Interpretable Image Classification with Differentiable Prototypes Assignment

Dawid Rymarczyk[1,2], Łukasz Struski[1], Michał Górszczak[1],
Koryna Lewandowska[3], Jacek Tabor[1], and Bartosz Zieliński[1,2]

[1] Faculty of Mathematics and Computer Science, Jagiellonian University
[2] Ardigen SA
[3] Department of Cognitive Neuroscience and Neuroergonomics,
Institute of Applied Psychology, Jagiellonian University

Abstract. Existing prototypical-based models address the black-box nature of deep learning. However, they are sub-optimal as they often assume separate prototypes for each class, require multi-step optimization, make decisions based on prototype absence (so-called negative reasoning process), and derive vague prototypes. To address those shortcomings, we introduce ProtoPool, an interpretable prototype-based model with positive reasoning and three main novelties. Firstly, we reuse prototypes in classes, which significantly decreases their number. Secondly, we allow automatic, fully differentiable assignment of prototypes to classes, which substantially simplifies the training process. Finally, we propose a new focal similarity function that contrasts the prototype from the background and consequently concentrates on more salient visual features. We show that ProtoPool obtains state-of-the-art accuracy on the CUB-200-2011 and the Stanford Cars datasets, substantially reducing the number of prototypes. We provide a theoretical analysis of the method and a user study to show that our prototypes capture more salient features than those obtained with competitive methods. We made the code available at `https://github.com/gmum/ProtoPool`.

Keywords: deep learning; interpretability; case-based reasoning

1 Introduction

The broad application of deep learning in fields like medical diagnosis [3] and autonomous driving [53], together with current law requirements (such as GDPR in EU [21]), enforces models to explain the rationale behind their decisions. That is why explainers [6,23,29,39,44] and self-explainable [4,7,58] models are developed to justify neural network predictions. Some of them are inspired by mechanisms used by humans to explain their decisions, like matching image parts with memorized prototypical features that an object can poses [8,27,34,43].

Recently, a self-explainable model called Prototypical Part Network (ProtoPNet) [8] was introduced, employing feature matching learning theory [40,41]. It focuses on crucial image parts and compares them with reference patterns

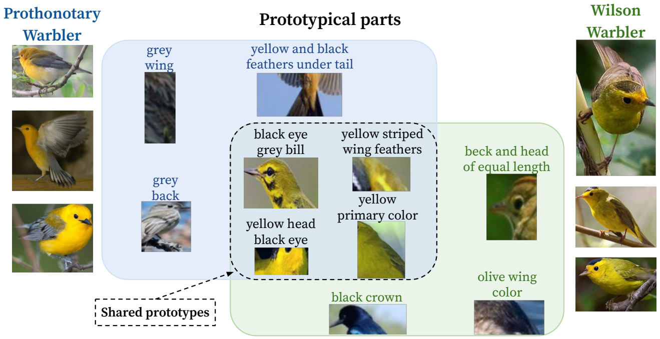


Fig. 1: Automatically discovered prototypes[4] for two classes, *Prothonotary Warbler* and *Wilson Warbler* (each class represented by three images on left and right side). Three prototypical parts on the blue and green background are specific for a *Prothonotary Warbler* and *Wilson Warbler*, respectively (they correspond to heads and wings feathers). At the same time, four prototypes shared between those classes (related to yellow feathers) are presented in the intersection. Prototypes sharing reduces their amount, leads to a more interpretable model, and discovers classes similarities.

(prototypical parts) assigned to classes. The comparison is based on a similarity metric between the image activation map and representations of prototypical parts (later called prototypes). The maximum value of similarity is pooled to the classification layer. As a result, ProtoPNet explains each prediction with a list of reference patterns and their similarity to the input image. Moreover, a global explanation can be obtained for each class by analyzing prototypical parts assigned to particular classes.

However, ProtoPNet assumes that each class has its own separate set of prototypes, which is problematic because many visual features occur in many classes. For instance, both *Prothonotary Warbler* and *Wilson Warbler* have yellow as a primary color (see Figure 1). Such limitation of ProtoPNet hinders the scalability because the number of prototypes grows linearly growing number of classes. Moreover, a large number of prototypes makes ProtoPNet hard to interpret by the users and results in many background prototypes [43].

To address these limitations, ProtoPShare [43] and ProtoTree [34] were introduced. They share the prototypes between classes but suffer from other drawbacks. ProtoPShare requires previously trained ProtoPNet to perform the merge-pruning step, which extends the training time. At the same time, ProtoTree builds a decision tree and exploits the negative reasoning process that may result in explanations based only on prototype absence. For example, a model can predict a *sparrow* because an image does not contain red feathers, a long beak,

[4] Names of prototypical parts were generated based on the annotations from CUB-200-2011 dataset (see details in Supplementary Materials).

and wide wings. While this characteristic is true in the case of a *sparrow*, it also matches many other species.

To deal with the above shortcomings, we introduce ProtoPool, a self-explainable prototype model for fine-grained images classification. ProtoPool introduces significantly novel mechanisms that substantially reduce the number of prototypes and obtain higher interpretability and easier training. Instead of using hard assignment of prototypes to classes, we implement the soft assignment represented by a distribution over the set of prototypes. This distribution is randomly initialized and binarized during training using the Gumbel-Softmax trick. Such a mechanism simplifies the training process by removing the pruning step required in ProtoPNet, ProtoPShare, and ProtoTree. The second novelty is a focal similarity function that focuses the model on the salient features. For this purpose, instead of maximizing the global activation, we widen the gap between the maximal and average similarity between the image activation map and prototypes (see Figure 4). As a result, we reduce the number of prototypes and use the positive reasoning process on salient features, as presented in Figure 2 and Figure 10.

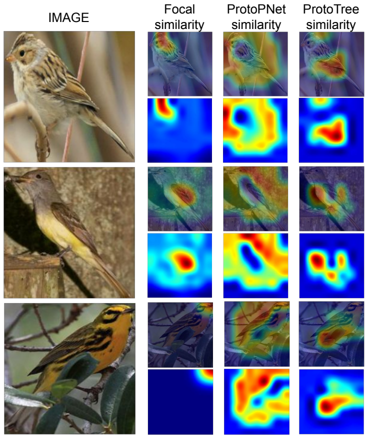


Fig. 2: Focal similarity focuses the prototype on a salient visual feature. While the other similarity metrics are more distributed through the image, making the interpretation harder to comprehend. It is shown with three input images, the prototype activation map, and its overlay.

We confirm the effectiveness of ProtoPool with theoretical analysis and exhaustive experiments, showing that it achieves the highest accuracy among models with a reduced number of prototypes. What is more, we discuss interpretability, perform a user study, and discuss the cognitive aspects of the ProtoPool over existing methods.

The main achievements of the paper can be summarized as follows:

- We construct ProtoPool, a case-based self-explainable method that shares prototypes between data classes without any predefined concept dictionary.
- We introduce fully differentiable assignments of prototypes to classes, allowing the end-to-end training.
- We define a novel similarity function, called focal similarity, that focuses the model on the salient features.
- We increase interpretability by reducing prototypes number and providing explanations in a positive reasoning process.

2 Related works

Attempts to explain deep learning models can be divided into the post hoc and self-explainable [42] methods. The former approaches assume that the reasoning process is hidden in a black box model and a new explainer model has to be created to reveal it. Post hoc methods include a saliency map [31,38,44,45,46] generating a heatmap of crucial image parts, or Concept Activation Vectors (CAV) explaining the internal network state as user-friendly concepts [9,14,23,25,55]. Other methods provide counterfactual examples [1,15,33,36,52] or analyze the networks' reaction to the image perturbation [6,11,12,39]. Post hoc methods are easy to implement because they do not interfere with the architecture, but they can produce biased and unreliable explanations [2]. That is why more focus is recently put on designing self-explainable models [4,7] that make the decision process directly visible. Many interpretable solutions are based on the attention [28,48,54,57,58,59] or exploit the activation space [16,37], e.g. with adversarial autoencoder. However, most recent approaches built on an interpretable method introduced in [8] (ProtoPNet) with a hidden layer of prototypes representing the activation patterns.

ProtoPNet inspired the design of many self-explainable models, such as TesNet [51] that constructs the latent space on a Grassman manifold without prototypes reduction. Other models like ProtoPShare [43] and ProtoTree [34] reduce the number of prototypes used in the classification. The former introduces data-dependent merge-pruning that discovers prototypes of similar semantics and joins them. The latter uses a soft neural decision tree that may depend on the negative reasoning process. Alternative approaches organize the prototypes hierarchically [17] to classify input at every level of a predefined taxonomy or transform prototypes from the latent space to data space [27]. Moreover, prototype-based solutions are widely adopted in various fields such as medical imaging [3,5,24,47], time-series analysis [13], graphs classification [56], and sequence learning [32].

3 ProtoPool

In this section, we describe the overall architecture of ProtoPool presented in Figure 3 and the main novelties of ProtoPool compared to the existing models, including the mechanism of assigning prototypes to slots and the focal similarity. Moreover, we provide a theoretical analysis of the approach.

Overall architecture The architecture of ProtoPool, shown in Figure 3, is generally inspired by ProtoNet [8]. It consists of convolutional layers f, a prototype pool layer g, and a fully connected layer h. Layer g contains a pool of M trainable prototypes $P = \{p_i \in \mathbb{R}^D\}_{i=1}^M$ and K slots for each class. Each slot is implemented as a distribution $q_k \in \mathbb{R}^M$ of prototypes available in the pool, where successive values of q_k correspond to the probability of assigning successive prototypes to slot k ($\|q_k\| = 1$). Layer h is linear and initialized to enforce

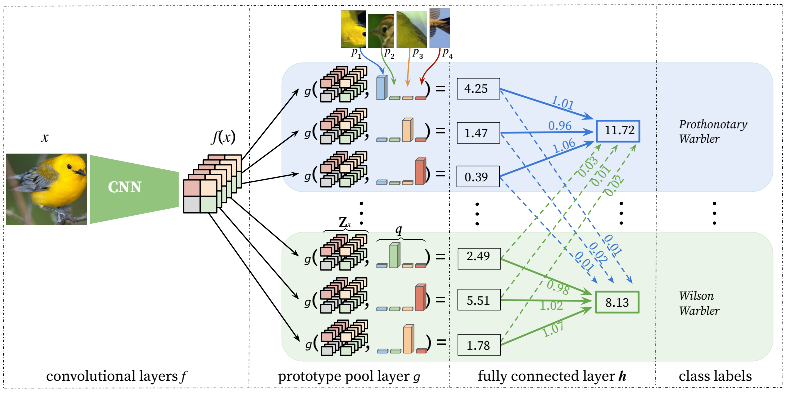


Fig. 3: The architecture of our ProtoPool with a prototype pool layer g. Layer g contains a pool of prototypes $p_1 - p_4$ and three slots per class. Each slot is implemented as a distribution $q \in \mathbb{R}^4$ of prototypes from the pool, where successive values of q correspond to the probability of assigning successive prototypes to the slot. In this example, p_1 and p_2 are assigned to the first slot of *Prothonotary Warbler* and *Wilson Warbler*, respectively. At the same time, the shared prototypes p_3 and p_4 are assigned to the second and third slots of both classes.

the positive reasoning process, i.e. weights between each class c and its slots are initialized to 1 while remaining weights of h are set to 0.

Given an input image $x \in X$, the convolutional layers first extract image representation $f(x)$ of shape $H \times W \times D$, where H and W are the height and width of representation obtained at the last convolutional layer for image x, and D is the number of channels in this layer. Intuitively, $f(x)$ can be considered as a set of $H \cdot W$ vectors of dimension D, each corresponding to a specific location of the image (as presented in Figure 3). For the clarity of description, we will denote this set as $Z_x = \{z_i \in f(x) : z_i \in \mathbb{R}^D, i = 1, ..., H \cdot W\}$. Then, the prototype pool layer is used on each k-th slot to compute the aggregated similarity $g_k = \sum_{i=1}^{M} q_k^i g_{p_i}$ between Z_x and all prototypes considering the distribution q_k of this slot, where g_p is defined below. Finally, the similarity scores (K values per class) are multiplied by the weight matrix w_h in the fully connected layer h. This results in the output logits, further normalized using softmax to obtain a final prediction.

Focal similarity In ProtoPNet [8] and other models using prototypical parts, the similarity of point z to prototype p is defined as[5] $g_p(z) = \log(1 + \frac{1}{\|z-p\|^2})$, and the final activation of the prototype p with respect to image x is given by $g_p = \max_{z \in Z_x} g_p(z)$. One can observe that such an approach has two possible disadvantages. First, high activation can be obtained when all the elements in Z_x are similar to a prototype. It is undesirable because the prototypes can then

[5] The following regularization is used to avoid numerical instability in the experiments: $g_p(z) = \log(\frac{\|z-p\|^2+1}{\|z-p\|^2+\varepsilon})$, with a small $\varepsilon > 0$.

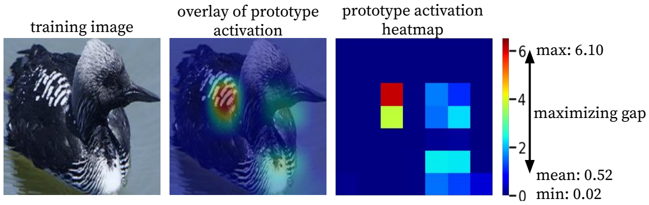


Fig. 4: Our focal similarity limits high prototype activation to a narrow area (corresponding to white and black striped wings). It is obtained by widening the gap between the maximal and average activation (equal 6.10 and 0.52, respectively). As a result, our prototypes correspond to more salient features (according to our user studies described in Section 5).

concentrate on the background. The other negative aspect concerns the training process, as the gradient is passed only through the most active part of the image.

To prevent those behaviors, in ProtoPool, we introduce a novel focal similarity function that widens the gap between maximal and average activation

$$g_p = \max_{z \in Z_x} g_p(z) - \operatorname*{mean}_{z \in Z_x} g_p(z), \tag{1}$$

as presented in Figure 4. The maximal activation of focal similarity is obtained if a prototype is similar to only a narrow area of the image x (see Figure 2). Consequently, the constructed prototypes correspond to more salient features (according to our user studies described in Section 5), and the gradient passes through all elements of Z_x.

Assigning one prototype per slot Previous prototypical methods use the hard predefined assignment of the prototypes to classes [8,43,51] or nodes of a tree [34]. Therefore, no gradient propagation is needed to model the prototypes assignment. In contrast, our ProtoPool employs a soft assignment based on prototypes distributions to use prototypes from the pool optimally. To generate prototype distribution q, one could apply softmax on the vector of size $\mathbb{R}^M$. However, this could result in assigning many prototypes to one slot and consequently could decrease the interpretability. Therefore, to obtain distributions with exactly one probability close to 1, we require a differentiable arg max function. A perfect match, in this case is the Gumbel-Softmax estimator [20], where for $q = (q^1, \ldots, q^M) \in \mathbb{R}^M$ and $\tau \in (0, \infty)$

$$\text{Gumbel-softmax}(q, \tau) = (y^1, \ldots, y^M) \in \mathbb{R}^M,$$

where $y^i = \frac{\exp\left((q^i + \eta_i)/\tau\right)}{\sum_{m=1}^{M} \exp((q^m + \eta_m)/\tau)}$ and η_m for $m \in 1, .., M$ are samples drawn from standard Gumbel distribution. The Gumbel-Softmax distribution interpolates between continuous categorical densities and discrete one-hot-encoded categorical distributions, approaching the latter for low temperatures $\tau \in [0.1, 0.5]$ (see Figure 5).

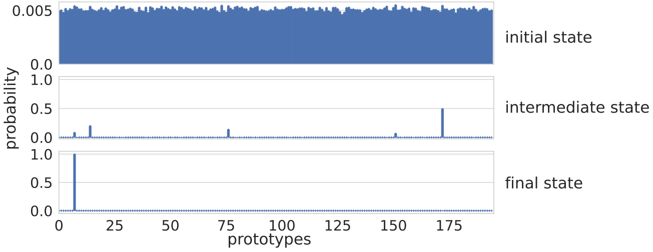


Fig. 5: A sample distribution (slot) at the initial, middle, and final step of training. In the beginning, all prototypes are assigned with a probability of 0.005. Then, the distribution binarizes, and finally, one prototype is assigned to this slot with a probability close to 1.

Slots orthogonality Without any additional constraints, the same prototype could be assigned to many slots of one class, wasting the capacity of the prototype pool layer and consequently returning poor results. Therefore, we extend the loss function with

$$\mathcal{L}_{orth} = \sum_{i<j}^{K} \frac{\langle q_i, q_j \rangle}{\|q_i\|_2 \cdot \|q_j\|_2}, \tag{2}$$

where $q_1, .., q_K$ are the distributions of a particular class. As a result, successive slots of a class are assigned to different prototypes.

Prototypes projection Prototypes projection is a step in the training process that allows prototypes visualization. It replaces each abstract prototype learned by the model with the representation of the nearest training patch. For prototype p, it can be expressed by the following formula

$$p \leftarrow \arg\min_{z \in Z_C} \|z - p\|_2, \tag{3}$$

where $Z_C = \{z : z \in Z_x \text{ for all } (x, y) : y \in C\}$. In contrast to [8], set C is not a single class but the set of classes assigned to prototype p.

Theoretical analysis Here, we theoretically analyze why ProtoPool assigns one prototype per slot and why each prototype does not repeat in a class. For this purpose, we provide two observations.

Observation 1 *Let $q \in [0,1]^M$, $\sum q_i = 1$ be a distribution (slot) of a particular class. Then, the limit of* Gumbel-softmax(q, τ)*, as τ approaches zero, is the canonical vector $e_i \in \mathbb{R}^M$, i.e. for q there exists $i = 1, .., M$ such that* $\lim_{\tau \to 0} \text{Gumbel-softmax}(q, \tau) = e_i$.

The temperature parameter $\tau > 0$ controls how closely the new samples approximate discrete one-hot vectors (the canonical vector). From paper [20] we know that as $\tau \to 0$, the softmax computation smoothly approaches the arg max, and the sample vectors approach one-hot q distribution (see Figure 5).

Table 1: Comparison of ProtoPool with other prototypical methods trained on the CUB-200-2011 and Stanford Cars datasets, which considers a various number of prototypes and types of convolutional layers f. In the case of the CUB-200-2011 dataset, ProtoPool achieves the highest accuracy than other models, even those containing ten times more prototypes. Moreover, the ensemble of three ProtoPools surpasses the ensemble of five TesNets with 17 times more prototypes. On the other hand, in the case of Stanford Cars, ProtoPool achieves competitive results with significantly fewer prototypes. Please note that the results are first sorted by backbone network and then by the number of prototypes, R stands for ResNet, iN means pretrained on iNaturalist, and Ex is an ensemble of three or five models.

CUB-200-2011			
Model	Arch.	Proto. #	Acc [%]
ProtoPool (ours)	R34	202	80.3±0.2
ProtoPShare [43]		400	74.7
ProtoPNet [8]		1655	79.5
TesNet [51]		2000	82.7±0.2
ProtoPool (ours)	R152	202	81.5±0.1
ProtoPShare [43]		1000	73.6
ProtoPNet [8]		1734	78.6
TesNet [51]		2000	82.8±0.2
ProtoPool (ours)	iNR50	202	85.5±0.1
ProtoTree [34]		202	82.2±0.7
ProtoPool (ours)	Ex3	202×3	87.5
ProtoTree [34]		202×3	86.6
ProtoPool (ours)	Ex5	202×5	**87.6**
ProtoTree [34]		202×5	87.2
ProtoPNet [8]		2000×5	84.8
TesNet [51]		2000×5	86.2

Stanford Cars			
Model	Arch.	Proto. #	Acc [%]
ProtoPool (ours)	R34	195	89.3±0.1
ProtoPShare [43]		480	86.4
ProtoPNet [8]		1960	86.1±0.2
TesNet [51]		1960	92.6±0.3
ProtoPool (ours)	R50	195	88.9±0.1
ProtoTree [34]		195	86.6±0.2
ProtoPool (ours)	Ex3	195×3	91.1
ProtoTree [34]		195×3	90.5
ProtoPool (ours)	Ex5	195×5	91.6
ProtoTree [34]		195×5	91.5
ProtoPNet [8]		1960×5	91.4
TesNet [51]		1960×5	**93.1**

Observation 2 *Let $K \in \mathbb{N}$ and $q_1, .., q_K$ be the distributions (slots) of a particular class. If $\mathcal{L}_{orth}$ defined in Eq. (2) is zero, then each prototype from a pool is assigned to only one slot of the class.*

It follows the fact that $\mathcal{L}_{orth} = 0$ only if $\langle q_i, q_j \rangle = 0$ for all $i < j \leq K$, i.e. only if q_i, q_j have non-zero values for different prototypes.

4 Experiments

We train our model on CUB-200-2011 [50] and Stanford Cars [26] datasets to classify 200 bird species and 196 car models, respectively. As the convolutional layers f of the model, we take ResNet-34, ResNet-50, ResNet-121 [18], DenseNet-121, and DenseNet-161 [19] without the last layer, pretrained on ImageNet [10].

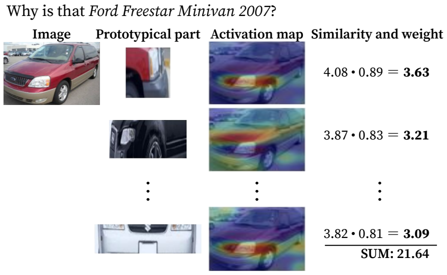


Fig. 6: Sample explanation of *Ford Freestar Minivan 2007* predictions. Except for an image, we present a few prototypical parts of this class, their activation maps, similarity function values, and the last layer weights. Moreover, we provide the sum of the similarities multiplied by the weights. ProtoPool returns the class with the largest sum as a prediction.

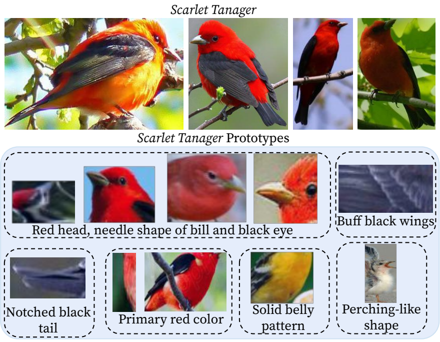


Fig. 7: Samples of *Scarlet Tanager* and prototypical parts assigned to this class by our ProtoPool model. Prototypes correspond, among others, to primary red color of feathers, black eye, perching-like shape, black notched tail, and black buff wings[6].

The one exception is ResNet-50 used with CUB-200-2011 dataset, which we pretrain on iNaturalist2017 [49] for fair comparison with ProtoTree model [34]. In the testing scenario, we make the prototype assignment hard, i.e. we set all values of a distribution q higher than 0.5 to 1, and the remaining values to 0 otherwise. We set the number of prototypes assigned to each class to be at most 10 and use the pool of 202 and 195 prototypical parts for CUB-200-2011 and Stanford Cars, respectively. Details on experimental setup and results for other backbone networks are provided in the Supplementary Materials.

Comparison with other prototypical models In Table 1 we compare the efficiency of our ProtoPool with other models based on prototypical parts. We report the mean accuracy and standard error of the mean for 5 repetitions. Additionally, we present the number of prototypes used by the models, and we use this parameter to sort the results. We compare ProtoPool with ProtoPNet [8], ProtoPShare [43], ProtoTree [34], and TesNet [51].

One can observe that ProtoPool achieves the highest accuracy for the CUB-200-2011 dataset, surpassing even the models with a much larger number of prototypical parts (TesNet and ProtoPNet). For Stanford Cars, our model still performs better than other models with a similarly low number of prototypes, like ProtoTree and ProtoPShare, and slightly worse than TesNet, which uses ten times more prototypes. The higher accuracy of the latter might be caused by prototype orthogonality enforced in training. Overall, our method achieves competitive results with significantly fewer prototypes. However, ensemble ProtoPool or TesNet should be used if higher accuracy is preferred at the expense of interpretability.

Table 2: Characteristics of prototypical methods for fine-grained image classification that considers the number of prototypes, reasoning type, and prototype sharing between classes. ProtoPool uses 10% of ProtoPNet's prototypes but only with positive reasoning. It shares the prototypes between classes but, in contrast to ProtoPShare, is trained in an end-to-end, fully differentiable manner. Please notice that 100% corresponds to 2000 and 1960 of prototypes for CUB-200-2011 and Stanford Cars datasets, respectively.

Model	ProtoPool	ProtoTree	ProtoPShare	ProtoPNet	TesNet
Portion of prototypes	∼10%	∼10%	[20%;50%]	100%	100%
Reasoning type	+	+/−	+	+	+
Prototype sharing	direct	indirect	direct	none	none

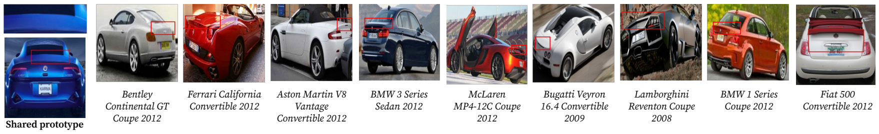

Shared prototype
Bentley Continental GT Coupe 2012
Ferrari California Convertible 2012
Aston Martin V8 Vantage Convertible 2012
BMW 3 Series Sedan 2012
McLaren MP4-12C Coupe 2012
Bugatti Veyron 16.4 Convertible 2009
Lamborghini Reventon Coupe 2008
BMW 1 Series Coupe 2012
Fiat 500 Convertible 2012

Fig. 8: Sample prototype of a *convex tailgate* (left top corner) shared by nine classes. Most of the classes correspond to luxury cars, but some exceptions exist, such as *Fiat 500*.

5 Interpretability

In this section, we analyze the interpretability of the ProtoPool model. Firstly, we show that our model can be used for local and global explanations. Then, we discuss the differences between ProtoPool and other prototypical approaches, and investigate its stability. Then, we perform a user study on the similarity functions used by the ProtoPNet, ProtoTree, and ProtoPool to assess the saliency of the obtained prototypes. Lastly, we consider ProtoPool from the cognitive psychology perspective.

Local and global interpretations Except for local explanations that are similar to those provided by the existing methods (see Figure 6), ProtoPool can provide a global characteristic of a class. It is presented in Figure 7, where we show the prototypical parts of *Scarlet Tanager* that correspond to the visual features of this species, such as red feathers, a puffy belly, and a short beak. Moreover, similarly to ProtoPShare, ProtoPool shares the prototypical parts between data classes. Therefore, it can describe the relations between classes relying only on the positive reasoning process, as presented in Figure 1 (in contrast, ProtoTree also uses negative reasoning). In Figure 8, we further provide visualization of the prototypical part shared by nine classes. More examples are provided in Supplementary Materials.

Differences between prototypical methods In Table 2, we compare the characteristics of various prototypical-based methods. Firstly, ProtoPool and

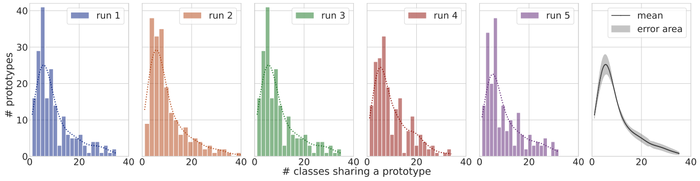


Fig. 9: Distribution presenting how many prototypes are shared by the specific number of classes (an estimation plot is represented with a dashed line). Each color corresponds to a single ProtoPool training on Stanford Cars dataset with ResNet50 as a backbone network. The right plot corresponds to the mean and standard deviation for five training runs. One can observe that the distribution behaves stable between runs.

ProtoTree utilize fewer prototypical parts than ProtoPNet and TesNet (around 10%). ProtoPShare also uses fewer prototypes (up to 20%), but it requires a trained ProtoPNet model before performing merge-pruning. Regarding class similarity, it is directly obtained from ProtoPool slots, in contrast to ProtoTree, which requires traversing through the decision tree. Moreover, ProtoPNet and TesNet have no mechanism to detect inter-class similarities. Finally, ProtoTree depends, among others, on *negative* reasoning process, while in the case of ProtoPool, it relies only on the positive reasoning process, which is a desirable feature according to [8].

Stability of shared prototypes The natural question that appears when analyzing the assignment of the prototypes is: *Does the similarity between two classes hold for many runs of ProtoPool training?* To analyze this behavior, in Figure 9 we show five distributions for five training runs. They present how many prototypes are shared by the specific number of classes. One can observe that difference between runs is negligible. In all runs, most prototypes are shared by five classes, but there exist prototypes shared by more than thirty classes. Moreover, on average, a prototype is shared by 2.73 ± 0.51 classes. A sample inter-class similarity graph is presented in the Supplementary Materials.

User study on focal similarity To validate if using focal similarity results in more salient prototypical parts, we performed a user study where we asked the participants to answer the question: *"How salient is the feature pointed out by the AI system?"*. The task was to assign a score from 1 to 5 where 1 meant *"Least salient"* and 5 meant *"Most salient"*. Images were generated using prototypes obtained for ProtoPool with ProtoPNets similarity or with focal similarity and from a trained ProtoTree[7]. To perform the user study, we used Amazon Mechan-

[6] Names of prototypical parts were generated based on the annotations from CUB-200-2011 dataset (see details in Supplementary Materials).

[7] ProtoTree was trained using code from `https://github.com/M-Nauta/ProtoTree` and obtained accuracy similar to [34]. For ProtoPNet similarity, we used code from `https://github.com/cfchen-duke/ProtoPNet`.

ical Turk (AMT) system[8]. To assure the reliability of the answers, we required the users to be masters according to AMT. 40 workers participated in our study and answered 60 questions (30 per dataset) presented in a random order, which resulted in 2400 answers. Each question contained an original training image and the same image with overlayed activation map, as presented in Figure 2.

Results presented in Figure 10 show that ProtoPool obtains mostly scores from 3 to 5, while other methods often obtain lower scores. We obtained a mean value of scores equal to 3.66, 2.87, and 2.85 for ProtoPool, ProtoTree, and ProtoPool without focal similarity, respectively. Hence, we conclude that ProtoPool with focal similarity generated more salient prototypes than the reference models, including ProtoTree. See Supplementary Materials for more information about a user study, detailed results, and a sample questionnaire.

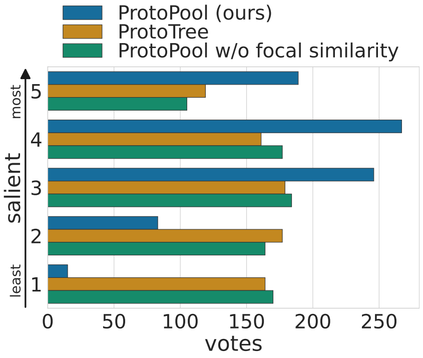


Fig. 10: Distribution of scores from user study on prototypes obtained for ProtoPool without and with focal similarity and for ProtoTree. One can observe that ProtoPool with focal similarity generates more salient prototypes than the other models.

ProtoPool in the context of cognitive psychology ProtoPool can be described in terms of parallel or simultaneous information processing, while ProtoTree may be characterized by serial or successive processing, which takes more time [22,30,35]. More specifically, human cognition is marked with the speed-accuracy trade-off. Depending on the perceptual situation and the goal of a task, the human mind can apply a categorization process (simultaneous or successive) that is the most appropriate in a given context, i.e. the fastest or the most accurate. Both models have their advantages. However, ProtoTree has a specific shortcoming because it allows for a categorization process to rely on an absence of features. In other words, an object characterized by none of the enlisted features is labeled as a member of a specific category. This type of reasoning is useful when the amount of information to be processed (i.e. number of features and categories) is fixed and relatively small. However, the time of object categorization profoundly elongates if the number of categories (and therefore the number of features to be crossed out) is high. Also, the chance of miscategorizing completely new information is increased.

6 Ablation study

In this section, we analyze how the novel architectural choices, the prototype projection, and the number of prototypes influence the model performance.

[8] https://www.mturk.com

Table 3: The influence of prototype projection on ProtoPool performance for CUB-200-2011 and Stanford Cars datasets is negligible. Note that for CUB-200-2011, we used ResNet50 pretrained on iNaturalist.

	CUB-200-2011		**Stanford Cars**	
Architecture	Acc [%] before	Acc [%] after	Acc [%] before	Acc [%] after
ResNet34	80.8±0.2	80.3±0.2	89.1±0.2	89.3±0.1
ResNet50	85.9±0.1	85.5±0.1	88.4±0.1	88.9±0.1
ResNet152	81.2±0.2	81.5±0.1	—	—

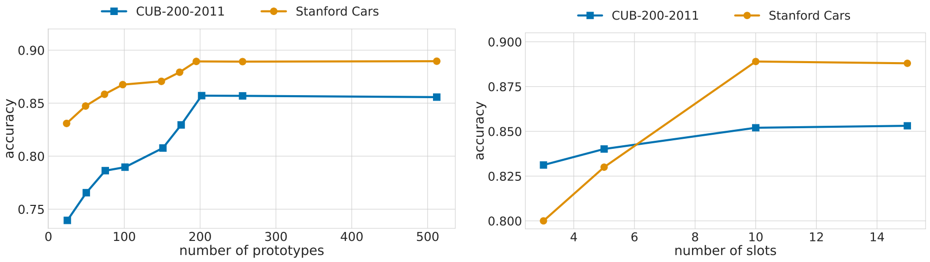
Table 4: The influence of novel architectural choices on ProtoPool performance for CUB-200-2011 and Stanford Cars datasets is significant. We consider training without orthogonalization loss, with softmax instead of Gumbel-Softmax, and with similarity from ProtoPNet instead of focal similarity. One can observe that the mix of the proposed mechanisms (i.e. ProtoPool) obtains the best accuracy.

	CUB-200-2011	**Stanford Cars**
Model	Acc [%]	Acc [%]
ProtoPool	**85.5**	**88.9**
w/o $\mathcal{L}_{orth}$	82.4	86.8
w/o Gumbel-Softmax trick	80.3	64.5
w/o Gumbel-Softmax trick and $\mathcal{L}_{orth}$	65.1	30.8
w/o focal similarity	85.3	88.8

Influence of the novel architectural choices Additionally, we analyze the influence of the novel components we introduce on the final results. For this purpose, we train ProtoPool without orthogonalization loss, with softmax instead of Gumbel-Softmax trick, and with similarity from ProtoPNet instead of focal similarity. Results are presented in Table 4 and in Supplementary Materials. We observe that the Gumbel-Softmax trick has a significant influence on the model performance, especially for the Stanford Cars dataset, probably due to lower inter-class similarity than in CUB-200-2011 dataset [34]. On the other hand, the focal similarity does not influence model accuracy, although as presented in Section 5, it has a positive impact on the interpretability. When it comes to orthogonality, it slightly increases the model accuracy by forcing diversity in slots of each class. Finally, the mix of the proposed mechanisms gets the best results.

Before and after prototype projection Since ProtoPool has much fewer prototypical parts than other models based on a positive reasoning process, applying projection could result in insignificant prototypes and reduced model performance. Therefore, we decided to test model accuracy before and after the projection (see Table 3), and we concluded that differences are negligible.

Number of prototypes and slots vs accuracy Finally, in Figure 11 we investigate how the number of prototypical parts or slots influences accuracy for



(a) Accuracy depending on the number of prototypes. One can observe that the model reaches a plateau for around 200 prototypical parts, and there is no gain in further increase of prototype number.

(b) Accuracy depending on the number of slots. One can observe that the model reaches a plateau for around 10 slots per class.

Fig. 11: ProtoPool accuracy with ResNet50 backbone depending on the number of prototypes and slots for CUB-200-2011 (blue square) and Stanford Cars (orange circle) datasets.

the CUB-200-2011 and Stanford Cars datasets. We observe that up to around 200 prototypical parts, the accuracy increases and reaches the plateau. Therefore, we conclude that the amount of prototypes optimal for ProtoTree is also optimal for ProtoPool. Similarly, in the case of slots, ProtoPool accuracy increases till the 10 slots and then reaches the plateau.

7 Conclusions

We presented ProtoPool, a self-explainable method that incorporates the paradigm of prototypical parts to explain its predictions. This model shares the prototypes between classes without pruning operations, reducing their number up to ten times. Moreover, it is fully differentiable. To efficiently assign the prototypes to classes, we apply the Gumbel-Softmax trick together with orthogonalization loss. Additionally, we introduced focal similarity that focuses on salient features. As a result, we increased the interpretability while maintaining high accuracy, as we showed through theoretical analysis, multiple experiments, and user study.

Acknowledgments

The research of J. Tabor, K. Lewandowska and D. Rymarczyk were carried out within the research project “Bio-inspired artificial neural network” (grant no. POIR.04.04.00-00-14DE/18-00) within the Team-Net program of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund. The work of Ł. Struski and B. Zieliński were supported by the National Centre of Science (Poland) Grant No. 2020/39/D/ST6/01332, and 2021/41/B/ST6/01370, respectively.

References

1. Abbasnejad, E., Teney, D., Parvaneh, A., Shi, J., Hengel, A.v.d.: Counterfactual vision and language learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10044–10054 (2020)
2. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), `https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf`
3. Afnan, M.A.M., Liu, Y., Conitzer, V., Rudin, C., Mishra, A., Savulescu, J., Afnan, M.: Interpretable, not black-box, artificial intelligence should be used for embryo selection. Human Reproduction Open (2021)
4. Alvarez Melis, D., Jaakkola, T.: Towards robust interpretability with self-explaining neural networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), `https://proceedings.neurips.cc/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf`
5. Barnett, A.J., Schwartz, F.R., Tao, C., Chen, C., Ren, Y., Lo, J.Y., Rudin, C.: Iaia-bl: A case-based interpretable deep learning model for classification of mass lesions in digital mammography. arXiv preprint arXiv:2103.12308 (2021)
6. Basaj, D., Oleszkiewicz, W., Sieradzki, I., Górszczak, M., Rychalska, B., Trzcinski, T., Zielinski, B.: Explaining self-supervised image representations with visual probing. In: International Joint Conference on Artificial Intelligence (2021)
7. Brendel, W., Bethge, M.: Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=SkfMWhAqYQ`
8. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: deep learning for interpretable image recognition. In: NeurIPS. pp. 8930–8941 (2019)
9. Chen, Z., Bei, Y., Rudin, C.: Concept whitening for interpretable image recognition. Nature Machine Intelligence **2**(12), 772–782 (2020)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
11. Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2950–2958 (2019)
12. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE international conference on computer vision. pp. 3429–3437 (2017)
13. Gee, A.H., Garcia-Olano, D., Ghosh, J., Paydarfar, D.: Explaining deep classification of time-series data with learned prototypes. In: CEUR workshop proceedings. vol. 2429, p. 15. NIH Public Access (2019)
14. Ghorbani, A., Wexler, J., Zou, J.Y., Kim, B.: Towards automatic concept-based explanations. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), `https://proceedings.neurips.cc/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf`
15. Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., Lee, S.: Counterfactual visual explanations. In: International Conference on Machine Learning. pp. 2376–2384. PMLR (2019)

16. Guidotti, R., Monreale, A., Matwin, S., Pedreschi, D.: Explaining image classifiers generating exemplars and counter-exemplars from latent representations. Proceedings of the AAAI Conference on Artificial Intelligence **34**(09), 13665–13668 (Apr 2020). https://doi.org/10.1609/aaai.v34i09.7116, `https://ojs.aaai.org/index.php/AAAI/article/view/7116`
17. Hase, P., Chen, C., Li, O., Rudin, C.: Interpretable image recognition with hierarchical prototypes. In: Proceedings of the AAAI Conference on Human Computation and Crowdsourcing. vol. 7, pp. 32–40 (2019)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
19. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
20. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv:1611.01144 (2016)
21. Kaminski, M.E.: The right to explanation, explained. In: Research Handbook on Information Law and Governance. Edward Elgar Publishing (2021)
22. Kesner, R.: A neural system analysis of memory storage and retrieval. Psychological Bulletin **80**(3), 177 (1973)
23. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In: International conference on machine learning. pp. 2668–2677. PMLR (2018)
24. Kim, E., Kim, S., Seo, M., Yoon, S.: Xprotonet: Diagnosis in chest radiography with global and local explanations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15719–15728 (2021)
25. Koh, P.W., Nguyen, T., Tang, Y.S., Mussmann, S., Pierson, E., Kim, B., Liang, P.: Concept bottleneck models. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5338–5348. PMLR (13–18 Jul 2020), `https://proceedings.mlr.press/v119/koh20a.html`
26. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proceedings of the IEEE international conference on computer vision workshops. pp. 554–561 (2013)
27. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
28. Liu, N., Zhang, N., Wan, K., Shao, L., Han, J.: Visual saliency transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4722–4732 (2021)
29. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st international conference on neural information processing systems. pp. 4768–4777 (2017)
30. Luria, A.: The origin and cerebral organization of man's conscious action. Children with learning problems: Readings in a developmental-interaction. New York, Brunner/Mazel pp. 109–130 (1973)
31. Marcos, D., Lobry, S., Tuia, D.: Semantically interpretable activation maps: what-where-how explanations within cnns. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 4207–4215. IEEE (2019)

32. Ming, Y., Xu, P., Qu, H., Ren, L.: Interpretable and steerable sequence learning via prototypes. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 903–913 (2019)
33. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. pp. 607–617 (2020)
34. Nauta, M., et al.: Neural prototype trees for interpretable fine-grained image recognition. In: CVPR. pp. 14933–14943 (2021)
35. Neisser, U.: Cognitive psychology (new york: Appleton). Century, Crofts (1967)
36. Niu, Y., Tang, K., Zhang, H., Lu, Z., Hua, X.S., Wen, J.R.: Counterfactual vqa: A cause-effect look at language bias. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12700–12710 (2021)
37. Puyol-Antón, E., Chen, C., Clough, J.R., Ruijsink, B., Sidhu, B.S., Gould, J., Porter, B., Elliott, M., Mehta, V., Rueckert, D., et al.: Interpretable deep models for cardiac resynchronisation therapy response prediction. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 284–293. Springer (2020)
38. Rebuffi, S.A., Fong, R., Ji, X., Vedaldi, A.: There and back again: Revisiting backpropagation saliency methods. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8839–8848 (2020)
39. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
40. Rosch, E.: Cognitive representations of semantic categories. Journal of experimental psychology: General **104**(3), 192 (1975)
41. Rosch, E.H.: Natural categories. Cognitive psychology **4**(3), 328–350 (1973)
42. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
43. Rymarczyk, D., et al.: Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification. In: SIGKDD. pp. 1420–1430 (2021)
44. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
45. Selvaraju, R.R., Lee, S., Shen, Y., Jin, H., Ghosh, S., Heck, L., Batra, D., Parikh, D.: Taking a hint: Leveraging explanations to make vision and language models more grounded. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2591–2600 (2019)
46. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: In Workshop at International Conference on Learning Representations. Citeseer (2014)
47. Singh, G., Yow, K.C.: These do not look like those: An interpretable deep learning model for image recognition. IEEE Access **9**, 41482–41493 (2021)
48. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning. pp. 3319–3328. PMLR (2017)
49. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8769–8778 (2018)

50. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
51. Wang, J., et al.: Interpretable image recognition by constructing transparent embedding space. In: ICCV. pp. 895–904 (2021)
52. Wang, P., Vasconcelos, N.: Scout: Self-aware discriminant counterfactual explanations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8981–8990 (2020)
53. Wiegand, G., Schmidmaier, M., Weber, T., Liu, Y., Hussmann, H.: I drive-you trust: Explaining driving behavior of autonomous cars. In: Extended abstracts of the 2019 chi conference on human factors in computing systems. pp. 1–6 (2019)
54. Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z.: The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 842–850 (2015)
55. Yeh, C.K., Kim, B., Arik, S., Li, C.L., Pfister, T., Ravikumar, P.: On completeness-aware concept-based explanations in deep neural networks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 20554–20565. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/ecb287ff763c169694f682af52c1f309-Paper.pdf
56. Zhang, Z., Liu, Q., Wang, H., Lu, C., Lee, C.: Protgnn: Towards self-explaining graph neural networks (2022)
57. Zheng, H., Fu, J., Mei, T., Luo, J.: Learning multi-attention convolutional neural network for fine-grained image recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 5209–5217 (2017)
58. Zheng, H., Fu, J., Zha, Z.J., Luo, J.: Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5012–5021 (2019)
59. Zhou, B., Sun, Y., Bau, D., Torralba, A.: Interpretable basis decomposition for visual explanation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 119–134 (2018)