ProtoPShare: Prototypical Parts Sharing for Similarity Discovery in Interpretable Image Classification

Dawid Rymarczyk

dawid.rymarczyk@student.uj.edu.pl

Faculty of Mathematics and Computer Science,
Jagiellonian University and Ardigen SA
Cracow, Poland

Jacek Tabor

jacek.tabor@uj.edu.pl

Faculty of Mathematics and Computer Science,
Jagiellonian University
Cracow, Poland

Łukasz Struski

lukasz.struski@uj.edu.pl

Faculty of Mathematics and Computer Science,
Jagiellonian University
Cracow, Poland

Bartosz Zieliński

bartosz.zielinski@uj.edu.pl

Faculty of Mathematics and Computer Science,
Jagiellonian University and Ardigen SA
Cracow, Poland

Figure 1: Comparison between ProtoPNet with exclusive prototypes and our ProtoPShare that shares prototypes between classes. The number of prototypes in ProtoPNet is large because each of them is assigned to only one class. E.g. in this example, classes *Henslow sparrow* and *Lincoln sparrow* have separate prototypes corresponding to *Eye* in ProtoPNet. In ProtoPShare, the similar prototypes are merged into one, decreasing model complexity and discovering similarities between classes.

ABSTRACT

In this work, we introduce an extension to ProtoPNet called ProtoPShare which shares prototypical parts between classes. To obtain prototype sharing we prune prototypical parts using a novel data-dependent similarity. Our approach substantially reduces the number of prototypes needed to preserve baseline accuracy and finds prototypical similarities between classes. We show the effectiveness of ProtoPShare on the CUB-200-2011 and the Stanford Cars datasets and confirm the semantic consistency of its prototypical parts in user-study.

CCS CONCEPTS

• Computing methodologies → Computer vision; Neural networks.

KEYWORDS

prototypical parts; interpretability; explainability; neural networks

ACM Reference Format:

Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. 2021. ProtoPShare: Prototypical Parts Sharing for Similarity Discovery in Interpretable Image Classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21)*, August 14–18, 2021, Virtual Event, Singapore. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3447548.3467245>

1 INTRODUCTION

Broad application of deep learning in domains like medical diagnosis and autonomous systems enforces models to explain their decisions. Ergo, more and more methods provide human-understandable justifications for their output [2–5, 8, 10, 13, 30]. Some of them are inspired by the human brain and how it explains its visual judgments by pointing to prototypical features that an object possesses [32].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08...\$15.00
<https://doi.org/10.1145/3447548.3467245>

Figure 2: Example prototypes similar according to our data-dependent similarity but distant in representation space. The tSNE projections of prototypes (left plots) trained for CUB-200-2011 and Stanford Cars datasets (top and bottom part, respectively) contains two prototypes marked as red/green squares. They correspond to semantically similar prototypical parts (top representing *bright belly with grayish wings* and bottom representing *fender*) that are distant in the representation space. Such pairs can be discovered by our data-dependent similarity defined in Section 3. Notice that each example prototype is represented by one row of 3 closest images’ parts (marked with yellow bounding-boxes). More examples are presented in Figure 11.

I.e., a certain object is a car because it has tires, roof, headlights, and horn.

Recently introduced *Prototypical Part Network (ProtoPNet)* [4] applies this paradigm by focusing on parts of an image and comparing them with *prototypical parts* of a given class. This comparison is achieved by pushing parts of an image and prototypical parts through the convolutional layers, obtaining their representation, and computing similarity between them. We will refer to the representations of prototypical parts as *prototypes*.

ProtoPNet is a self-explaining model that generates intuitive explanations and achieves accuracy comparable with its analogous non-interpretable counterparts. However, it has limited applicability due to two main reasons. Firstly, the number of prototypes is large because each of them is assigned to only one class. It negatively influences the interpretability whose much-desirable properties are small size and low complexity [7, 39]. Secondly, due to the training that pushes away the prototypes of different classes in the representation space, the prototypes with similar semantic can be distant (see Figure 2). Therefore, the predictions obtained from the network can be unstable.

In this work, we address those limitations by introducing the *Prototypical Part Shared network (ProtoPShare)*¹ that shares the prototypes between the classes, like presented in Figure 1. As a

¹The code is available here: <https://github.com/gmum/ProtoPShare>.

Figure 3: Inter class similarity visualization as a graph generated based on the prototypes shared in ProtoPShare trained on Stanford Cars dataset. Each node corresponds to a class, and the strength of the edge between two nodes corresponds to the number of shared prototypes. E.g. *Audi S5 Coupe* shares 4 prototypes with *Audi S5 Convertible* but does not share prototypes with *Acura RL Sedan*. This graph can be used to find similarities between the classes or to cluster them into groups. Notice that each class is represented by three images located around the node.

result, the number of prototypes is relatively small and prototypes with similar semantic are close to each other. Additionally, it is possible to discover similarities between the classes, as presented in Figure 3. ProtoPShare method consists of two phases, initial training and prototypes’ pruning. In the first phase, the model is trained using exclusive prototypes and a loss function described in [4]. In the second phase, the iterative pruning merges successive portions of the most similar prototypes. For this purpose, we introduce the data-dependent similarity described in Section 3 that finds prototypes with similar semantics, even if they are distant in representation space. We demonstrate the superiority of the ProtoPShare approach by comparing it to the other methods based on the prototypes’ paradigm. Our contribution can be therefore summarized as follows:

- We construct ProtoPShare, a self-explained method built on the paradigm of prototypical parts that shares prototypes between the classes.
- We introduce data-dependent similarity that can find semantically similar prototypes even if they are distant in the representation space.
- We significantly reduce the number of prototypes comparing to state of the art, reducing model complexity and discovering similarities between classes.

This paper is organized as follows. Section 2 investigates related works on interpretability and pruning. In Section 3, we introduce our ProtoPShare method together with its theoretical understanding. Section 4 illustrates experimental results on CUB-200-2011 and

Figure 4: The architecture of our ProtoPShare. It consists of convolutional layers f , followed by a prototype layer g , and a fully connected layer h . Layer g contains prototypes that are shared between the classes. E.g. in this example, prototype p_2 is assigned both to class *Common yellowthroat* and *Prothonotary warbler* which corresponds to stronger connections between p_2 and both classes in layer h . As a result of sharing the prototypes, we reduce their number and discover similarities between classes.

Stanford Cars datasets, while Section 5 concentrates on ProtoPShare interpretability. Finally, we conclude the work in Section 6.

2 RELATED WORKS

ProtoPShare is a self-explained method with a strong focus on the prototypes' pruning. Therefore, in the related works, we consider the articles about interpretability and pruning.

Interpretability. Interpretability approaches can be divided into post hoc and self-explaining methods [2]. *Post hoc* techniques include saliency maps, showing how important each pixel of a particular image is for its classification [30, 33–35]. Another technique called concept activation vectors provides an interpretation of the internal network state in terms of human-friendly concepts [5, 12, 19, 42]. Other methods analyze how the network's response changes for perturbed images [8, 9, 31]. *Post hoc* methods are easy to use in practice as they do not require changes in the architecture. However, they can produce unfaithful and fragile explanations [1]. Therefore, as a remedy, *self-explainable* models were introduced by applying the attention mechanism [10, 36, 40, 44, 45] or the bag of local features [3]. Other works [13, 23, 29] focus on exploiting the latent feature space obtained, e.g. with adversarial autoencoder. The interesting self-explaining methods are prototype-based models represented, like Prototypical Part Network [4] with a hidden layer of prototypes representing the activation patterns. A similar approach for hierarchically organized prototypes is presented in [14] to classify objects at every level of a predefined taxonomy. Some works concentrate on transforming prototypes from the latent space to data space [14]. The others try to apply prototypes to other domains like sequence learning [27] or time series analysis [11].

Pruning. Pruning is mostly used to accelerate deep neural networks by removing unnecessary weights or filters [25]. The latter can be roughly divided into data-dependent and data-independent

filter pruning [16], depending on whether the data is used or not to determine the pruned filters. The basic type of *data-independent* pruning removes filters with the smallest sum of absolute values of weights [22]. More complicated approaches designate irrelevant or redundant filters using scaling parameters of batch normalization layers [41] or the concept of geometric median [16]. *Data-dependent* pruning can be solved as an optimization problem on the statistics computed from its subsequent layer [26], by minimizing the reconstruction error on the subsequent feature map using Lasso method [17], or by using a criterion based on Taylor expansion that approximates the change in the cost function induced by pruning [28]. Filters' importance can also be propagated from the final response layer to minimize its reconstruction error [43]. Another possibility is to introduce additional discrimination-aware losses and select the most discriminative channels for each layer [46]. Finally, feature maps can be clustered and replaced by the average representative from each cluster [38]. Moreover, filters can be pruned together with other structures of the network [24].

Our ProtoPShare extends the ProtoPNet [4] by the shared prototypes obtained with the data-dependent pruning based on the feature maps.

3 PROTOPSHARE

In this section, we first describe the architecture of ProtoPShare and then define our data-dependent merge-pruning algorithm. Finally, we provide a theoretical understanding of how the prototypes' merge affects classification accuracy.

Architecture. The architecture of ProtoPShare, shown in Figure 4, consists of convolutional layers f , followed by a prototype layer g , and a fully connected layer h (with weight w_h and no bias). Given an input image $x \in X$, the convolutional layers extract image representation $f(x)$ of shape $H \times W \times D$. For the clarity of description, let $Z_x = \{z_i \in f(x) : z_i \in \mathbb{R}^D, i = 1..HW\}$. Then, for

Figure 5: The process of merging two prototypes. If prototypes p_1 and p_3 are similar (e.g. they both present a birds' leg), we delete one of them (in this case p_3). Moreover, to compensate this removal, we assign p_1 to p_3 's class and add p_3 's weights to p_1 's weights in h (here done by summing weights represented as the blue and green arrows).

each class k , the network learns m_k prototypes $P_k = \{p_i\}_{i=1}^{m_k}$, where $p_i \in \mathbb{R}^D$ represents prototypical parts trained for class k . Given a convolutional output Z_x and prototype p , ProtoPSHare computes the distances between them, inverts them to obtain the similarity scores, and takes the maximum among all $z \in Z_x$:

$$g(Z_x, p) = \max_{z \in Z_x} \log \left(\frac{\|z-p\|^2 + 1}{\|z-p\|^2 + \epsilon} \right) \text{ for } \epsilon > 0. \quad (1)$$

Finally, the similarity scores produced by the prototype layer (m_k values per class) are multiplied by the weight matrix w_h in the fully connected layer h . This results in the output logits that are normalized using softmax to obtain a final prediction.

After training with exclusive prototypes and a loss function adapted from [4], most representations of the image's parts are clustered around semantically similar 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 Figure 2), resulting in unstable predictions. In the next paragraph, we present how to overcome this issue with our data-dependent merge-pruning.

Data-dependent merge-pruning. Here, we describe the pruning phase of our method. As the input, it obtains the network trained with exclusive prototypes, and as the output, it returns the network with a smaller number of shared prototypes.

Let P be the set of all prototypes (after training phase), K be the number of classes, and ζ be the percentage of prototypes to merge per pruning step. Each step begins with computing the similarities between the pairs of prototypes, then for each pair $(p, \tilde{p})$ among ζ percent of the most similar pairs, the prototype p is removed together with its weights $w_h(p)$. In exchange, the class to which p was assigned reuses prototype $\tilde{p}$ whose weights $w_h(\tilde{p})$ are aggregated with $w_h(p)$. We present this procedure in Figure 5. Note that in each step, we cannot always merge exactly ζ percent of prototypes. It is because the similarity metric can have the same value for many pairs. Lastly, we finetune layer h after each step.

As described in the architecture description, training with a loss function described in [4] can result in prototypes of similar semantic that are far from each other. To overcome this problem, we introduce a special type of data-dependent similarity. It uses training images $x \in X$ to generate the representation of all training patches, and compare the difference between $g(Z_x, p)$ and $g(Z_x, \tilde{p})$. The *data-dependent similarity* for pair of prototypes $(p, \tilde{p}) \in P^2$ is defined as the compliance on the similarity scores computed for all patches $x \in X$:

$$d_{DD}(p, \tilde{p}) = \frac{1}{\sum_{x \in X} (g(Z_x, p) - g(Z_x, \tilde{p}))^2}. \quad (2)$$

As a result, prototypes are considered similar if they activate alike on the training images, even if they are far from each other in the representation space.

Theoretical results. The following theorem provides some theoretical understanding of how the prototypes' merge affects the classification accuracy. Intuitively, the theorem states that if the pruning merges similar prototypes, then the predictions after merge do not change if predictions before the merge had certain confidence. Due to the page limits, the proof is in the Appendix A.

THEOREM 1. *Let $x \in X$ be an input image correctly classified by ProtoPSHare $h \circ g_p \circ f$ as class c before the prototypes' merge, and let:*

- (A1) *some of the class k prototypes remain unchanged $P'_k \subset P_k$ while the others merge into the other classes' prototypes $S_k \subset \bigcup_{i \neq k} P'_i$,*
- (A2) *$p \in P_k \setminus P'_k$ is merged into $\tilde{p} \in S_k$,*
- (A3) *$z_p = \arg \min_{z \in \bigcup_{x \in X} Z_x} \|z - p\|$ is the representation of the training patch nearest to any prototype p ,*
- (A4) *there exist $\delta \in (0, 1)$ such that:*
 - a) *for $k \neq c$, $p \in P_k \setminus P'_k$ and $\tilde{p} \in S_k$, we suppose that $\|p - \tilde{p}\| \leq \theta \|z_p - p\| - \sqrt{\epsilon}$ and $\theta = \min(\sqrt{1 + \delta} - 1, 1 - \frac{1}{\sqrt{2 - \delta}})$ (ϵ comes from function g defined in (1)),*
 - b) *for $p \in P_c \setminus P'_c$ and $\tilde{p} \in S_c$, we suppose that $\|\tilde{p} - p\| \leq (\sqrt{1 + \delta} - 1) \|z_p - p\|$ and $\|z_p - p\| \leq \sqrt{1 - \delta}$,*
- (A5) *for each class k , weights connecting class with assigned prototypes equal 1, and the other weights equal 0 (i.e., $w_h(p) = 1$ for $p \in P_k \cup S_k$ and $w_h(p) = 0$ for $p \in \bigcup_{i \neq k} P_i \setminus S_k$).*

Figure 6: The accuracy of ProtoPShare and baseline methods with respect to decreasing number of prototypes. We compare the accuracy (higher is better) for various pruning rates on CUB-200-2011 and Stanford Cars datasets. As observed, our ProtoPShare achieves higher accuracy than the variations of our method (data-independent and random), ProtoPNet with the original pruning, and ProtoPNet (shared) trained with shared prototypes. Moreover, ProtoPShare maintains high accuracy even for only 25% of the initial prototypes. Notice that the accuracy for the initial pruning steps is zoomed to increase plot clarity, and that we stop pruning around 300 remaining prototypes due to the model collapse.

Then, after the prototypes' merge, the output logit for class c can decrease at most by $\Delta_{\max}^c$, and the output logit for the other classes $k \neq c$ can increase at most by $\Delta_{\max}^k$, where:

$$\Delta_{\max}^k := |P_k \setminus P'_k| \log((1 + \delta)(2 - \delta)) \quad \text{for } k \in \{1, \dots, K\}.$$

Hence, if the output logits between the top-2 classes are at least $\Delta_{\max}^c + \max_{k \neq c} (\Delta_{\max}^k)$ apart, then the merge of prototypes does not change the prediction of x .

4 EXPERIMENTS

In this section, we analyze the accuracy and robustness of ProtoPShare in various scenarios of the merge-pruning, and we explain our architectural choices by comparing it to other methods.

We train ProtoPShare to classify 200 bird species from the CUB-200-2011 dataset [37] and 196 car models from the Stanford Cars dataset [21]. As the convolutional layers f , we take the convolutional blocks of ResNet-34, ResNet-152 [15], DenseNet-121, and DenseNet-161 [18] pretrained on ImageNet [6]. We set the number of prototypes per class to 10, which results in 2000 prototypes for birds and 1960 prototypes for cars after the first phase of ProtoPShare. Moreover, in the second phase, we assume finetuning with 25 iterations after each pruning step.

How many prototypes are required? In Figure 6, we compare the accuracy of ProtoPShare with ProtoPNet and variations of our method (data-independent and random) for various pruning rates. Data-independent uses inverse Euclidean norm as the similarity measure instead of similarity defined in (2), while random corresponds to random joining. The results for ProtoPShare and its variations were obtained from the successive steps of pruning, while the results for ProtoPNet were obtained for different values of κ and τ from Appendix S8 in [4]. One can observe that ProtoPShare

Figure 7: The accuracy of ProtoPShare for different percentages of prototypes merged per pruning step with respect to decreasing number of prototypes. Comparing the accuracy (higher is better) on the CUB-200-2011 dataset, we observe that the drop in accuracy is lower for a smaller number of prototypes (5%) merged per step. It is expected because small modifications in the set of prototypes make finetuning more effective than in the case of broader changes.

Table 1: ProtoPShare and baseline method results for the smallest number of prototypes that preserve baseline accuracy. Results show the superiority of our method achieved for various architectures trained on both datasets, which in some cases (e.g. ResNet34 trained on CUB-200-2011) obtains almost 10% higher accuracy. Additional results correspond to different finetuning strategies used after ProtoPShare pruning steps, including optimizing only the last layer (h), optimizing the prototypes and last layers (P and h), and optimizing all the layers (f , P , and h). Note that RN and DN correspond to ResNet and DenseNet architectures, respectively.

Model	Finetuning after pruning step			$ P $ before pruning	RN34	RN152	DN121	DN161	$ P $ before pruning	RN34	DN121
	f	P	h		$ P $ in the final model				$ P $ in the final model		
				400	1000	1000	600	480	980		
				CUB-200-2011				Stanford Cars			
ProtoPNet (shared)	no pruning				0.6833	0.6938	0.6985	0.6823		0.8231	0.8264
ProtoPNet			✓	2000	0.5751	0.7207	0.6972	0.7509	1960	0.8428	0.8031
ProtoPShare (ours)	✓	✓	✓	2000	0.6355	0.6821	0.6612	0.6526	1960	0.7864	0.7767
		✓	✓	2000	0.6591	0.7161	0.6836	0.7047	1960	0.7902	0.7903
			✓	2000	0.7472	0.7361	0.7472	0.7645	1960	0.8638	0.8481

achieves higher accuracy for all pruning rates and works reasonably well even for only 25% of the initial prototypes in case of ResNet34. Moreover, data-independent obtains similarly good results at the initial steps of prunings, but its accuracy drops more rapidly with a higher pruning rate. At the same time, the results of ProtoPNet, either with the original pruning or with shared prototypes, are always worse than ProtoPShare. Moreover, results for those two ProtoPNet variations are between results for data-independent and random models. Finally, for each model, we observe a critical step of pruning with a significant accuracy drop. Hence, we suggest monitoring the accuracy of the validation set when applying ProtoPShare in specific domains.

Accuracy vs. step size. In Figure 7, we present how the ProtoPShare behaves for different percentages of prototypes merged per pruning step ($\zeta = 5\%$, 10% , or 15%). One can observe that the accuracy is higher for smaller ζ . It is expected because small modifications in the set of prototypes make finetuning with 25 iterations more effective than in the case of broader changes. Notice that we have not tested $\zeta < 5\%$, which would most probably further increase the accuracy at the cost of extensive computations.

Why two training phases? To demonstrate the need for two training phases in ProtoPShare, we compare it to ProtoPNet trained with shared prototypes. In this setting, ProtoPShare is first trained with plenty of prototypes (e.g. 2000 for the CUB-200-2011) and then pruned to a small number of prototypes (e.g. 400 for ResNet34). Simultaneously, we train ProtoPNet with a small number of prototypes from the beginning (e.g. 400 for ResNet34) that are shared between classes. The results presented in Figure 6 and Table 1 confirm that ProtoPShare achieves better accuracy for both datasets. The biggest gain in accuracy, between 5% to 10%, is obtained for the CUB-200-2011. However, the trend is similar in the case of the Stanford Cars. Additionally, we compare the effectiveness of different finetuning strategies in Table 1. Based on the results, we conclude that the finetuning only the last layer is the best strategy. Notice that the results in Table 1 are obtained for the smallest number of

Figure 8: The difference in the accuracy between ProtoPShare with 600 prototypes and ProtoPNet with 1400 prototypes. The results obtained for DenseNet161 and various image perturbations suggest that ProtoPShare preserves robustness to the image perturbations even though it uses a smaller number of prototypes.

prototypes that preserve baseline accuracy, while in Figure 6 we plot the complete results.

Resistance to perturbations. In Figure 8 we present the difference in accuracy between ProtoPShare trained for 2000 and then pruned to 600 prototypes with ProtoPNet trained for 2000 and then pruned to 1400 prototypes (using $\kappa = 6$ and $\tau = 5$ from Appendix S8 in [4]) for various image perturbations. We incorporate different perturbations and magnitudes that modify brightness, contrast, saturation, hue, and perspective using *Torchvision ColorJitter* and *Perspective* transformations with probability of perturbation equal 1 and perturbation values from range $[0; 0.5]$ for hue and $[0; 1]$ for the others. As presented, the ProtoPShare with smaller number of prototypes performs on par to the ProtoPNet with a larger number of prototypes. Therefore, we conclude that ProtoPShare preserves robustness to the image perturbations even though it uses a smaller number of prototypes.

Figure 9: Eight prototypes from eight classes merged into one prototype by our ProtoPShare method. One can see that all prototypes correspond to the prototypical part of a wing with light smudge.

Discussion. We confirmed that ProtoPShare achieves better accuracy than other methods for almost all pruning rates. We demonstrated the need for small-steps pruning with finetuning only the last fully connected layer instead of the broader optimization. Additionally, we explained the need for two phases (training and pruning) and verified that the smaller number of prototypes does not negatively influence the model’s robustness.

5 INTERPRETABILITY OF PROTOPHARE

In the section, we focus on the interpretability of ProtoPShare using qualitative results and user study. We explain why the merge of prototypes is better than the original pruning from [4], illustrate how our model can discover the inter-class similarity, and demonstrate the superiority of data-dependent similarity over its data-independent counterpart.

Why merge-pruning instead of pruning? We decided to introduce our merge-pruning algorithm after a detailed analysis of the prototypes pruned by ProtoPNet and the conclusion that around 30% of them represent significant prototypical parts instead of a background. This observation is somehow consistent with Section 3.2 in [4], which states that “the prototypes whose nearest training patches have mixed class identities *usually* correspond to background patches, and they can be automatically pruned from the model.” Nevertheless, we were surprised by the high percentage of meaningful prototypes that are pruned. To illustrate the problem, we present some of them in Figure 10. In our opinion, it negatively influences the model accuracy (what we show in Section 4) and the model explanations, as some of the important prototypes can be removed only because of its similarity to the other class. That is why we provide the merge-pruning that can join many semantically similar prototypes from different classes without significant decrease in accuracy. As an example, in Figure 9, we present eight prototypes from eight classes merged into one prototype corresponding to a wing with light smudge.

Inter-class similarity discovery. The positive property of ProtoPShare is its ability to discover the similarity between classes based on the prototypes they share. Such similarity can be represented, i.e., by the graph that we present in Figure 3 where each node corresponds to a class and the strength of an edge between two nodes corresponds to the number of shared prototypes. Such analysis has many applications, like finding similar prototypical parts between two classes or clustering them into groups. Moreover, such a graph can be the starting point for more advanced visualizations.

Figure 10: Example prototypes (one per row) pruned by ProtoPNet that do not represent the background. The first column corresponds to the image with the prototypical part (in a yellow bounding box). The second column shows the prototype’s activation map for this image obtained as described in [4]. The remaining five columns present images’ parts (in yellow bounding boxes) whose representations are closest to the considered prototype. We observe that none of the presented prototypes represent the background what illustrates that ProtoPNet removes many meaningful prototypes.

Why data-dependent similarity? To explain the superiority of our data-dependent over the data-independent similarity (an inverse of Euclidean distance), in Figure 2, we present the pairs of prototypes close according to the former, but distant according to the later. More examples are presented in Figure 11. It can be noticed that the prototypes of one pair are semantically similar, even though they sometimes differ in colors’ distribution. In our opinion, the ability to find such pairs of similar prototypes is the main advantage of our data-dependent similarity. This ability is extremely important after the merge-pruning step because, as presented in Figure 12, ProtoPShare representations of the image’s parts more often activate the prototypes assigned to their true class, which indirectly

(a) Pairs of semantically similar prototypes distant in the representation space but close according to our data-dependent similarity.

(b) Pairs of prototypes close in the representation space.

Figure 11: Similar prototypes according to data-dependent and data-independent metrics. Notice that each prototype is represented by one row of 3 closest images' parts (marked with yellow bounding-boxes), so each pair corresponds to 6 pictures.

results in higher accuracy and robustness. Notice that, to generate Figure 12, we create a dataset with 5 parts from each testing image with the highest prototype activation, and then analyze which of those activations correspond to a true class. We conclude that only ProtoPShare constantly increases this number, which means that it uses model capacity more effectively.

User study. To further verify the superiority of our data-dependent pruning, we conducted user study where we presented two pairs of prototypes to the users. One of the pair contains prototypes most similar according to our data-dependent similarity, while the other contains prototypes closest in the representation space. The pairs were presented at once but in random order to prevent the order bias, and the users were asked to indicate a more consistent pair of prototypes (with an additional option "Impossible to decide").

Figure 12: Percent of patches correctly assigned to their classes (higher is better) in the successive steps of the pruning. The ProtoPShare pruning method constantly increases this number, which means that it uses the model capacity more effectively than the original ProtoPNet pruning.

The study was conducted on 57 users asked 5 times for a different randomly chosen set of prototypes, resulting in 285 answers altogether. Among all the answers, 195 favored our approach, 61 preferred data-independent similarity, and in 29 cases users could not decide which similarity is better. Therefore, we conclude that our data-dependent similarity is considered as more consistent than its data-independent counterpart.

Discussion. In this section, we explained why prototype merge is preferred over background pruning and why it is crucial to use data-dependent instead of the data-independent similarity. Moreover, we showed how ProtoPShare can be used for similarity discovery.

6 CONCLUSIONS

We presented ProtoPShare, a self-explained method that incorporates the paradigm of prototypical parts to explain its predictions. The method extends the existing approaches because it can share the prototypes between classes, reducing their number up to three times and still preserving baseline accuracy. To efficiently share the prototypes, we introduced our data-dependent pruning that merges prototypes with similar semantics. As a result, we increased the model's interpretability and enabled similarity discovery while maintaining high accuracy, as we showed through theoretical results and many experiments, including user study.

ACKNOWLEDGMENTS

Dawid Rymarczyk part in this research was funded by the Priority Research Area Digiworld under the program Excellence Initiative – Research University at the Jagiellonian University in Kraków.

Łukasz Struski, Jacek Tabor and Bartosz Zieliński part was founded by the POIR.04.04.00-00-14DE/18-00 project carried out within the Team-Net programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

REFERENCES

- [1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *NIPS*. 9505–9515.
- [2] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. 2019. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012* (2019).
- [3] Wieland Brendel and Matthias Bethge. 2019. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *ICLR* (2019).
- [4] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: deep learning for interpretable image recognition. In *NIPS*. 8930–8941.
- [5] Zhi Chen, Yijie Bei, and Cynthia Rudin. 2020. Concept Whitening for Interpretable Image Recognition. *arXiv:2002.01650* (2020).
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. Ieee, 248–255.
- [7] Finale Doshi-Velez and Been Kim. 2017. A roadmap for a rigorous science of interpretability. *arXiv:1702.08608* 2 (2017).
- [8] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. 2019. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*. 2950–2958.
- [9] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*. 3429–3437.
- [10] Jianlong Fu, Heliang Zheng, and Tao Mei. 2017. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*. 4438–4446.
- [11] Alan H Gee, Diego Garcia-Olano, Joydeep Ghosh, and David Paydarfar. 2019. Explaining deep classification of time-series data with learned prototypes. *arXiv:1904.08935* (2019).
- [12] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. 2019. Towards automatic concept-based explanations. In *NIPS*. 9277–9286.
- [13] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. 2019. Black Box Explanation by Learning Image Exemplars in the Latent Feature Space. In *ECML PKDD*. Springer, 189–205.
- [14] Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. 2019. Interpretable Image Recognition with Hierarchical Prototypes. In *AAAI*. 32–40.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
- [16] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*. 4340–4349.
- [17] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*. 1389–1397.
- [18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*. 4700–4708.
- [19] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*. PMLR, 2668–2677.
- [20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
- [21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D Object Representations for Fine-Grained Categorization. In *3dRR-13*. Sydney, Australia.
- [22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. In *ICLR*.
- [23] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *AAAI* (2018).
- [24] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. 2019. Towards optimal structured cnn pruning via generative adversarial learning. In *CVPR*. 2790–2799.
- [25] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the value of network pruning. *ICLR* (2019).
- [26] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*. 5058–5066.
- [27] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. 2019. Interpretable and steerable sequence learning via prototypes. In *KDD*. 903–913.
- [28] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. 2019. Pruning convolutional neural networks for resource efficient inference. In *ICLR*.
- [29] Esther Puyol-Antón, Chen Chen, James R Clough, Bram Ruijsink, Baldeep S Sidhu, Justin Gould, Bradley Porter, Marc Elliott, Vishal Mehta, Daniel Rueckert, et al. 2020. Interpretable Deep Models for Cardiac Resynchronisation Therapy Response Prediction. In *MICCAI*. Springer, 284–293.
- [30] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. 2020. There and Back Again: Revisiting Backpropagation Saliency Methods. In *CVPR*. 8839–8848.
- [31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *KDD*. 1135–1144.
- [32] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. 2012. One-shot learning with a hierarchical nonparametric bayesian model. In *Proceedings of ICM Workshop on Unsupervised and Transfer Learning*. 195–206.
- [33] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*. 618–626.
- [34] Ramprasaath R Selvaraju, Stefan Lee, Yilin Shen, Hongxia Jin, Shalini Ghosh, Larry Heck, Dhruv Batra, and Devi Parikh. 2019. Taking a hint: Leveraging explanations to make vision and language models more grounded. In *ICCV*. 2591–2600.
- [35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034* (2013).
- [36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. *ICML* (2017), 3319–3328.
- [37] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. 2011. The caltech-ucsd birds-200-2011 dataset. (2011).
- [38] Dong Wang, Lei Zhou, Xueni Zhang, Xiao Bai, and Jun Zhou. 2018. Exploring linear relationship in feature map subspace for convnets compression. In *WACV*.
- [39] Tong Wang. 2019. Gaining free or low-cost interpretability with interpretable partial substitute. In *ICML*. PMLR, 6505–6514.
- [40] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. 2015. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*. 842–850.
- [41] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. 2018. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*.
- [42] Chih-Kuan Yeh, Been Kim, Serkan O Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2019. On Completeness-aware Concept-Based Explanations in Deep Neural Networks. *arXiv:1910.07969* (2019).
- [43] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *CVPR*. 9194–9203.
- [44] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. 2017. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*. 5209–5217.
- [45] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Interpretable basis decomposition for visual explanation. In *ECCV*. 119–134.
- [46] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. 2018. Discrimination-aware channel pruning for deep neural networks. In *NIPS*. 875–886.

A PROOF FOR THEOREM 1

In this section, we present proof of Theorem 1 that examines the changes in the network's predictions after the prototypes' merge.

PROOF OF THEOREM 1. For any class k , let

$$L_k(x, P) = \sum_{p \in P} w_h(p) \cdot \log \left(\frac{\|z_p - p\|^2 + 1}{\|z_p - p\|^2 + \varepsilon} \right)$$

denotes its output logit for an input image $x \in X$. From assumption (A5), before the merge, we have

$$L_k(x, P_k) = \sum_{p \in P_k} \log \left(\frac{\|z_p - p\|^2 + 1}{\|z_p - p\|^2 + \varepsilon} \right),$$

and after merge

$$L_k(x, P'_k \cup S_k) = \sum_{p \in P'_k \cup S_k} \log \left(\frac{\|z_p - p\|^2 + 1}{\|z_p - p\|^2 + \varepsilon} \right).$$

If none of the prototypes from class k is merged into the other prototype (i.e., $S_k = \emptyset$), then there are no changes at the output (i.e., $L_k(x, P'_k \cup S_k) - L_k(x, P_k) = 0$). In the opposite case

$$\begin{aligned} L_k(x, P'_k \cup S_k) - L_k(x, P_k) &= \sum_{p \in S_k} \log \left(\frac{\|z_p - p\|^2 + 1}{\|z_p - p\|^2 + \varepsilon} \right) - \\ &\sum_{p \in P_k \setminus P'_k} \log \left(\frac{\|z_p - p\|^2 + 1}{\|z_p - p\|^2 + \varepsilon} \right) = \sum_{i=1}^{|P_k \setminus P'_k|} \log \left(\frac{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \cdot \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + \varepsilon} \right). \end{aligned}$$

For each class $k \in \{1..K\}$ and its prototypes $i \in \{1..|P_k \setminus P'_k|\}$, let

$$\vartheta_i := \frac{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \cdot \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + \varepsilon}.$$

To obtain the lower bound of ϑ_i for $k = c$ (correct class of x), we use the second inequality in (A4b), receiving

$$\frac{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \geq \frac{1}{\|z_{p_i} - p_i\|^2 + 1} \stackrel{\text{by (A4b)}}{\geq} \frac{1}{2 - \delta}.$$

By the assumption (A3) and the triangle inequality we obtain $\|z_{\tilde{p}_i} - \tilde{p}_i\| \leq \|z_{p_i} - \tilde{p}_i\| \leq \|z_{p_i} - p_i\| + \|\tilde{p}_i - p_i\|$. From the assumption (A4b) we obtain $\|\tilde{p}_i - p_i\| \leq (\sqrt{1 + \delta} - 1)\|z_{p_i} - p_i\|$, what implicate $\|\tilde{p}_i - p_i\| + \|z_{p_i} - p_i\| \leq \sqrt{1 + \delta}\|z_{p_i} - p_i\|$. Hence

$$\frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + \varepsilon} \geq \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{(\|z_{p_i} - p_i\| + \|\tilde{p}_i - p_i\|)^2 + \varepsilon} \geq \frac{1}{1 + \delta}.$$

Combining the above inequalities, we have

$$\vartheta_i = \frac{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \cdot \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + \varepsilon} \geq \frac{1}{(1 + \delta)(2 - \delta)}.$$

It means that the output logit change of class c as a result of the prototype merge satisfies

$$L_c(x, P'_c \cup S_c) - L_c(x, P_c) = \sum_{i=1}^{|P_c \setminus P'_c|} \log \vartheta_i \geq -|P_c \setminus P'_c| \log((1 + \delta)(2 - \delta))$$

or equivalently, $L_c(x, P_c) - L_c(x, P'_c \cup S_c) \leq |P_c \setminus P'_c| \log((1 + \delta)(2 - \delta))$. Hence, the worst decrease of the class c output logit as a result of prototype merge is $\Delta_{\max}^c$.

To obtain the lower bound of ϑ_i for $k \neq c$ (incorrect class of x), we use the triangle inequality together with assumptions (A3) and (A4a), receiving

$$\begin{aligned} \|z_{\tilde{p}_i} - \tilde{p}_i\|^2 &\stackrel{\text{by (A3)}}{\leq} \|z_{p_i} - \tilde{p}_i\|^2 \leq (\|z_{p_i} - p_i\| + \|\tilde{p}_i - p_i\|)^2 \\ &\stackrel{\text{by (A4a)}}{\leq} (\|z_{p_i} - p_i\| + (\sqrt{1 + \delta} - 1)\|z_{p_i} - p_i\|)^2 = (1 + \delta)\|z_{p_i} - p_i\|^2. \end{aligned}$$

Hence

$$\frac{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \leq \frac{(1 + \delta)\|z_{p_i} - p_i\|^2 + 1}{\|z_{p_i} - p_i\|^2 + 1} \leq 1 + \delta. \quad (3)$$

To derive an upper bound for the second part of ϑ_i , we use the assumption (A3) and (A4a)

$$\begin{aligned} \|p_i - \tilde{p}_i\| &\stackrel{\text{by (A4a)}}{\leq} \theta \|z_{p_i} - p_i\| - \sqrt{\varepsilon} \stackrel{\text{by (A3)}}{\leq} \theta \|z_{\tilde{p}_i} - p_i\| - \sqrt{\varepsilon} \\ &\stackrel{\text{by (A4a)}}{\leq} \left(1 - \frac{1}{\sqrt{2 - \delta}}\right) \|z_{\tilde{p}_i} - p_i\| - \sqrt{\varepsilon}. \end{aligned}$$

From which we know that $\|z_{\tilde{p}_i} - p_i\| - \|\tilde{p}_i - p_i\| > 0$. Again, we use the triangle inequality $\|z_{\tilde{p}_i} - p_i\| \leq \|z_{\tilde{p}_i} - \tilde{p}_i\| + \|\tilde{p}_i - p_i\|$, what implicate $\|z_{\tilde{p}_i} - \tilde{p}_i\| \geq \|z_{\tilde{p}_i} - p_i\| - \|\tilde{p}_i - p_i\|$. By reusing assumptions (A3) and (A4a), we have

$$\begin{aligned} \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\sqrt{2 - \delta}} &\leq \frac{1}{\sqrt{2 - \delta}} \|z_{p_i} - p_i\| + \sqrt{\varepsilon} \stackrel{\text{by (A4a)}}{\leq} \|z_{p_i} - p_i\| - \|\tilde{p}_i - p_i\| \\ &\stackrel{\text{by (A3)}}{\leq} \|z_{\tilde{p}_i} - p_i\| - \|\tilde{p}_i - p_i\|. \end{aligned}$$

Thanks to the above, we get

$$\frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{\|z_{\tilde{p}_i} - \tilde{p}_i\|^2 + \varepsilon} \leq \frac{\|z_{p_i} - p_i\|^2 + \varepsilon}{(\|z_{\tilde{p}_i} - p_i\| - \|\tilde{p}_i - p_i\|)^2 + \varepsilon} \leq \left(\frac{\|z_{p_i} - p_i\| + \varepsilon}{\|z_{\tilde{p}_i} - p_i\| - \|\tilde{p}_i - p_i\|} \right)^2 \leq 2 - \delta. \quad (4)$$

By inequalities (3), (4) get that the output logit change of class k as a result of the prototype merge satisfies

$$L_k(x, P'_k \cup S_k) - L_k(x, P_k) = \sum_{i=1}^{|P_k \setminus P'_k|} \log \vartheta_i \leq |P_k \setminus P'_k| \log((1 + \delta)(2 - \delta)).$$

Hence, the worst increase of the class k output logit as a result of prototype merge is $\Delta_{\max}^k$.

To prove the last thesis, let us assume that the output logit $L_c(x, P_k)$ of the correct class c before prototype merge is at least $\Delta_{\max}^c + \max_{k \neq c} (\Delta_{\max}^k)$ higher than the output logit $L_k(x, P_k)$ of any other class $k \neq c$, i.e.,

$$L_c(x, P_c) \geq L_k(x, P_k) + \Delta_{\max}^c + \max_{k \neq c} (\Delta_{\max}^k) \quad \text{for } k \neq c. \quad (5)$$

Since the output logit of the correct class c satisfies

$$L_c(x, P'_c \cup S_c) \geq L_c(x, P_c) - \Delta_{\max}^c \quad (6)$$

and the output logit of any class $k \neq c$ satisfies

$$L_k(x, P'_k \cup S_k) \leq L_k(x, P_k) + \Delta_{\max}^k, \quad (7)$$

for any $k \neq c$ we have $L_c(x, P'_c \cup S_c) \geq L_c(x, P_c) - \Delta_{\max}^c \geq L_k(x, P_k) + \max_{k \neq c} (\Delta_{\max}^k) \geq L_k(x, P'_k \cup S_k)$ (inequalities (6), (5), (7) give the individual inequalities, respectively). Hence, the input image x will still be correctly classified as class c after prototype merge. $\square$

B TRAINING DETAILS

We decided to omit the training details in the paper for its clarity. However, to make the paper reproducible, we provide them in this section.

First of all, we perform exhaustive offline data augmentation using rotation, skewing, flipping, and shearing with the probability of 0.5 for each operation. When it comes to the architecture, f is followed by two additional 1×1 convolutional layers before the prototypes' layer, and we use ReLU as the activation function for all convolutional layers except the last one (with the sigmoid function). The training bases on the images cropped (using the bounding-boxes provided with the datasets) and resized to 224×224 pixels. As a result, we obtain a convolutional feature map of size $7 \times 7 \times 256$. Hence, prototypes have size $1 \times 1 \times 256$. During training, we use Adam optimizer [20] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, batch size 80, learning rates 10^{-4} for f and h , and $3 \cdot 10^{-3}$ for additional 1×1 convolutions and prototypes' layer. For the loss function defined in [4], we use weights $\lambda_1 = 0.8$, $\lambda_2 = 0.08$, and $\lambda_l = 10^{-4}$. Moreover, the learning rates for convolutional parts and prototype layer are multiplied by 0.1 every 5 epochs of the training.

C USER STUDY QUESTIONNAIRE

Which of the two sets presents a more consistent meaning? *

☐ Option 1
 ☐ Option 2
 ☐ Impossible to decide

Figure 13: Example question from our user study.

To further verify the superiority of our data-dependent pruning, we conduct a user study where we present five questions to the user. There were four versions of the questionnaire with different sets of questions. Each questionnaire had the following initial instruction: “In this form, you will see two sets of images with yellow bounding-boxes. All bounding-boxes of one set should present one semantic meaning that corresponds to a particular characteristic of a bird’s part (e.g. white head, dark neck, striped wing, or black legs). Please, decide which of the two sets presents more consistent bounding-boxes. This task can be difficult, as two sets you compare can have a different meaning. Nevertheless, try to decide which of them is more consistent, and if it is impossible, please choose the option Impossible to decide.” After the initial instruction, we asked five times, “Which of the two sets presents a more consistent meaning?”. An example question is presented in Figure 13. We present the detailed results of the study in the paper.