Sanity checks for patch visualisation in prototype-based image classification

Romain Xu-Darme[1,2], Georges Quénot[2], Zakaria Chihani[1], Marie-Christine Rousset[2]

[1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[2] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
{romain.xu-darme, zakaria.chihani(at)cea.fr}
{georges.quenot, marie-christine.rousset(at)imag.fr}

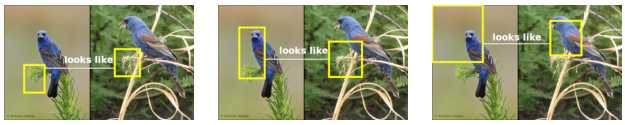
Abstract

In this work, we perform an analysis of the visualisation methods implemented in ProtoPNet and ProtoTree, two self-explaining visual classifiers based on prototypes. We show that such methods do not correctly identify the regions of interest inside of the images, and therefore do not reflect the model behaviour, which can create a false sense of bias in the model. We also demonstrate quantitatively that this issue can be mitigated by using other saliency methods that provide more faithful image patches.

1. Introduction

During the last decade, the field of Explainable AI (XAI) has gained wide-spread recognition among the scientific community [6, 16]. One major avenue of research in this field consists in developing architectures and training procedures such that the resulting model should be *self-explaining*. In computer vision, such architectures often use a case-based reasoning mechanism [7, 9, 17, 19, 20] where new instances of a problem are solved and explained using comparisons with visual examples (*prototypes*) extracted from the training data. In particular, ProtoPNet [7] and ProtoTree [17] have shown that self-explaining architectures can reach accuracy levels comparable to more opaque models on fine-grained recognition tasks [32, 33]. However, both models sometimes produce explanations using image patches that seem to be focused on elements unrelated to the object itself (Fig. 1a). Importantly, a prototype focusing on the background might indicate a *systemic* bias in the model and seriously hinder the user's trust in it. However, recent work [10] indicates that there exists in reality an imprecision in the patch visualisation method implemented in ProtoPNet. More generally, imprecise visualisation methods may suggest model bias where there is none, while sometimes hiding more systemic issues.

Figure 1. **Explanations of a ProtoTree** when using different visualisation methods. Due to the imprecision of upsampling when visualising both the prototype (right) and the part in the test image (left), the user might deduce that the model is comparing tree branches, when it is actually also taking the bird into account.



(a) Upsampling. (b) SmoothGrads. (c) PRP

Our contribution: In this work, we perform an analysis of the visualisation methods implemented in ProtoPNet and ProtoTree, answering the following research questions: do these architectures generate faithful explanations reflecting their decision-making process? do they produce decisions based on relevant parts of the image? We confirm quantitatively the results of [10] on ProtoPNet and show that ProtoTree also generates imprecise visual patches. Additionally, using the object segmentation provided in the CUB-200-2011 dataset, we propose a new relevance metric and show that in both architectures, such imprecise visualisations often create a false sense of bias that is largely mitigated by the use of more faithful methods. Finally, we discuss the implications of our findings to other prototype-based models sharing the same visualisation method.

2. Related work

Prototype-based classifiers first encode images into a high dimensional feature space (*latent space*) - generally using the first layers of a pre-trained convolutional neural network (CNN) as the backbone of the encoder. During training, these classifiers extract a set of reference feature vectors and their visual counterparts from the training set, called *part prototypes* (for simplicity, we simply use the term *prototypes*). Prototypes are either discriminative of a particular class [7, 9, 30] or shared among multiple classes [17, 19, 20]. During inference, the similarity between a given prototype

and the test image is computed using the L2-distance (or cosine distance [30]) between their respective latent representations. In the case of ProtoPNet [7], ProtoPShare [20], ProtoPool [19], Deformable ProtoPNet [9] and TesNet [30], all similarity scores are then processed through a fully connected layer to produce the prediction. In the case of ProtoTree [17], they are used to compute a path across a soft decision tree where each leaf corresponds to a categorical distribution among classes.

Note that although our study focuses on ProtoPNet and ProtoTree, all the aforementioned methods (ProtoPShare, ProtoPool, Deformable ProtoPNet, TesNet) share a common code base inherited from ProtoPNet and therefore are theoretically susceptible to the issue raised in this contribution.

Saliency methods aim at identifying the most important pixels of an image w.r.t. the output of a given neuron. Gradient-based approaches [24] compute the partial derivative of the target neuron output w.r.t. to each input pixel, with improvements such as Integrated Gradients [28], gradients $\odot$ input [22] (with $\odot$ denoting the element-wise multiplication), or SmoothGrads [26]. In particular, SmoothGrads "adds noise to remove noise" by averaging gradients over noisy copies of the input image. For ProtoPNet, [10] proposes a variant of LRP [5] called Prototype Relevance Propagation (PRP), implementing a dedicated rule to propagate relevance across the layer in charge of computing similarity scores. Since Integrated Gradients, LRP$-\epsilon$ and Deep-LIFT [22] are all equivalent to gradient $\odot$ input for CNNs based on ReLU activation [3, 23], in this work we choose to compare the original part visualisation generated by ProtoPNet and ProtoTree to visualisations generated using SmoothGrads $\odot$ input and PRP. Note that we exclude Guided-Backpropagation [27] and its application to GRAD-CAM [21] due to the results of [1] which raise some concerns regarding the faithfulness of the saliency maps generated by these methods.

Evaluation metrics: We first focus on the property known as *faithfulness* [2] which quantifies the adequacy between a saliency method and the model behaviour. Faithfulness can be evaluated by model parameter randomisation [1] or deletion/insertion methods, which monitor the evolution of a neuron's output when the most/least important pixels of the input image are removed incrementally [18, 29] or individually [2]. Deletion/insertion metrics check the ability of a saliency method to correctly *sort* pixels by importance w.r.t. to a given neuron's output, *i.e*. "removing" the most salient pixels identified by a faithful saliency method should result in a strong variation of the neuron's output.

Secondly, we wish to evaluate the *relevance* of prototype-based explanations. [15] applies perturbations (*e.g*. changes in colour or shape) on images and monitors the evolution of the similarity score for each prototype. Note that since these perturbations are applied on the *entire* image, this method does not require a precise location of the image patches.

3. Theoretical background

In this work, we consider a classification problem with a training set $X_{train} \subseteq \mathcal{X} \times \mathcal{Y}$. Let f be a fully convolutional neural network (fCNN) processing images in $\mathcal{X}$ and producing a D-dimensional latent representation of size (H, W). For $x \in \mathcal{X}$, we denote $f^{(h,w)}(x) \in \mathbb{R}^D$ the vector corresponding to the h-th row and w-th column of $f(x)$.

3.1. ProtoPNet and ProtoTree

Learning prototypes For $x \in \mathcal{X}$, ProtoPNet and ProtoTree compute their decision (classification) based on similarities between the latent representation $f(x)$ and a set of feature vectors $(r_1, \dots r_p)$ that are learned during training and act as reference points in the latent space. More precisely, the similarity between r_i and a particular vector $f^{(h,w)}(x)$ is defined as $s_i^{(h,w)}(x) = log\big((\|f^{(h,w)}(x) - r_i\|_2^2 + 1)/(\|f^{(h,w)}(x) - r_i\|_2^2 + \epsilon)\big)$ (ProtoPNet) or $s_i^{(h,w)}(x) = e^{-\|f^{(h,w)}(x) - r_i\|_2^2}$ (ProtoTree), where $\|.\|_2$ denotes the L2 distance. $s_i(x) \in \mathbb{R}^{H \times W}$ is called the *similarity map* between x and r_i. The model decision process d is a function of the aggregation of high similarity scores $s(x) = (\max(s_1(x)), \dots \max(s_p(x)))$: weighted sum for ProtoPNet, soft decision tree for ProtoTree. During training, the parameters of the feature extractor f, of the decision function[1] d, and the reference points r_i are jointly learned in order to minimize the cross-entropy loss between the prediction $d \circ s(x)$ and the label y, for $(x, y) \in X_{train}$. After training, the reference points r_i are "pushed" toward latent representations of parts of training images, in a process called *prototype projection*. More formally, a prototype P_i is a tuple $P_i = (p_i, h_i, w_i, r_i)$, computed using the training set X_{train}, where

$$\begin{cases} p_i, h_i, w_i = \underset{x \in X_{train}, h, w}{\arg\min} \|f^{(h,w)}(x) - r_i\|_2^2 \\ r_i \leftarrow f^{(h_i, w_i)}(p_i) \end{cases} \quad (1)$$

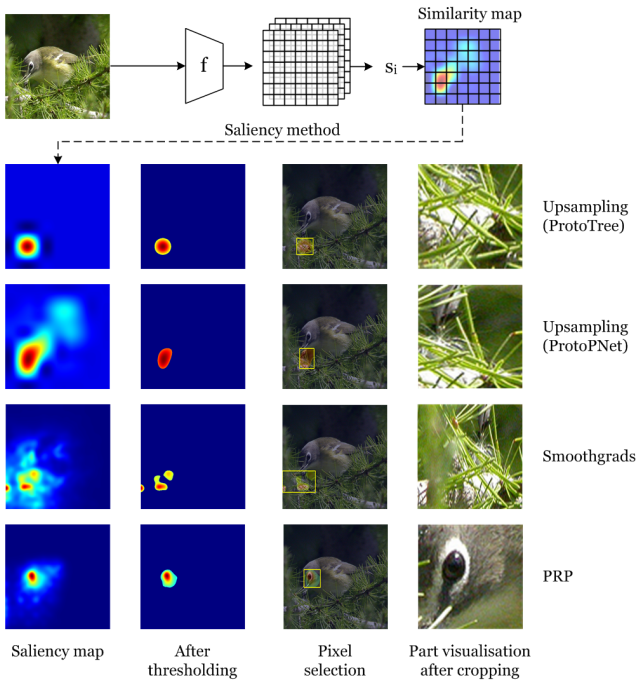
Thus, prototype projection moves each r_i to a nearby point that is, by construction, obtained from an image p_i.

From similarity map to part visualisation. Given an image x and a prototype P_i, ProtoPNet generates a visualisation of the most similar image patch in x by upsampling the similarity map $s_i(x) \in \mathbb{R}^{H \times W}$ to the size of x using cubic interpolation, then cropping the resulting saliency map to the 95th percentile. ProtoTree retains only the location of highest similarity in $s_i(x)$ before upsampling (setting all other locations to 0), as shown in Fig. 2. Note that the same

[1] The details of the decision function d are not relevant to our work, which focuses on the method used by ProtoPNet and ProtoTree to build a saliency map out of the similarity map $s_i(x)$.

method is also used to visualise the prototype itself by setting $x = p_i$. However, neither approach factors in the size

Figure 2. **From a similarity map to a part visualisation**. A saliency method generates a saliency map from the similarity map. Then, we retain only the most salient pixels through thresholding and crop the original image to produce a part visualisation.



of the receptive field[2] of each neuron in the last layers of a CNN. Indeed, the value $s_i^{(h,w)}(x)$ may actually depend on the entire image x rather than a localized region [4].
Finally, during inference, for $x \in \mathcal{X}$ and for each prototype P_i, both ProtoPNet and ProtoTree find the vector in $f(x)$ closest to r_i, corresponding to the highest score of the similarity map $s_i(x)$. If this score is above a given threshold, then they show side by side patches of images extracted from the prototype image p_i and x (*this looks like that*).

3.2. Improving part visualisation

In this paper, we wish to showcase the benefits of using other saliency methods for generating part visualisation. Similar to ProtoTree, for $x \in \mathcal{X}$ and a prototype P_i, we first find the coordinates of the highest similarity score

$$\begin{cases} \mathbf{H}_i(x), \mathbf{W}_i(x) = \arg\max_{h,w} s_i^{(h,w)}(x) \\ \mathbf{S}_i(x) = \max_{h,w} s_i^{(h,w)}(x) = s_i^{(\mathbf{H}_i(x),\mathbf{W}_i(x))}(x) \end{cases} \quad (2)$$

We generate saliency maps by applying SmoothGrads $\odot$ input or PRP on the output of the neuron $s_i^{(\mathbf{H}_i(x),\mathbf{W}_i(x))}$.

[2]It is actually computed in the code of ProtoPNet, but never put to use.

Then, we obtain a part visualisation by retaining only the 2% of most important pixels from x and cropping the image accordingly (Fig. 2). Again, the same method is applied in order to extract a part visualisation for each prototype.

3.3. Measuring faithfulness

Similar to [18], $\forall x \in \mathcal{X}$ and $\forall P_i$, we compare the faithfulness of the saliency methods by computing the Area Under the Deletion Curve (AUDC)

$$\int_0^{a_{max}} \tau(a)da = \int_0^{a_{max}} \frac{\mathbf{S}_i(x)}{s_i^{(\mathbf{H}_i(x),\mathbf{W}_i(x))}(x \odot m_a(x))} da \quad (3)$$

where $m_a(x)$ is the binary mask obtained after "deleting" (colouring in black) the $a\%$ most salient pixels in x. $\tau(a)$ measures the relative drop in the similarity scores between the original image x and a perturbed input $x \odot m_a(x)$ at the location $(\mathbf{H}_i(x), \mathbf{W}_i(x))$ inside the similarity map s_i. $\tau(a) \approx 1$ indicates that these pixels have no impact on the similarity score (unfaithful saliency method), while $\tau(a) \approx 0$ indicates that these pixels have a high impact on the similarity score, thus that the saliency method correctly identifies relevant pixels w.r.t. to the similarity score (faithful method). In order to reduce unexpected behaviours from the CNNs [11], we restrict ourselves to deleting a small portion of the original image (a_{max} =2%). Note that $\tau(a) < 0.2$ indicates that the deleted pixels amount to 80% of the similarity score, thus that the area of *the effective receptive field* of f w.r.t. to $\mathbf{S}_i(x)$ is probably close to a.

4. Experiments

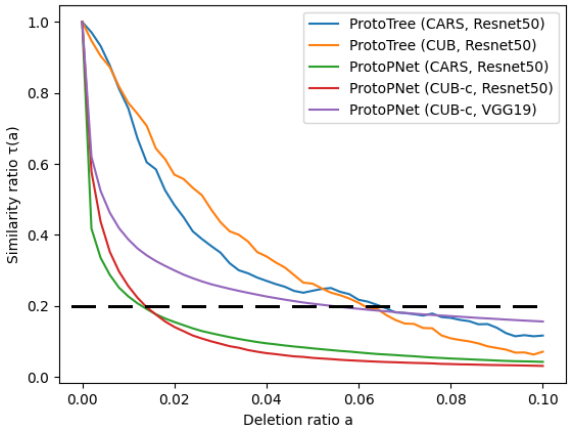
We perform our experiments on two popular datasets for fine-grained recognition: CUB-200-2011 [32] (CUB) and Stanford Cars [33] (CARS). For ProtoPNet, we use the images of the CUB dataset cropped to the object bounding box (we denote this dataset CUB-c) during training and inference. We primarily use a Resnet50 [12] backbone, pretrained on the iNaturalist [14] dataset (CUB) or the ImageNet [8] dataset (CARS), with images of size 224×224. To compare results with a different backbone, we also train a ProtoPNet on CUB-c using a VGG19 [25] network.
For PRP, we use the code kindly provided by the authors. For SmoothGrads, we use 10 noisy samples per image and a noise level of 0.2. In order to provide a clear baseline for the evaluation of the faithfulness and relevance of saliency maps, we also implemented a trivial method RandGrads returning a random saliency map drawn from a uniform distribution. For all three methods, we post-process the saliency map by averaging the gradients at each pixel location across the RGB channels, taking the absolute value (putting equal emphasis on positive and negative gradients), and applying a 5×5 Gaussian filter in order to avoid isolated gradients due to pooling layers inside of f.

Table 1. **Average AUDC** of prototypes (left value) and test patches (right value) generated by ProtoPNet and Prototree when using upsampling, SmoothGrads, PRP and RandGrads. For each architecture/dataset, values in **bold** indicates the most faithful saliency method.

Model	Backbone	Dataset	Method			
			Upsampling	PRP	SmoothGrads	RandGrads
ProtoPNet	VGG19	CUB-c	0.41 (± 0.12) / 0.74 (± 0.19)	0.39 (± 0.10) / 0.70 (± 0.19)	**0.37 (± 0.11) / 0.68 (± 0.20)**	0.77 (± 0.18) / 0.94 (± 0.12)
	ResNet50	CUB-c	0.39 (± 0.18) / 0.71 (± 0.28)	**0.31 (± 0.14) / 0.61 (± 0.28)**	0.37 (± 0.18) / 0.66 (± 0.29)	0.77 (± 0.23) / 0.93 (± 0.18)
		CARS	0.46 (± 0.19) / 0.88 (± 0.20)	**0.31 (± 0.11) / 0.68 (± 0.24)**	0.34 (± 0.13) / 0.71 (± 0.24)	0.60 (± 0.18) / 0.94 (± 0.14)
ProtoTree	ResNet50	CUB	0.98 (± 0.06) / 0.95 (± 0.16)	**0.78 (± 0.28) / 0.65 (± 0.34)**	0.91 (± 0.22) / 0.82 (± 0.28)	0.99 (± 0.08) / 0.98 (± 0.10)
		CARS	0.96 (± 0.13) / 0.91 (± 0.21)	**0.75 (± 0.25) / 0.71 (± 0.30)**	0.88 (± 0.21) / 0.84 (± 0.25)	0.99 (± 0.05) / 0.97 (± 0.12)

Faithfulness of patch visualisation. We measure the AUDC when visualising both prototypes and patches of images from the test set during inference: for ProtoTree, we apply the saliency method only when the prototype is considered "present" (right branch of each decision node); for ProtoPNet, we apply the saliency method on the 10 patches of the test image most similar to any prototype of the inferred class. The AUDC score is approximated by averaging the similarity ratio $\tau(a)$ for deletion areas between 0% and 2%, with an increment value of 0.1% (Table 1). In all cases, the upsampling method used in ProtoPNet and ProtoTree leads to a higher AUDC score than SmoothGrads or PRP. In the particular case of ProtoTree, it is only marginally better than our random baseline RandGrads. This confirms the imprecision pointed out in [10] and *extends the issue to ProtoTree*. Moreover, in general PRP seems to provide a more faithful saliency maps than SmoothGrads (lower AUDC). Note that AUDC scores are fairly similar across methods (except RandGrads) on the CUB-c dataset, which is probably due to the image cropping that increases the relative area of the bird inside of the image and decreases the probability to miss the important pixels. When extending the deletion area to 10% of the image, we note that on average, the drop in similarity ratio ($\tau(a) < 0.2$) occurs below 2% for ProtoPNet prototypes when using ResNet50, and around 7% for ProtoTree prototypes or ProtoPNet using VGG19 (Fig. 3). This indicates that the size of the effective receptive field depends not only on the choice of backbone (VGG, ResNet50), but also on the model (ProtoTree, ProtoPNet). For ProtoTree, this may be due to the fact that the decision tree shares prototypes among all classes and therefore does not focus on very small details, contrary to ProtoPNet[3]. Moreover, this clarifies the discrepancy in AUDC scores between ProtoPNet and ProtoTree visualisations.

Figure 3. **Average similarity ratio v. deletion area** when using PRP for visualising prototypes. Best viewed in colour.



Relevance of patch visualisation. To evaluate the relevance of image patches (prototypes or test patches), we measure their intersection with the object segmentation, assuming that such information is available (CUB dataset). If less than 5% of the image patch intersects the object segmentation, then we consider it irrelevant, as it mostly focuses on the background. This experiment shows that the imprecision of the visualisation method used in ProtoPNet and ProtoTree leads to a false sense of model bias (Table 2). In particular, when using the upsampling method, 35.4% of ProtoTree prototypes and 51.9% of the test image patches seem to be focusing on elements of the background rather than the bird. However, when using a more faithful saliency method such as PRP, we notice that only 0.5% of the prototypes or test image patches are actually irrelevant. Note that this gap between methods is again more limited for ProtoPNet with CUB-c, where the upsampling method is less

Table 2. **Measuring relevance**. Percentage of prototypes (left value) and test patches (right value) with less than 5% of intersection with the object on the CUB dataset.

Model	Backbone	Method		
		Upsampling	PRP	SmoothGrads
ProtoPNet	VGG19	15.4% / 23.3%	10.0% / 16.6%	11.3% / 19.9%
	ResNet50	2.1% / 8.8%	1.4% / 8.0%	0.9% / 6.1%
ProtoTree	ResNet50	35.4% / 51.9%	0.5% / 0.5%	8.7% / 14.5%

[3]This effect is also present when using SmoothGrads and seems uncorrelated to the depth of the prototype inside the tree

likely to "miss" the object entirely. Finally, we also notice that the percentage of irrelevant prototypes and test image patches is significantly more important when using VGG19 as a backbone, compared to using Resnet50, which raises the question of the sensitivity of prototype-based architectures to the underlying backbone architecture.

5. Discussion

Case-based reasoning architectures for image classification are undoubtedly a stepping stone towards more interpretable computer vision models, but they are highly dependent on the choice of a backbone and suffer from shortcomings that may hinder their widespread usage. Indeed, even when such models produce a correct decision for the right reasons, they may yet fail to explain this decision by incorrectly locating appropriate parts of the images. This issue is likely not restricted to ProtoPNet or ProtoTree, since ProtoPShare, ProtoPool, Deformable ProtoPNet and TesNet share a common code base inherited from ProtoPNet that includes the upsampling method for patch visualisation. On the contrary, more faithful saliency methods can help uncover biases [10] or - in our experiments - disprove *apparent* biases of the model. Crucially, proving that the model is indeed focusing on the object does not imply that the decision is based on understandable information. Indeed, the assumption that proximity in the latent space entails perceptual similarity in the visual space may not always hold [13, 15]. Thus, we argue that a decision-making process based on distance in the latent space is not sufficient to guarantee interpretability, *i.e*. that case-based reasoning architectures using CNNs for feature extraction are currently not *inherently* self-explainable. Consequently, such models should not be compared according to their classification accuracy, but rather using metrics for evaluating various properties of explanations in a systematic manner (*e.g*. *sparsity* [11]) and, in the case of prototype-based architectures, for quantifying visual similarity [15, 31].

Acknowledgments Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).
This work has been partially supported by MIAI@Grenoble Alpes, (ANR-19-P3IA-0003) and TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

Reproducibility Our code will be available shortly at the following url: https://github.com/romain-xu-darme/prototype_sanity_checks.git

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems 32*, page 11, 2018. 2

[2] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7786–7795, Red Hook, NY, USA, 2018. Curran Associates Inc. 2

[3] M. Ancona, E. Ceolini, A. C. Öztireli, and M. Gross. A unified view of gradient-based attribution methods for deep neural networks. In *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning*, 2017. 2

[4] André Araujo, Wade Norris, and Jack Sim. Computing receptive fields of convolutional neural networks. *Distill*, 2019. https://distill.pub/2019/computing-receptive-fields. 3

[5] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In Kuinam J. Kim and Nikolai Joukov, editors, *Information Science and Applications (ICISA) 2016*, pages 913–922, Singapore, 2016. Springer Singapore. 2

[6] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *ArXiv*, abs/2102.13076, 2021. 1

[7] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. *This* looks like *That*: Deep learning for interpretable image recognition. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, page 8930–8941, 2019. 1, 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[9] Jonathan Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10255–10265, 2021. 1, 2

[10] Srishti Gautam, Marina M.-C. Höhne, Stine Hansen, Robert Jenssen, and Michael Kampffmeyer. This looks more like that: Enhancing self-explaining models by prototypical relevance propagation. *Pattern Recognition*, page 109172, 2022. 1, 2, 4, 5

[11] Tristan Gomez, Thomas Fréour, and Harold Mouchère. Metrics for saliency map evaluation of deep learning explanation methods. In *International Conferences on Pattern Recognition and Artificial Intelligence*, 2022. 3, 5

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[13] Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *ICML*

2021 Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI, 2021. 5

[14] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist species classification and detection dataset. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2017. 3

[15] Meike Nauta, Annemarie Jutte, Jesper C. Provoost, and Christin Seifert. This looks like that, because ... explaining prototypes for interpretable image recognition. In *PKDD/ECML Workshops*, 2020. 2, 5

[16] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Comput. Surv.*, feb 2023. Just Accepted. 1

[17] Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14928–14938, 2021. 1, 2

[18] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018. 2, 3

[19] Dawid Rymarczyk, Lukasz Struski, Michal G'orszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieli'nski. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, 2021. 1, 2

[20] Dawid Rymarczyk, Lukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021. 1, 2

[21] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *ArXiv*, abs/1611.07450, 2016. 2

[22] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, 2017. 2

[23] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *ArXiv*, abs/1605.01713, 2016. 2

[24] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. 2

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015. 3

[26] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017. 2

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. 2

[28] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017. 2

[29] Richard J. Tomsett, Daniel Harborne, Supriyo Chakraborty, Prudhvi K. Gurram, and Alun David Preece. Sanity checks for saliency metrics. *ArXiv*, abs/1912.01451, 2019. 2

[30] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable image recognition by constructing transparent embedding space. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 875–884, 2021. 1, 2

[31] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004. 5

[32] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, 2010. 1, 3

[33] L. Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A Large-Scale Car Dataset for Fine-Grained Categorization and Verification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, 2015. 1, 3