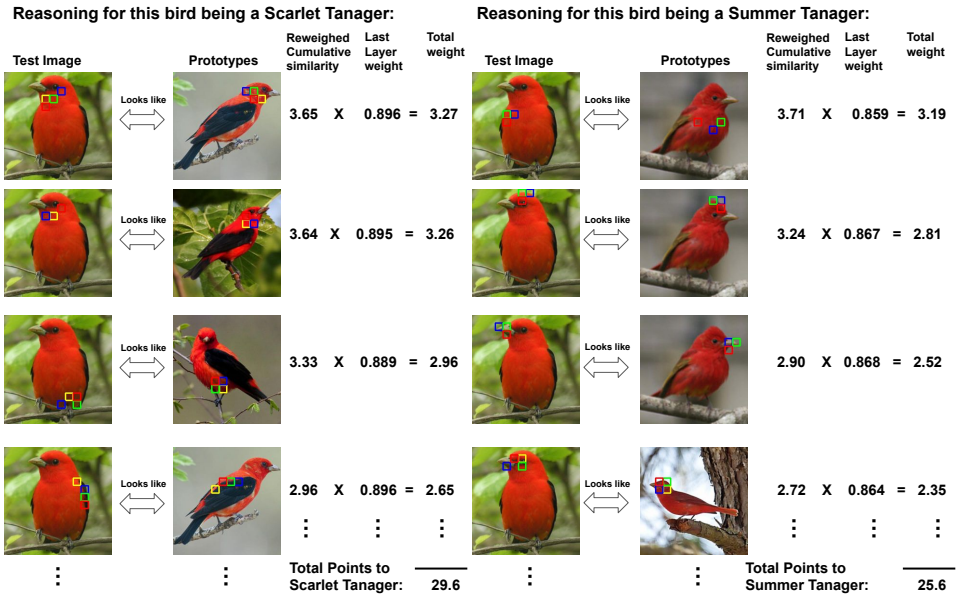

Figure 21: How ProtoViT with Deit-Tiny backbone classifies a test image of a Scarlet Tanager to the correct class (left), and the second most likely class Summer Tanager (right).

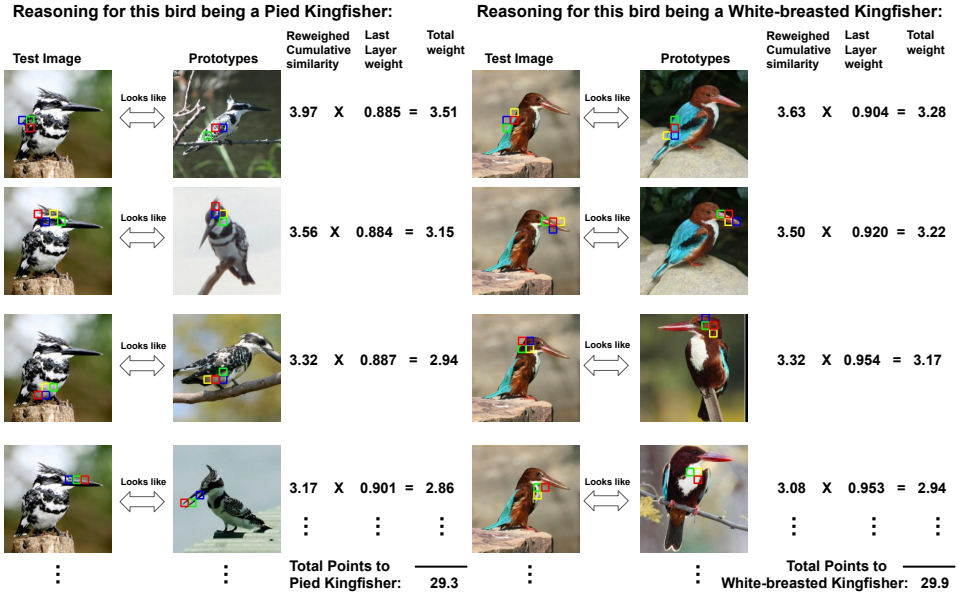

Figure 22: How ProtoViT with Deit-Tiny backbone classifies a test image of a Pied Kingfisher to the correct class (left), and classifies a test image of a White-breasted Kingfisher to the correct class (right).

https://doi.org/10.52202/079017-1312

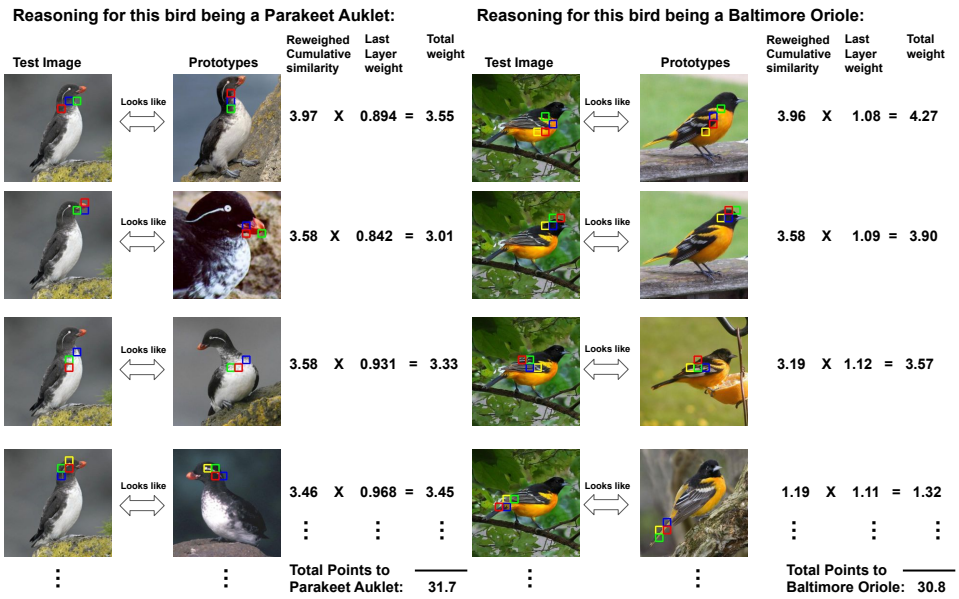

Figure 23: How ProtoViT with Deit-Tiny backbone classifies a test image of a Parakeet Auklet to the correct class (left), and classifies a test image of a Baltimore Oriole to the correct class (right).

I More examples of reasoning process for misclassification

In this section, we provide the reasoning process of how our model misclassified a test image of a summer tanager and a slaty backed gull in Fig. 24. We found that the misclassification of the given summer tanager example may be because of data mislabeling in the original dataset. A summer tanager does not have a black colored wing. That test image should indeed belong to the scarlet tanager class as the model predicted. Similar mislabeling cases also happen for Red Headed Woodpecker with image ID Red_Headed_Woordpecker_0018_183455 and Red_Headed_Woordpecker_0006_183383, as we found out when randomly selecting examples to present for the paper. On the other hands, misclassifications can be contributed by the similarity between different classes. As shown in the bottom of Fig. 24, the West Gull looks very similar to Slaty Gull. And the model's reasoning process indeed shows that the model believes that these two classes are the top two most likely classes for prediction.

These examples showcase the ability of our method to help us understand the reasoning process of the model, not just when it is right, but also when it is wrong.

J More examples of analysis

This section provides more examples for the local and global analysis. Fig. 25, Fig. 26, and Fig. 27 are the examples for analysis for ProtoViT with Deit-Small, CaiT-xxs24 and DeiT-tiney backbones respectively. The visualizations demonstrate that **the prototypes of Protovit exhibit consistent, strong semantic meanings across different ViT backbones.**

K Details on User Studies

Overview: This section provides details on the user study we conducted. Through our user study, we show that ProtoVit improves both the clarity of reasoning process and coherence of prototypical features relative to ProtoPNet[10] and Deformable ProtoPNet[13]. We randomly selected 10 bird images from test set, and show the comparison with the three most similar prototypes from its top-1 predicted class of the three models. We then presented the test to 10 participants, all of whom attend college in the U.S. and have some background in Machine Learning. We instructed participants to rate how well they understood the models' reasoning process through the presented examples,

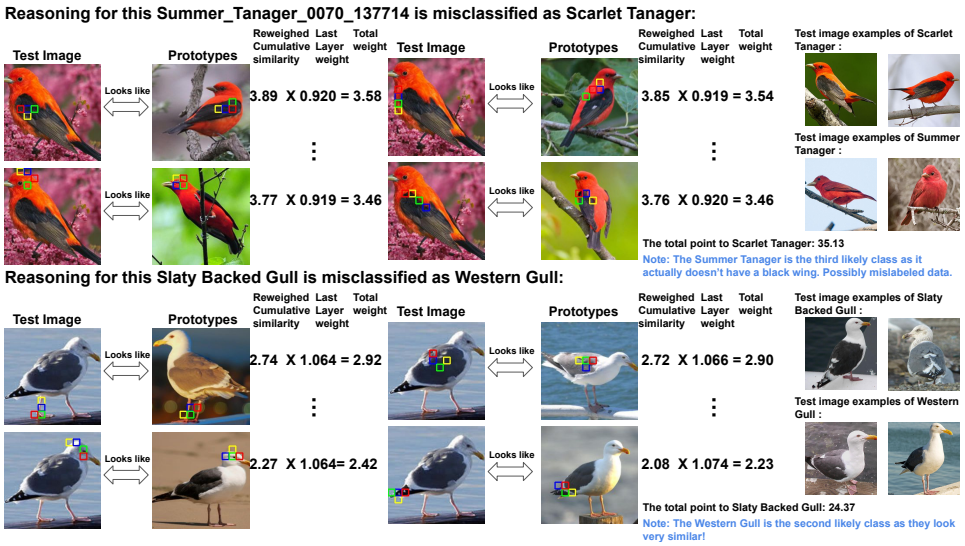

Figure 24: Misclassification examples. Reasoning process of how our model misclassified a test image of a summer tanager (top) and a slaty backed gull (bottom). The top misclassification is due to mislabeling.

and asked them to rate their confidence in their evaluation. Intuitively, the user should easily be able to distinguish which feature the model is comparing to, if the model has superb clarity. At the end of the experiment, we asked participants to rank the three models based on overall clarity of reasoning as well as coherence of prototypical features learned. Coherence of feature refers to when the visualization represents one, and only one feature. Some sample questions are shown in Appendix. Fig. 28. Although this task does not pose any risk to the participants, they were nevertheless informed of their rights, and were asked for consent before data collection. The participants took on average 10 to 20 minutes to complete their assigned tasks, and were compensated at $30 per hour.

Result: To quantify the user study result on model clarity, we assign a score from 4 to 1 as the participant rated their understanding of the model's reasoning from best to bad. Similarly, we assign a score from 4 to 1 for the confidence rating from completely confident to not confident. We then multiply the confidence score and the understanding score to have the total score on the clarity of the model reasoning. The maximum total score for a question is 16, which indicates that the participant has a very clear and confident understanding of the model's reasoning. The minimum total score for a question is 1, which indicates that the participant does not understand the model's reasoning at all. As shown in Table. 9, on average, the participants believe that they understand the reasoning process of ProtoViT the best with the highest confidence. The result of the one-sided t-test on understanding score and total score further illustrate that there is a statistically significant improvement in model clarity of ProtoViT to the vanilla ProtoPNet. Fig. 29 shows the result of participants ranking on the three models based on coherence of prototypical features and the clarity of the reasoning process. Again, ProtoViT is mostly ranked as the best in providing coherent prototypical features and clear reasoning process. It is worth noting that the majority of the participants commented that the prototypical features by ProtoPNet are usually too broad to understand what specific part the model is looking at. Moreover, the prototypical features by Deformable ProtoPNet are less accurate in pinpointing the specific parts. On the other hand, the prototypical features by Protovit can provide more accurate and specific visualizations. It is easy to tell if the prototypical features are representing the head of birds or the feet of the birds.

https://doi.org/10.52202/079017-1312

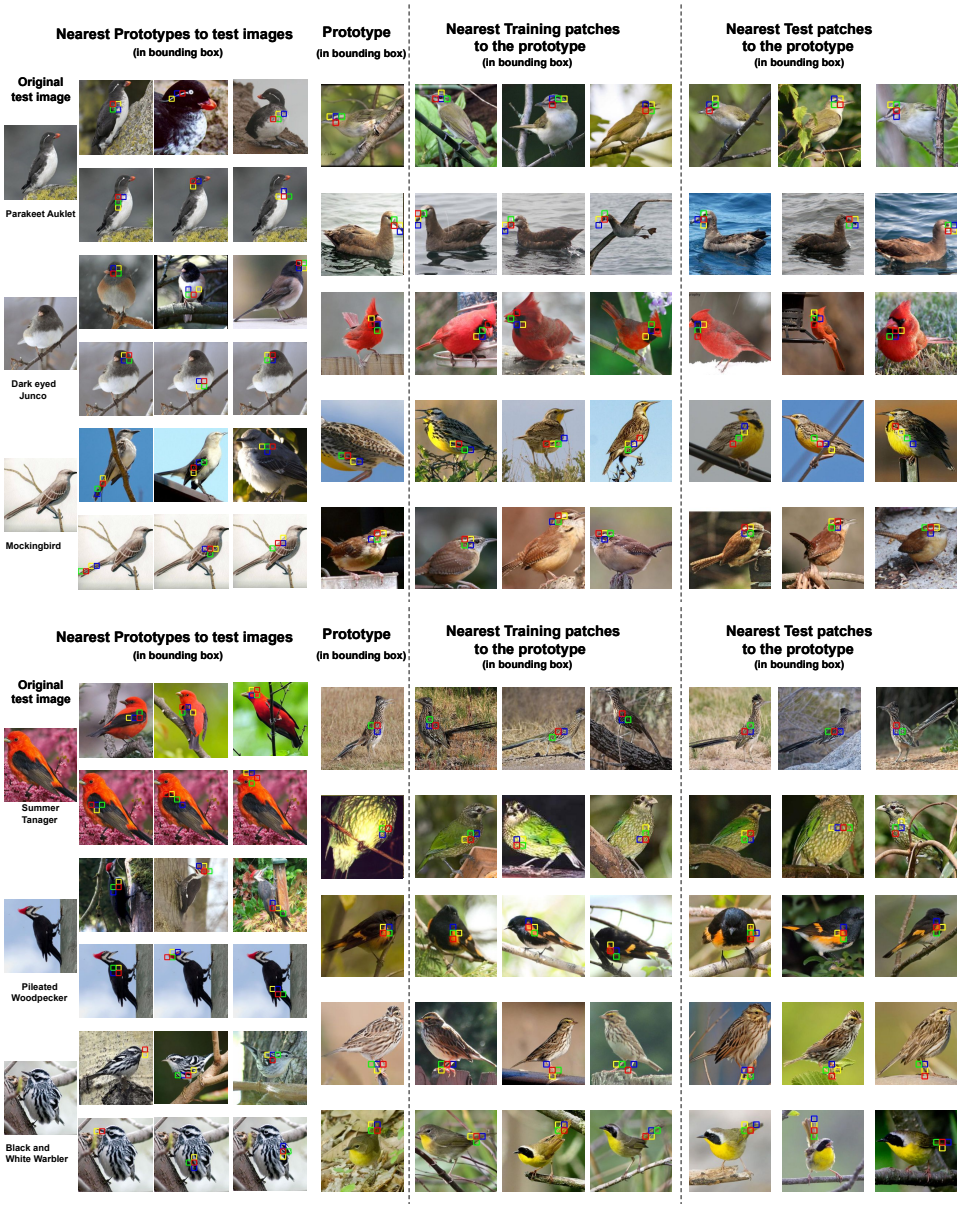

Figure 25: More examples of local analysis(left) and global analysis (right) of ProtoViT with DeiT-small backbone.

https://doi.org/10.52202/079017-1312

Figure 26: More examples of local analysis (left) and global analysis (right) of ProtoViT with CaiT-xxs24 backbone.

https://doi.org/10.52202/079017-1312

Figure 27: More examples of local analysis (left) and global analysis (right) of ProtoViT with DeiT-Tiny backbone.

https://doi.org/10.52202/079017-1312

Figure 28: Example of the survey questions. The examples are randomly selected, and the prototype with the highest similarity score for each example is selected.

https://doi.org/10.52202/079017-1312

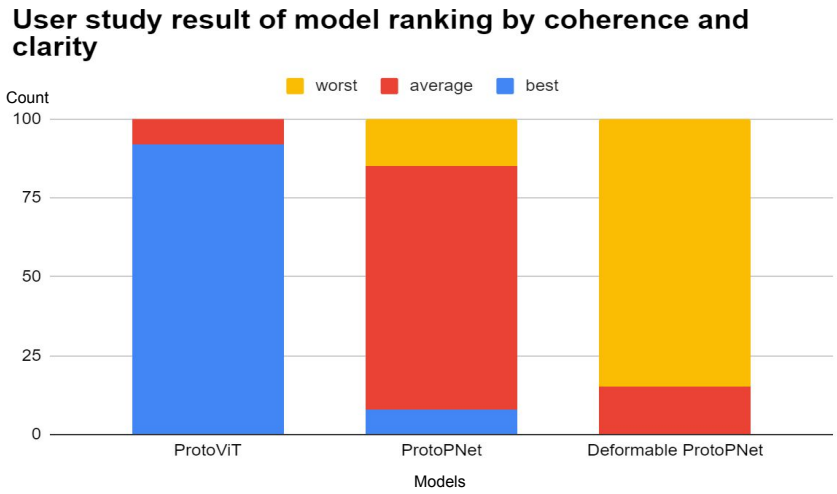

Figure 29: User Study result on model rankings based on its clarity and coherence

Table 9: User Study results on model clarity. We perform one-sided t-test on the total score and understanding score of ProtoViT and ProtoPNet.The result shows that ProtoViT has a statistically significant improvement in clarity of the reasoning process to ProtoPNet.

Model	Mean understanding score	Mean confidence score	Mean total score
Deformable ProtoPNet [13]	1.86	3.15	5.76
ProtoPNet [10]	2.68	3.17	8.67
ProtoViT	3.85	3.47	13.42

T-Test	Mean $\pm$ 1.96 std	T-stats	P-value
ProtoViT total score - ProtoPNet total score	4.75 ± 0.894	10.37	2.16×10^{-20}
ProtoViT Understanding score - ProtoPNet Understanding score	1.17 ± 0.209	10.89	5.75×10^{-22}

L Details on Car dataset

This section provides details on the implementation of ProtoViT on a second dataset. The Standford Car dataset [24] contains 8144/8041 images for training and testing from 196 different car models. We performed a similar offline data-augmentation as described in Sec. 4.1. After augmentation, the training set has roughly 1,600 images per class. We assigned the algorithm to choose 10 class-specific prototypes for each of the 196 classes. Each of the prototypes is composed of 4 sub-prototypes. We kept the training schedule and hyper-parameters same as on the bird dataset. The specific training schedule and hyper-parameters settings are documented in Appendix Tables 5 and 4 respectively. Fig. 30 and Fig. 31 show examples of the reasoning process for the DeiT-Small backbone. Fig. 32 and Fig. 33 show examples of the reasoning process for the CaiT-xxs-24 backbone. Fig. 34 and Fig. 35 show examples of the reasoning process for the DeiT-Tiny backbone. Examples of global analysis and local analysis for ProtoViT with different backbones are shown in Fig. 37, Fig. 36, and Fig. 38 respectively.

M Broader Impact

Interpretability is an essential ingredient to trustworthy AI systems. Our work successfully integrates one of the most popular and powerful model families into prototype-based networks, which are one of the leading techniques for interpretable neural networks in computer vision. Our technique can

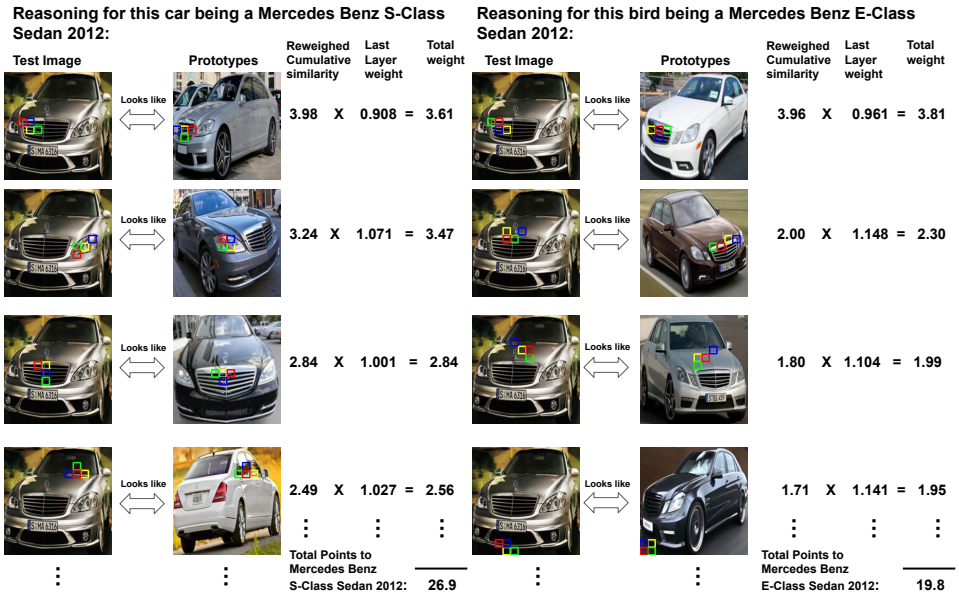

Figure 30: How ProtoViT with Deit-Small backbone classifies a test image of a Mercedes Benz S-class Sedan 2012 to the correct class (left), and the second most likely class Mercedes Benz E-class sedan 2012 (right).

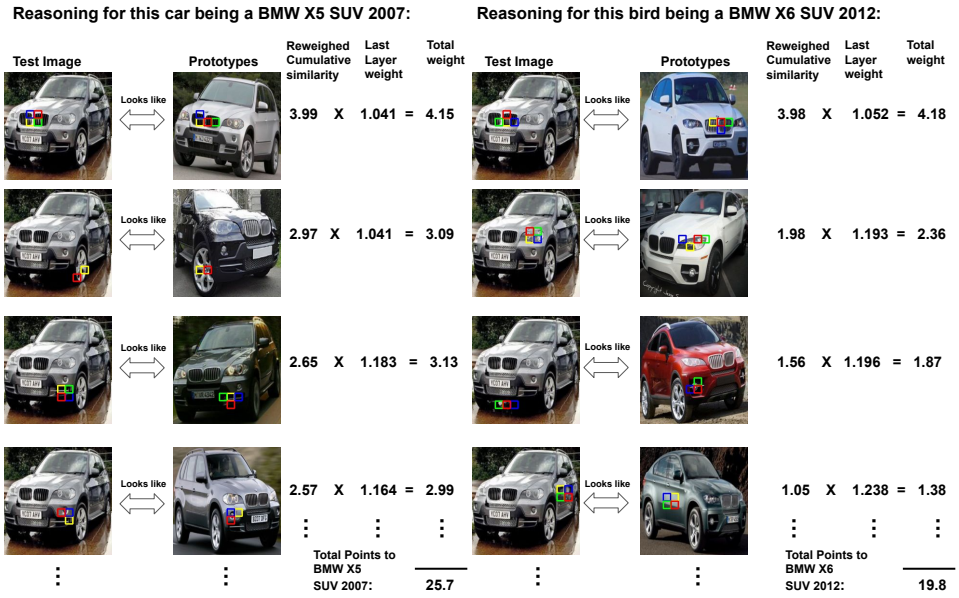

Figure 31: How ProtoViT with Deit-Small backbone classifies a test image of a BMW X5 SUV 2007 to the correct class (left), and the second most likely class BMW X6 SUV 2012 (right).

https://doi.org/10.52202/079017-1312

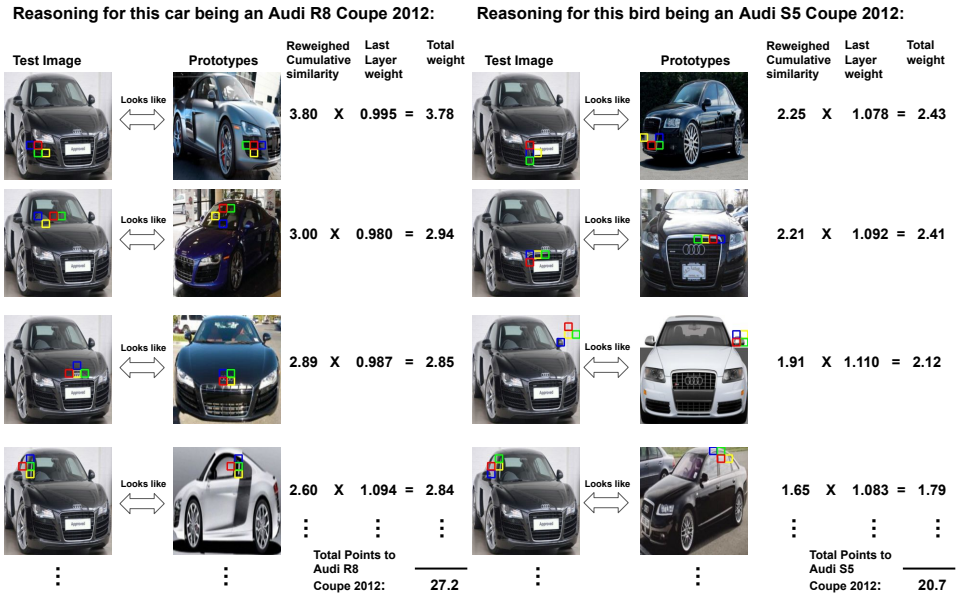

Figure 32: How ProtoViT with CaiT-xxs24 backbone classifies a test image of an Audi R8 Coupe 2012 to the correct class (left), and the second most likely class Audi S5 Coupe 2012 (right).

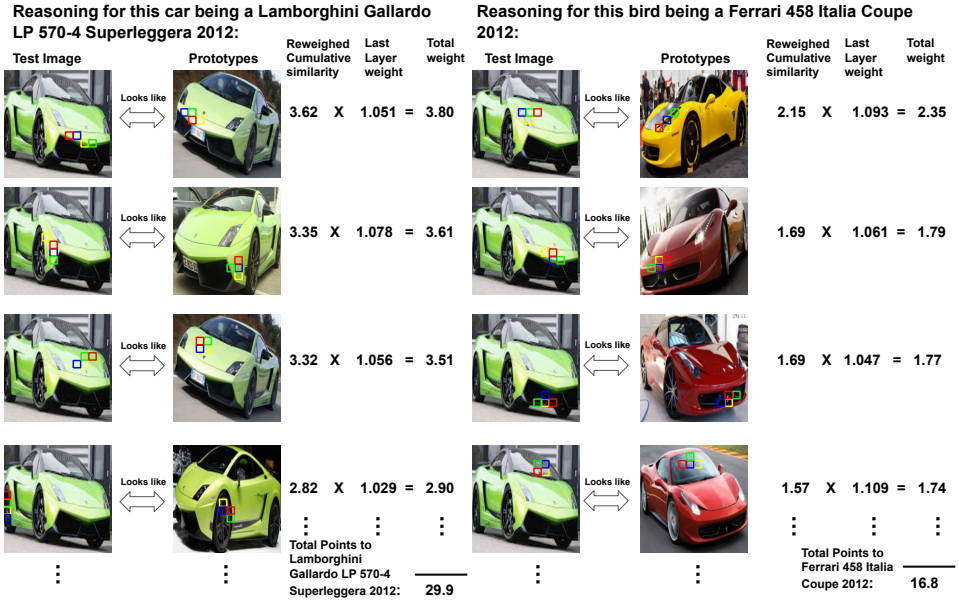

Figure 33: How ProtoViT with CaiT-xxs24 backbone classifies a test image of a Lamborghini Gallardo LP 570-4 Superleggera 2012 to the correct class (left), and the second most likely class Ferrari 458 Italia Coupe 2012 (right).

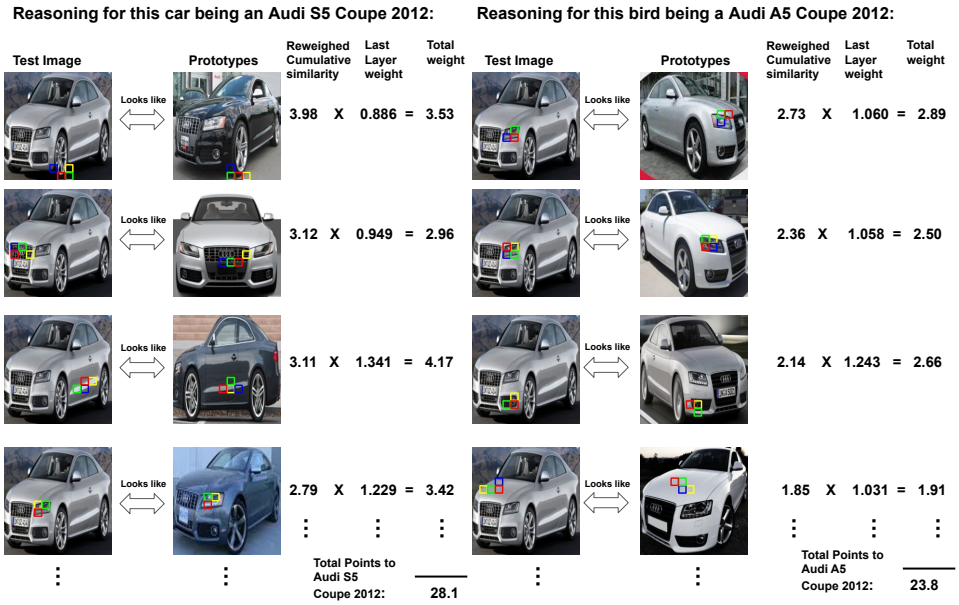

Figure 34: How ProtoViT with Deit-Tiny backbone classifies a test image of an Audi S5 Coupe 2012 to the correct class (left), and the second most likely class Audi A5 Coupe 2012 (right).

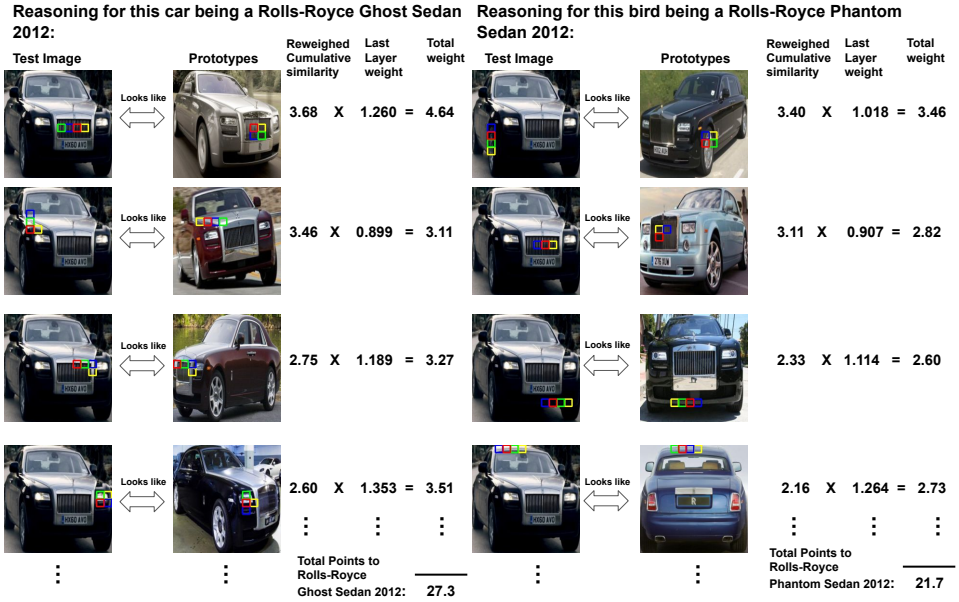

Figure 35: How ProtoViT with Deit-Tiny backbone classifies a test image of a Rolls-Royce Ghost Sedan 2012 to the correct class (left), and the second most likely class Rolls-Royce Phantom Sedan 2012 (right).

https://doi.org/10.52202/079017-1312

Figure 36: Examples of local analysis (left) and global analysis (right) of ProtoViT with CaiT-xxs24 backbone on the Stanford Cars dataset.

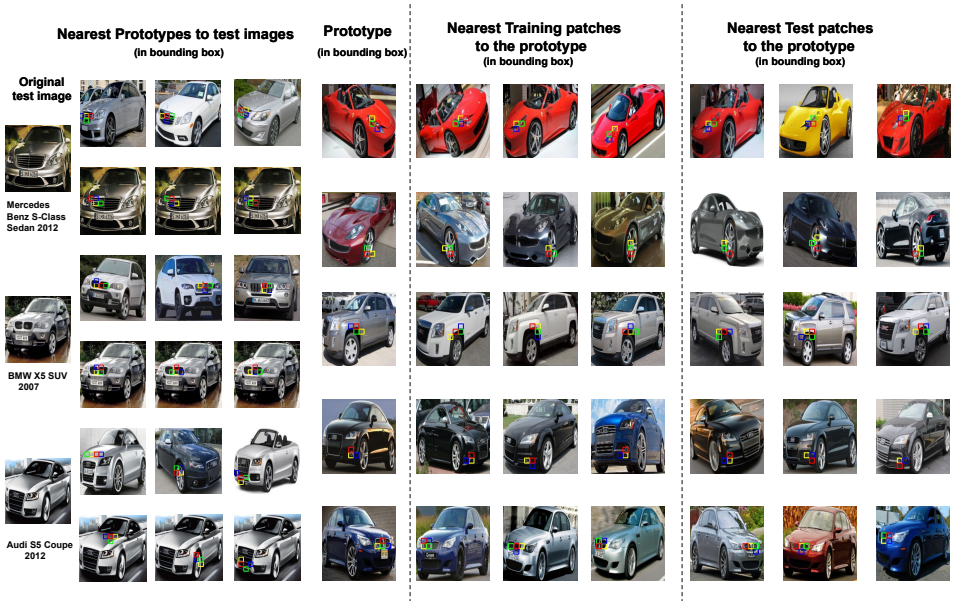

Figure 37: Examples of local analysis (left) and global analysis (right) of ProtoViT with Deit-Small backbone on the Stanford Cars dataset.

Figure 38: Examples of local analysis (left) and global analysis (right) of ProtoViT with Deit-tiny backbone on the Stanford Cars dataset.

be used for important computer vision applications to discover new knowledge and create better human-AI interfaces for difficult, high-impact applications.

N Computational Cost

By introducing greedy matching and coherence loss, we do not observe a significant increase in the computational cost. On the other hand, the computation of the adjacency mask may rely on the power of CPUs, as it involves more of matrix broadcasting and iterations. On a 13th Gen Intel R Core(TM) i9-13900KF CPU, it takes 0.2 seconds to compute each iteration with batch size 128, and $r = 1$ with 2000 prototypes. Overall, it takes roughly 16 hours to train the model with DeiT-Small backbone on the bird dataset, which is a similar amount of training time as ProtoPool [15] and TesNet [46] using a similar amount of backbone parameters.

O Training software and platform

We implemented our ProtoViT using Pytorch. The experiments were run on 1 NVIDIA Quadro RTX 6000 (24 GB), 1 NVIDIA Ge Force RTX 4090 (24 GB) or 1 NVIDIA RTX A6000 (48 GB).

https://doi.org/10.52202/079017-1312

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The experiments and results to support the claims in the introduction and abstract are all in the main paper and the appendix.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in the main paper Sec. 5 Limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There is no theoretical result and proof involved.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The training schedule and hyper-parameters are listed in Appendix Sec. B and Sec. A. The data information and augmentation are described in the Main and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/079017-1312

Answer: [Yes]

Justification: The data are public, and code is provided in the github link.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have listed all the settings in the appenix for paramtere settings and training schedules. The data splits are decribed under each case study section in the main paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All of our results are listed with error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute resources of experiments are documented in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impact is discussed in the appendix and shortly discussed in conclusion section in the main paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-1312

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer:[NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: They are properly cited and credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: the new model introduced in the paper is well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: The paper includes an user study and example questions are included.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The details of the user studies, including the consent process are described in Appendix Sec. K As the risks to the participants are minimal, we were exempted by our institution's IRB.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-1312