Interpretable Image Classification with Differentiable Prototypes Assignment – Supplementary Materials

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

1 Details on experimental setup

We use two datasets, CUB-200-2011 [10] consisted of 200 species of birds and Stanford Cars [5] with 196 car models. For both datasets, images are augmented offline using parameters from Table 1, and the process of data preparation is the same as in [1][4].

Table 1: Augmentation policy.

Augmentation	Value	Probability
Rotation	$[-15°, 15°]$	1.0
Flip	Vertical	0.5
Flip	Horizontal	0.5
Skew	$< 45°$	0.5
Shear	$[-10°, 10°]$	1.0
Mix-up [9]	50% : 50%	1.0

Our model consists of the convolutional part f that is a convolutional block from ResNet or DenseNet followed by 1×1 convolutional layer required to transform the latent space depth to 128 for Stanford Cars and 256 for CUB-200-2011. We perform a warmup training where the weights of f are frozen for 10 epochs, and then we train the model until it converges with 12 epochs early stopping. After convergence, we perform prototype projection and fine-tune the last layer. We use the learning schema presented in Table 2.

Additionally, we employ Adam optimizer [4] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the batch size to 80 and use input images of resolution

[4] see *Instructions for preparing the data* at `https://github.com/cfchen-duke/ProtoPNet`

Table 2: Learning schema for the ProtoPool model.

Phase	Model layers	Learning rate	Scheduler	Weight decay	Duration
Warm-up	add-on 1×1 convolution prototypical pool	$1.5 \cdot 10^{-3}$ $1.5 \cdot 10^{-3}$	None	None	10 epochs
Joint	convolutions f add-on 1×1 convolution prototypical pool	$5 \cdot 10^{-5}$ $1.5 \cdot 10^{-3}$ $1.5 \cdot 10^{-3}$	by half every 5 epochs	10^{-3}	12 epoch early stopping
After projection	last layer	10^{-4}	None	None	15 epochs

$224 \times 224 \times 3$. Moreover, we use prototypical parts of size $1 \times 1 \times 128$ and $1 \times 1 \times 256$ for Stanford Cars and CUB-200-2011 respectively. The weights between the class logit and its slots are initialized to 1, while the remaining weights of the last layer are set to 0. All other parameters of the network are initialized with Xavier's normal initializer.

We utilize the Gumbel-Softmax trick to unambiguously assign prototypes to class slots. However, in contrast to the classic variant of this parametrization trick, we reduce the influence of the noise in subsequent iterations. For this purpose, we use $y^i = \frac{\exp\left(q^i/\tau + \eta_i\right)}{\sum_{m=1}^{M} \exp(q^m/\tau + \eta_m)}$ instead of $y^i = \frac{\exp\left((q^i + \eta_i)/\tau\right)}{\sum_{m=1}^{M} \exp((q^m + \eta_m)/\tau)}$ in Gumbel-Softmax distribution

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

where $\tau \in (0, \infty)$ and $q \in \mathbb{R}^M$. Moreover, we start the Gumbel-Softmax distribution with $\tau = 1$, decreasing it to 0.001 for 30 epochs. As a decrease function, we use

$$\tau(epoch) = \begin{cases} 1/\sqrt{\alpha \cdot \text{epoch}} & \text{if epoch } < 30 \\ 0.001 & \text{otherwise} \end{cases},$$

where $\alpha = 3.4 \cdot 10^4$. We use the following weighting schema for loss function: $\mathcal{L}_{entropy} = 1.0$, $\mathcal{L}_{clst} = 0.8$, $\mathcal{L}_{sep} = -0.08$, $\mathcal{L}_{orth} = 1.0$, and $\mathcal{L}_{l_1} = 10^{-4}$. Finally, we normalize $\mathcal{L}_{orth}$, dividing it by the number of classes multiplied by the number of slots per class.

2 Generating names of prototypical parts

To name the prototype for CUB-200-2011, we used the attributes of images collected with Amazon Mechanical Turk that are provided together with the dataset. Firstly, for a given image, we filter out attributes assigned by less than 20% users. Then, for each class, we remove attributes present at less than 20% of testing images assigned to this class. Later, we use five nearest patches of a given prototype to determine if they are consistent and point to the same part of the bird. Eventually, we choose the attributes accurately describing the nearest patches for a given prototype (see Figure 1 from the main paper).

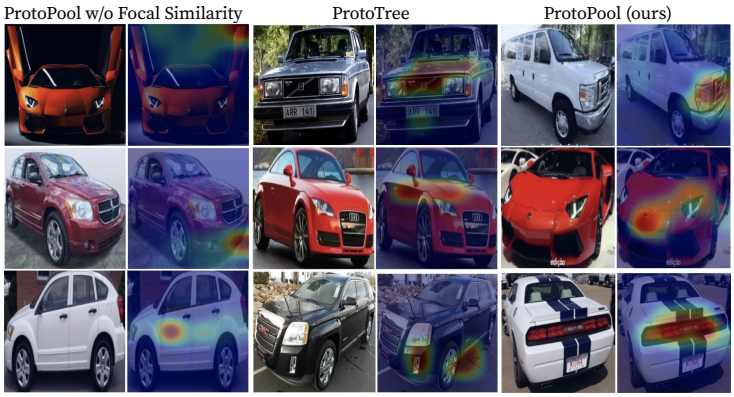


Fig. 1: Sample prototype activation maps obtained with models using various similarity functions. One can observe that ProtoPool with the focal similarity is the only one that focuses on car features like a spoiler, reflector, and *Ferrari* logo when the other similarities focus on image borders or much larger regions. Since similarity function is a design choice of the model, the images from which the prototypical parts derive are different.

3 Results for other backbone networks

In Table 3, we present the results for ProtoPool with DenseNet as a backbone network. Moreover, in Figure 1, we present examples of prototypes derived from three different methods: ProtoPool, ProtoTree, and ProtoPool without focal similarity.

4 Comparison of the prototypical models

In Table 4, we present the extended version of prototypical-based model comparison. One can observe that ProtoPool is the only one that has a differentiable prototypical parts assignment. Additionally, it processes the data simultaneously, which is faster and easier to comprehend by humans [2] than sequential processing obtained from the ProtoTree [6]. Lastly, ProtoPool, as ProtoPShare, directly provides class similarity that is visualized in the Figure 2.

5 Details on ablation study

As an attachment to Table 4 from the main paper, in Figure 3 we provide the matrices of prototype assignment. Moreover, in Figure 4, we present the distribution of values from the prototype assignment matrix for corresponding datasets. As presented, only the ProtoPool model obtains bimodal distribution of 0 and 1, resulting in the binary matrix.

Table 3: Comparison of ProtoPool with other methods based on prototypical parts trained on the CUB-200-2011 and Stanford Cars datasets, considering a various number of prototypes and types of convolutional layers f. ProtoPool achieves competitive results, even for models with ten times more prototypes in the case of both datasets. Please note that the results are first sorted by backbone network and then by the number of prototypes.

Data	Model	Architecture	Proto. #	Acc [%]
CUB-200-2011	ProtoPool (ours)	DenseNet121	202	73.6 ± 0.4
	ProtoPShare [8]		600	74.7
	ProtoPNet [1]		1476	79.2
	TesNet [11]		2000	$\mathbf{84.8 \pm 0.2}$
	ProtoPool (ours)	DenseNet161	202	80.3 ± 0.3
	ProtoPShare [8]		600	76.5
	ProtoPNet [1]		1527	79.9
	TesNet [11]		2000	84.6 ± 0.3
Cars	ProtoPool (ours)	DenseNet121	202	86.4 ± 0.1
	ProtoPShare [8]		980	84.8
	ProtoPNet [1]		2000	86.8 ± 0.1
	TesNet [11]		2000	$\mathbf{92.0 \pm 0.3}$

Table 4: Comparison of prototypical methods for fine-grained image classification. One can observe that ProtoPool uses only 10% of ProtoPNet's prototypes, remaining interpretable due to the positive reasoning process. Additionally, ProtoPool, similarly to the ProtoPShare, detects inter-class similarity and shares the prototypes. But, it is trained end-to-end thanks to the differentiable prototypes assignment. On the other hand, ProtoTree is the only model presenting the explanation in a hierarchical way, which requires more time to comprehend according to human cognitive system theory [2].

Model	Proto. #	Diff. proto. assignment	Information processing	Reasoning type	Proto. sharing	Class similarity
ProtoPNet	100%	no	simultaneous	positive	none	none
TesNet	100%	no	simultaneous	positive	none	none
ProtoPShare	[20%;50%]	no	simultaneous	positive	direct	direct
ProtoTree	10%	no	successive	positive/negative	indirect	indirect
ProtoPool	10%	yes	simultaneous	positive	direct	direct

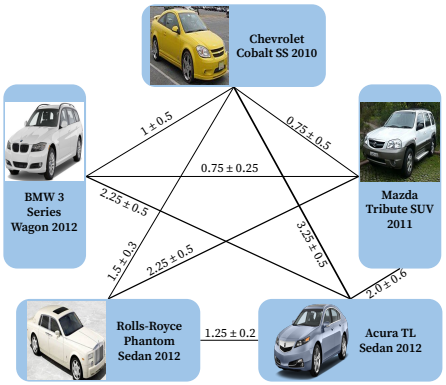
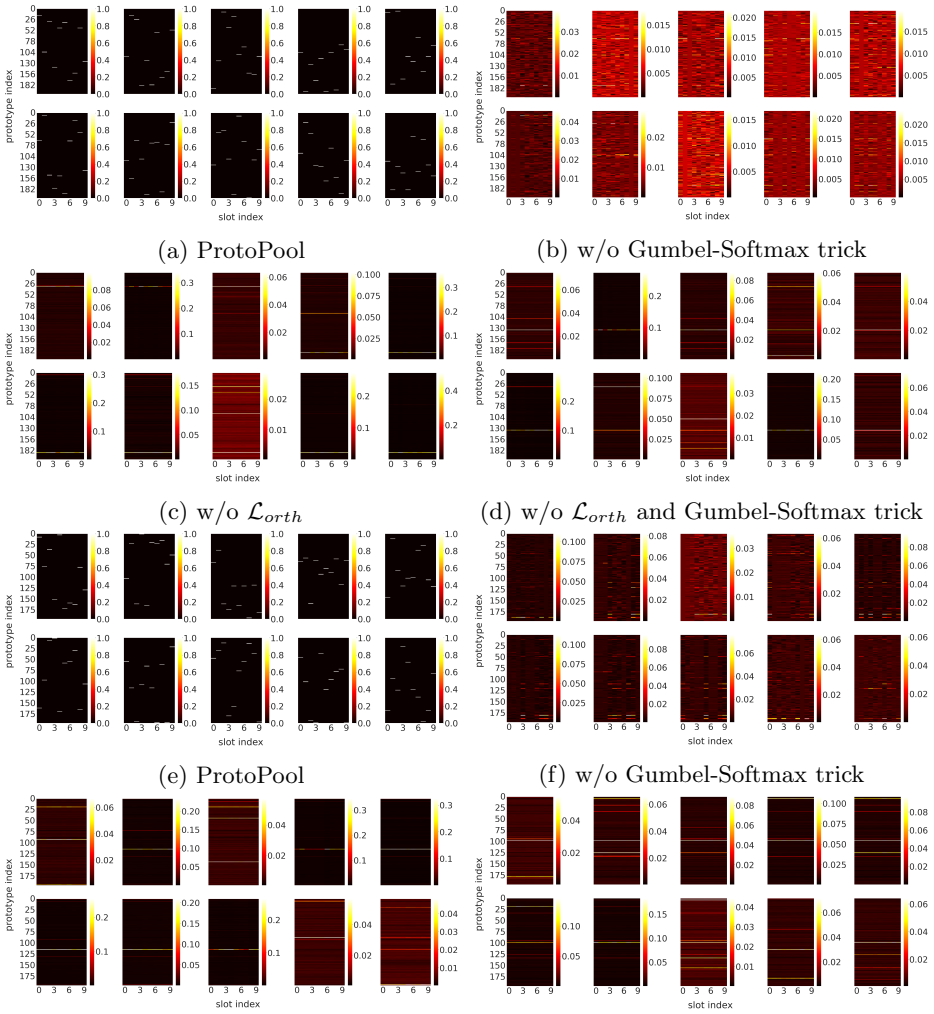


Fig. 2: Sample graph obtained from ProtoPool for five classes from the Stanford Cars dataset. Each class is represented by a single image, and the edges correspond to the mean number of prototypes shared between classes over five repetitions. One can observe that ProtoPool discovered similarities between SUVs and Sedans, while Rolls-Royce and BMW have nothing in common. Moreover, the graphs are consistent between runs and discover similar relations between classes.

6 Details on user study

To ensure a broad spectrum of the users, we ran the AMT batches at four different hours (8 AM, 2 PM, 8 PM, 12 PM CET) and required balanced sex in (53% of women) and versatile age (from 20 to 60) of the users. Each user assessed examples of prototypical parts generated by ProtoPool, ProtoTree [7] and ProtoPool without focal similarity in a randomized order. The user did not know which image was generated by which model, and there was no difference in preprocessing those images between models. Each person answered ten questions for each dataset and model combination, resulting in 60 responses per participant. These 60 images were randomly selected from the pool of 180 images (30 for each combination of dataset and model). Each user had unlimited time for the answer. The task was to assign a score from 1 to 5 where 1 meant *"Least salient"* and 5 meant *"Most salient"*. A sample question is presented in Figure 5 and the results are shown in Table 5. One can observe that ProtoPool achieves the highest number of positive answers (4 and 5). Additionally, ProtoTree is better for Stanford Cars rather than CUB-200-2011, which can be correlated to the weaker intra-class similarity in the case of car models [7]. Overall, we conclude that the enrichment of the model with focal similarity substantially improves the model interpretability and better detects salient features.



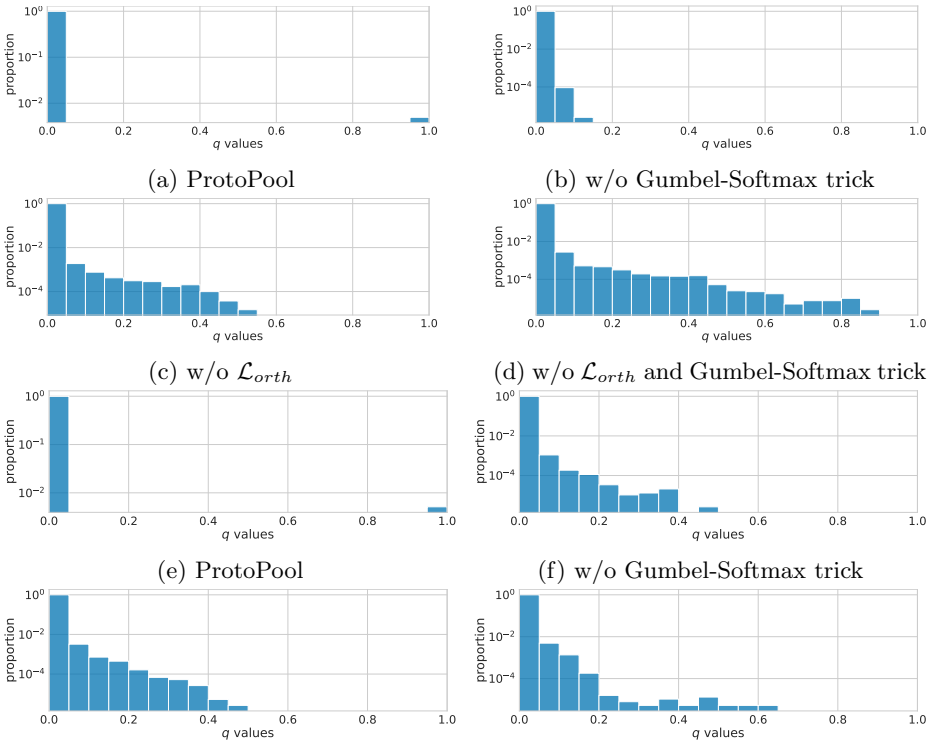
(a) ProtoPool (b) w/o Gumbel-Softmax trick

(c) w/o $\mathcal{L}_{orth}$ (d) w/o $\mathcal{L}_{orth}$ and Gumbel-Softmax trick

(e) ProtoPool (f) w/o Gumbel-Softmax trick

(g) w/o $\mathcal{L}_{orth}$ (h) w/o $\mathcal{L}_{orth}$ and Gumbel-Softmax trick

Fig. 3: The influence of novel architectural changes on prototypes to slots assignments for ten randomly chosen classes of the CUB-200-2011 (a-d) and Stanford Cars (e-h) datasets. Each class has ten slots (corresponding to columns) to which a prototype (corresponding to rows) can be assigned. As observed, the binarization of the assignment (hard assignment of a prototype) is obtained only for a mix of Gumbel-Softmax and L_{orth}. Moreover, if one of those factors is missing, the assignment matrix is random or aims to assign the same prototype to all slots. Note that the scale of heatmap colors differs between examples for clarity.



(a) ProtoPool (b) w/o Gumbel-Softmax trick

(c) w/o $\mathcal{L}_{orth}$ (d) w/o $\mathcal{L}_{orth}$ and Gumbel-Softmax trick

(e) ProtoPool (f) w/o Gumbel-Softmax trick

(g) w/o $\mathcal{L}_{orth}$ (h) w/o $\mathcal{L}_{orth}$ and Gumbel-Softmax trick

Fig. 4: The influence of novel architectural changes on the values of prototypes to slots assignments (q distributions) for the CUB-200-2011 (a-d) and Stanford Cars (e-h) datasets. One can observe that only the ProtoPool model binarizes q distributions. Note that the histograms are in logarithmic scale and normalized.

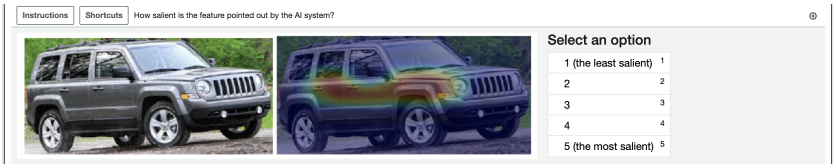


Fig. 5: Sample question from the user study questionnaire.

Table 5: User study results show that only ProtoPool has the most positive votes (4 and 5) in both datasets. Additionally, ProtoTree is more interpretable for Stanford Cards than for CUB-200-2011.

Model	Dataset	Answers				
		1	2	3	4	5
ProtoPool	CUB-200-2011	3	26	107	151	113
ProtoTree		143	76	73	53	55
ProtoPool w/o focal similarity		100	61	88	98	53
ProtoPool	Stanford Cars	12	57	139	116	76
ProtoTree		21	101	106	108	64
ProtoPool w/o focal similarity		70	103	96	79	52

7 Limitations

Our ProtoPool model inherits its limitations from the other prototype-based models, including non-obvious prototype meaning. Hence, even after prototype projection from a training dataset, there is still uncertainty on which attributes it represents. However, there exist ways to mitigate this limitation, e.g. using a framework defined in [6]. Additionally, the choice of Gumbel-Softmax temperature τ and its decreasing strategy are not straightforward and require a careful hyperparameter search. Lastly, in the case of ProtoPool, increasing the number of prototypes does not increase the model accuracy after some point because the model saturates.

8 Negative impact

We base our solution on prototypical parts, which are vulnerable to a new type of adversarial attacks [3]. Hence, practitioners must consider this danger when deploying a system with a ProtoPool. Additionally, it can spread disinformation when prototypes derive from spoiled data or are presented without an expert comment, especially in fields like medicine.

9 Additional discussion

Why focal similarity works – the intuition. Focal similarity computes the similarity between patches and prototypes, which is then passed to the classification layer of the network, where standard CE loss is used. The big advantage of focal similarity is its ability to propagate gradient through all patches by subtracting the mean from the maximum similarity. In contrast to the original approach [8], which propagates gradient only through the patch with the maximum similarity. This way, ProtoPool generates salient prototypes that activate

only in a few locations and return values close to zero for the remaining image parts (see Fig. 6). From this perspective, using the median instead of the mean would again limit gradient propagation to two patches (with maximum and median similarity).

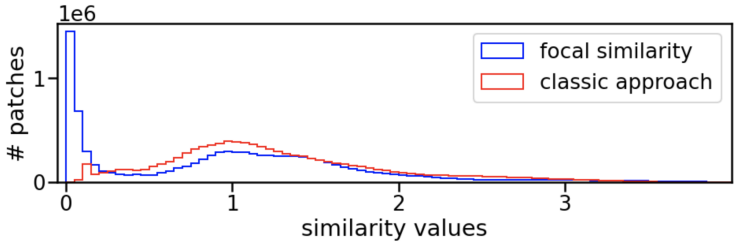


Fig. 6: Distribution of similarity values for focal similarity and classic approach [8] over 1000 images. As a result of replacing the original approach with focal similarity, the distribution changes from unimodal to bimodal.

Saturation of model capacity. The model reaches a plateau for around 200 prototypes, and there is no gain in further increase of prototype number. Therefore, the practitioners cannot sacrifice some interpretability to gain higher accuracy. In fact, this trend is also observed in the other methods with shared prototypes, like ProtoTree (see Fig. 7 in [7]). While we have no clear explanation for this phenomenon, we assume it can be caused by the entanglement of the prototypes. Therefore, one possible solution would be to enforce the prototypes orthogonality, as proposed in TesNet [11]. The other option would be to modify the training procedure so that it iteratively adds new slots to each class corresponding to new pools of prototypes.

Focal similarity vs. reasoning type While the negative reasoning process draws conclusions based on the prototype's absence ("this does not look like that prototype"), the focal similarity concludes based on the prototype's presence ("this looks like that salient prototype, which usually occurs only one time"). For example, the negative reasoning could say: "this is a goat because it has no wings", while the focal similarity would rather say: "this is a goat because it has a goatee (a salient goat feature)".

Necessity for sharing prototypes We would like to recall the observation provided in [8]. It shows that after training ProtoPNet with exclusive prototypes, patches' representations are clustered around prototypes of their true classes, and the prototypes from different classes are well-separated. As a result, the prototypes with similar semantics can be distant in representation space (see Fig. 2 in [8]), resulting in unstable predictions. That is why it is essential to share the prototypes between classes.

User studies statistics We provide additional statistics regarding the results of user studies. The confidence intervals are as follows: ProtoPool (our):

3.66 ± 2.00, ProtoPool w/o focal similarity: 2.85 ± 2.63, ProtoTree: 2.87 ± 2.68. Moreover, we performed a Mann-Whitney U test to determine whether of ProtoPool (our) scores are higher than 'ProtoPool w/o focal similarity and ProtoTree. We obtained p-value$= 2.78 \cdot 10^{-12}$ and $1.16 \cdot 10^{-11}$, respectively. Since both p-values are smaller than 0.05, we reject the null hypothesis and conclude that the ProtoPool is significantly better than other methods.

Prototypes in a hard vs soft assignment Through the development process, we experimented with the soft assignment (Softmax instead of Gumbel-Softmax). However, we observed that the model struggles to separate the prototypes in the latent space, even with the increased weight of a separation cost, and prototypes usually converge to one point in representation space. On the other hand, using only the hard assignments hinders the change in assignments during training. That is why we decided to use a hybrid approach, where we start with soft assignments and binarize them with Gumbel-Softmax. It allows the cluster and separation costs to roughly organize latent space with a soft assignment at the beginning and then refine it as the hard assignment dominates.

References

1. 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)
2. Fiske, S.T., Taylor, S.E.: Social cognition. Mcgraw-Hill Book Company (1991)
3. Hoffmann, A., Fanconi, C., Rade, R., Kohler, J.: This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. arXiv preprint arXiv:2105.02968 (2021)
4. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: ICLR 2015 : International Conference on Learning Representations 2015 (2015)
5. 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)
6. Nauta, M., Jutte, A., Provoost, J., Seifert, C.: This looks like that, because... explaining prototypes for interpretable image recognition. arXiv preprint arXiv:2011.02863 (2020)
7. Nauta, M., et al.: Neural prototype trees for interpretable fine-grained image recognition. In: CVPR. pp. 14933–14943 (2021)
8. Rymarczyk, D., et al.: Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification. In: SIGKDD. pp. 1420–1430 (2021)
9. Thulasidasan, S., Chennupati, G., Bilmes, J.A., Bhattacharya, T., Michalak, S.: On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In: Advances in Neural Information Processing Systems. vol. 32, pp. 13888–13899 (2019)
10. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
11. Wang, J., et al.: Interpretable image recognition by constructing transparent embedding space. In: ICCV. pp. 895–904 (2021)