Intrinsically Interpretable Image Recognition with Neural Prototype Trees

Meike Nauta[1] Ron van Bree[1] Christin Seifert[1,2]
[1] University of Twente, the Netherlands [2] University of Duisburg-Essen, Germany
m.nauta@utwente.nl, r.j.vanbree@student.utwente.nl, christin.seifert@uni-due.de

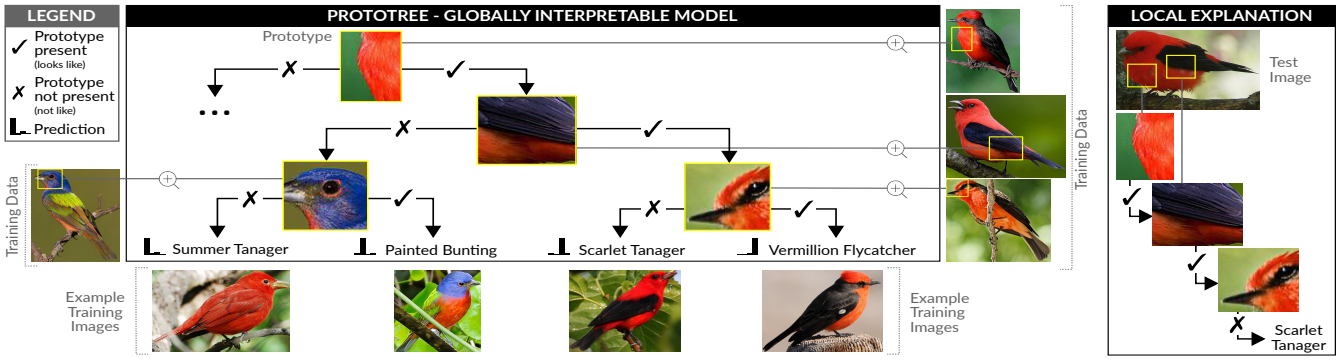

Figure 1: A ProtoTree is a globally interpretable model faithfully explaining its entire reasoning (left, partially shown). Additionally, the decision making process for a single prediction can be followed (right): the presence of a red chest and black wing, and the absence of a black stripe near the eye, identifies a Scarlet Tanager. A pruned ProtoTree learns roughly 200 prototypes for CUB (dataset with 200 bird species), making only 8 local decisions on average for one test image.

Abstract

Prototype-based methods use interpretable representations to address the black-box nature of deep learning models, in contrast to post-hoc explanation methods that only approximate such models. We propose the Neural Prototype Tree (ProtoTree), an intrinsically interpretable deep learning method for fine-grained image recognition. ProtoTree combines prototype learning with decision trees, and thus results in a globally interpretable model by design. Additionally, ProtoTree can locally explain a single prediction by outlining a decision path through the tree. Each node in our binary tree contains a trainable prototypical part. The presence or absence of this learned prototype in an image determines the routing through a node. Decision making is therefore similar to human reasoning: Does the bird have a red throat? And an elongated beak? Then it's a hummingbird! We tune the accuracy-interpretability trade-off using ensemble methods, pruning and binarizing. We apply pruning without sacrificing accuracy, resulting in a small tree with only 8 learned prototypes along a path to classify a bird from 200 species. An ensemble of 5 ProtoTrees achieves competitive accuracy on the CUB-200- 2011 and Stanford Cars data sets.

1. Introduction

There is an ongoing scientific dispute between simple, interpretable models and complex black boxes, such as Deep Neural Networks (DNNs). DNNs have achieved superior performance, especially in computer vision, but their complex architectures and high-dimensional feature spaces has led to an increasing demand for transparency, interpretability and explainability [1]. In contrast, decision trees are easy to understand and interpret [5, 8], because they transparently arrange decision rules in a hierarchical structure. Their predictive performance is however far from competitive for computer vision tasks. We address this so-called 'accuracy-interpretability trade-off' [1, 16] by combining the expressiveness of deep learning with the interpretability of decision trees.

We present the *Neural Prototype Tree*, ProtoTree in short, an intrinsically interpretable method for fine-grained image recognition. A ProtoTree has the representational power of a neural network, and contains a built-in binary decision tree structure, as shown in Fig. 1 (left). Each internal node in the tree contains a trainable *prototype*. Our prototypes are prototypical *parts* learned with backpropagation, as introduced in the Prototypical Part Network (ProtoPNet) [4] where a prototype is a trainable tensor that can be visualized as a patch of a training sample. The extent to which this prototype is present in an input image determines the routing of the image through the corresponding node. Leaves of the ProtoTree learn class distributions. The paths from root to leaves represent the learned classification rules.

To this end, a ProtoTree consists of a Convolutional Neural Network (CNN) followed by a binary tree structure and can be trained end-to-end with a standard cross-entropy loss function. We only require class labels and do not need any other annotations. To make the tree differentiable and back-propagation compatible, we utilize a *soft* decision tree, meaning that a sample is routed through both children, each with a certain weight.

We present a novel routing procedure based on the similarity between the latent image embedding and a prototype.

A ProtoTree approximates the accuracy of non-interpretable classifiers, while being *interpretable-by-design* and offering truthful global and local explanations. This way it provides a novel take on interpretable machine learning. In contrast to *post-hoc* explanations, which approximate a trained model or its output [18, 14], a ProtoTree is inherently interpretable since it directly incorporates interpretability in the structure of the predictive model [18]. A ProtoTree therefore faithfully shows its entire classification behaviour, independent of its input, providing a *global* explanation (Fig. 1). As a consequence, our compact tree enables a human to convey, or even print out, the *whole* model. In contrast to *local* explanations, which explain a single prediction and can be unstable and contradicting [3, 11], global explanations enable *simulatability* [16]. Additionally, our ProtoTree can produce *local* explanations by showing the routing of a specific input image through the tree (Fig. 1, right). Hence, ProtoTree allows retraceable decisions in a human-comprehensible number of steps. In case of a misclassification, the responsible node can be identified by tracking down the series of decisions, which eases error analysis.

Scientific Contributions

- An intrinsically interpretable neural prototype tree architecture for fine-grained image recognition.
- Outperforming ProtoPNet [4] while having roughly only 10% of the number of prototypes, included in a built-in hierarchical structure.
- An ensemble of 5 interpretable ProtoTrees achieves competitive performance on CUB-200-2011 [22] (CUB) and Stanford Cars [13].

2. Neural Prototype Tree

A Neural Prototype Tree (ProtoTree) hierarchically routes an image through a binary tree for interpretable image recognition. We consider a classification problem with training set $\boldsymbol{\mathcal{T}}$ containing N labelled images $\{(\boldsymbol{x}^{(1)}, y^{(1)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\} \in \mathcal{X} \times \mathcal{Y}$. Given an input image $\boldsymbol{x}$, a ProtoTree predicts the class probability distribution over K classes, denoted as $\hat{\boldsymbol{y}}$. We use $\boldsymbol{y}$ to denote the one-hot encoded ground-truth label y. A ProtoTree can also be trained with soft labels from a trained model for knowledge distillation, similar to [6].

A ProtoTree T is a combination of a convolutional neural network (CNN) f with a soft neural binary decision tree structure. As shown in Fig. 2, an input image is first forwarded through f. The resulting convolutional output $\boldsymbol{z} = f(\boldsymbol{x}; \boldsymbol{\omega})$ consists of D two-dimensional ($H \times W$) feature maps, where $\boldsymbol{\omega}$ denotes the trainable parameters of f. Secondly, the latent representation $\boldsymbol{z} \in \mathbb{R}^{H \times W \times D}$ serves as input for a binary tree. This tree consists of a set of internal nodes $\mathcal{N}$, a set of leaf nodes $\mathcal{L}$, and a set of edges $\mathcal{E}$. Each internal node $n \in \mathcal{N}$ has exactly two child nodes: $n.left$ connected by edge $e(n, n.left) \in \mathcal{E}$ and $n.right$ connected by $e(n, n.right) \in \mathcal{E}$. Each internal node $n \in \mathcal{N}$ corresponds to a trainable prototype $\boldsymbol{p}_n \in \boldsymbol{P}$. We follow the prototype definition of ProtoPNet [4] where each prototype is a trainable tensor of shape $H_1 \times W_1 \times D$ (with $H_1 \leq H$, $W_1 \leq W$, and in our implementation $H_1 = W_1 = 1$) such that the prototype's depth corresponds to the depth of the convolutional output $\boldsymbol{z}$.

We use a form of generalized convolution without bias [7], where each prototype $\boldsymbol{p}_n \in \boldsymbol{P}$ acts as a kernel by 'sliding' over $\boldsymbol{z}$ of shape $H \times W \times D$ and computes the Euclidean distance between $\boldsymbol{p}_n$ and its current receptive field $\tilde{\boldsymbol{z}}$ (called a *patch*). We apply a minimum pooling operation to select the patch in $\boldsymbol{z}$ of shape $H_1 \times W_1 \times D$ that is closest to prototype $\boldsymbol{p}_n$. The distance between the nearest latent patch $\tilde{\boldsymbol{z}}^*$ and prototype $\boldsymbol{p}_n$ determines to what extent the prototype is present *anywhere* in the input image, which influences the routing of $\boldsymbol{z}$ through corresponding node n. In contrast to traditional decision trees, where an internal node routes sample $\boldsymbol{z}$ either right or left, our node $n \in \mathcal{N}$ is *soft* and routes $\boldsymbol{z}$ to both children, each with a fuzzy weight within $[0, 1]$, giving it a probabilistic interpretation [6, 10, 12, 21]. We define the similarity between $\tilde{\boldsymbol{z}}^*$ and $\boldsymbol{p}_n$, and therefore the probability of routing sample $\boldsymbol{z}$ through the right edge as

$$p_{e(n,n.right)}(\boldsymbol{z}) = \exp(-||\tilde{\boldsymbol{z}}^* - \boldsymbol{p}_n||), \tag{1}$$

such that $p_{e(n,n.left)} = 1 - p_{e(n,n.right)}$. Thus, the similarity between prototype $\boldsymbol{p}_n$ and the nearest patch in the convolutional output, $\tilde{\boldsymbol{z}}^*$, determines to what extent $\boldsymbol{z}$ is routed to the right child of node n. Because of the soft routing, $\boldsymbol{z}$ is traversed through all edges and ends up in each leaf node $\ell \in \mathcal{L}$ with a certain probability. Path $\mathcal{P}_\ell$ denotes the sequence of edges from the root node to leaf ℓ. The probability of sample $\boldsymbol{z}$ arriving in leaf ℓ, denoted as $\boldsymbol{\pi}_\ell$, is the product of probabilities of the edges in path $\mathcal{P}_\ell$:

$$\boldsymbol{\pi}_\ell(\boldsymbol{z}) = \prod_{e \in \mathcal{P}_\ell} p_e(\boldsymbol{z}). \tag{2}$$

Each leaf node $\ell \in \mathcal{L}$ carries a trainable parameter $\boldsymbol{c}_\ell$, denoting the distribution in that leaf over the K classes that needs to be learned. The softmax function $\sigma(\boldsymbol{c}_\ell)$ normalizes $\boldsymbol{c}_\ell$ to get the class *probability* distribution of leaf ℓ. To obtain the final predicted class probability distribution $\hat{\boldsymbol{y}}$ for input image $\boldsymbol{x}$, latent representation $\boldsymbol{z} = f(\boldsymbol{x}|\boldsymbol{\omega})$ is traversed through all edges in T such that all leaves contribute to the final prediction $\hat{\boldsymbol{y}}$. An example prediction is shown on the right of Fig. 2. The contribution of leaf ℓ is weighted by path probability $\boldsymbol{\pi}_\ell$, such that:

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{\ell \in \mathcal{L}} \sigma(\boldsymbol{c}_\ell) \cdot \boldsymbol{\pi}_\ell(f(\boldsymbol{x}; \boldsymbol{\omega})). \tag{3}$$

3. Training and Visualization

Training a ProtoTree requires to learn the parameters $\boldsymbol{\omega}$ of a pre-trained CNN f for informative feature maps, the nodes' prototypes $\boldsymbol{P}$ for routing and the leaves' class distribution logits $\boldsymbol{c}$ for prediction. The number of prototypes to be learned, *i.e.* $|\boldsymbol{P}|$, depends on the tree size. A binary tree structure is initialized by defining a maximum height h, which creates 2^h leaves and $2^h - 1$ prototypes. It is sensible to set h such that the number

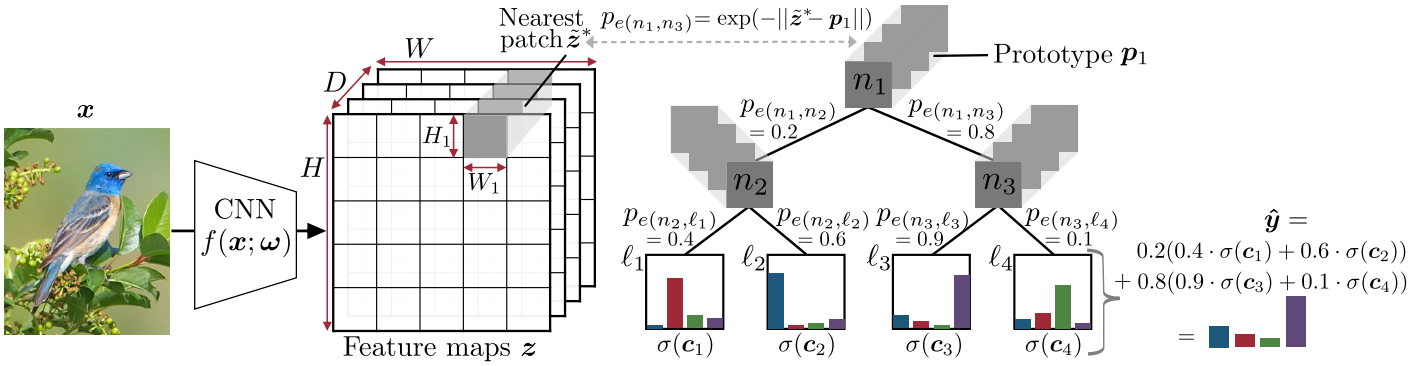

Figure 2: Decision making process of a ProtoTree to predict class probability distribution $\hat{\boldsymbol{y}}$ of input image $\boldsymbol{x}$. During training, prototypes $\boldsymbol{p}_n \in \boldsymbol{P}$, leaves' class distributions $\boldsymbol{c}$ and CNN parameters ω are learned. Probabilities p_e (shown with example values) depend on the similarity between a patch in the latent input image and a prototype.

of leaves is at least as large as the number of classes K. During training, prototypes in $\boldsymbol{P}$ are trainable tensors. Parameters $\boldsymbol{\omega}$ and $\boldsymbol{P}$ are simultaneously learned with back-propagation by minimizing the cross-entropy loss between the predicted class probability distribution $\hat{\boldsymbol{y}}$ and ground-truth $\boldsymbol{y}$. A derivative-free learning algorithm, adapted from Kontschieder *et al*. [12], learns the leaves' distributions $\boldsymbol{c}$.

After training, we analyse the learned class probability distributions in the leaves and prune leaves with nearly uniform distributions, *i.e*. little discriminative power. Specifically, we define a threshold τ and prune all leaves where $\max(\sigma(\boldsymbol{c}_\ell)) \leq \tau$, with τ being slightly greater than $1/K$ where K is the number of classes. If all leaves in a full subtree $T' \subset T$ are pruned, T' (and its prototypes) can be pruned. After pruning ProtoTree, learned latent prototypes are mapped to pixel space to enable interpretability. Similar to ProtoPNet [4], we replace each prototype $\boldsymbol{p}_n \in \boldsymbol{P}$ with its nearest latent patch present in the training data, $\tilde{\boldsymbol{z}}_n^*$. Since $\tilde{\boldsymbol{z}}_n^*$ is used for routing at test time, the visualized ProtoTree is a faithful model explanation. Lastly, since hard decision trees easier to interpret than soft trees [2], we can convert a trained soft ProtoTree to a hard tree at test time, by selecting the path to the leaf with the highest path probability, or by greedily traversing the tree.

4. Experiments and Results

We compare our ProtoTrees with ProtoPNet [4] (which uses a bag of class-specific prototypes) and state-of-the-art uninterpretable models, by evaluating on CUB-200-2011 [22] with 200 bird species (CUB) and Stanford Cars [13] with 196 car types (CARS). We implemented ProtoTree in PyTorch, and resized all images to 224×224 such that the resulting feature maps are 7×7. A ResNet50 [9] backbone is extended with a 1×1 convolutional layer to reduce the dimensionality of latent output $\boldsymbol{z}$ to D, the prototype depth. Based on cross-validation from $\{128, 256, 512\}$, we used D=256 for CUB and D=128 for CARS, such that a prototype is of size $1 \times 1 \times 256$ for CUB.

Data set	Method	Interpret.	Top-1 Accuracy	#Prototypes
CUB (224×224)	Triplet Model [15]	-	**87.5**	n.a.
	TranSlider [24]	-	85.8	n.a.
	TASN [23]	o	87.0	n.a.
	ProtoPNet [4]	+	79.2	2000
	ProtoTree h=9 (ours)	++	82.2±0.7	**202**
	ProtoPNet ens. (3) [4]	+	84.8	6000
	ProtoTree ens. (3)	+	86.6	605
	ProtoTree ens. (5)	+	**87.2**	1008
CARS (224×224)	RAU [17]	-	**93.8**	n.a.
	Triplet Model [15]	-	93.6	n.a.
	TASN [23]	o	93.8	n.a.
	ProtoPNet [4]	+	86.1	1960
	ProtoTree h=11 (ours)	++	86.6±0.2	**195**
	ProtoPNet ens. (3) [4]	+	91.4	5880
	ProtoTree ens. (3)	+	90.3	586
	ProtoTree ens. (5)	+	**91.5**	977

Table 1: Mean accuracy and standard deviation of our ProtoTree (5 runs) and ensemble with 3 or 5 ProtoTrees compared with self-reported accuracy of uninterpretable state-of-the-art (-), attention-based models (o) and interpretable ProtoPNet (+).

4.1. Accuracy and Interpretability

Table 1 shows that our ProtoTree outperforms ProtoPNet for both datasets. We also evaluated the accuracy of ProtoTree ensembles by averaging the predictions of 3 or 5 individual ProtoTrees, all trained on the same dataset. An ensemble of ProtoTrees outperforms a ProtoPNet ensemble, and approximates the accuracy of uninterpretable or attention-based methods, while providing intrinsically interpretable global and faithful local explanations.

Pruning. Most leaves learn either one class label, or an almost uniform distribution, as shown in Fig. 3 (top left) for CUB

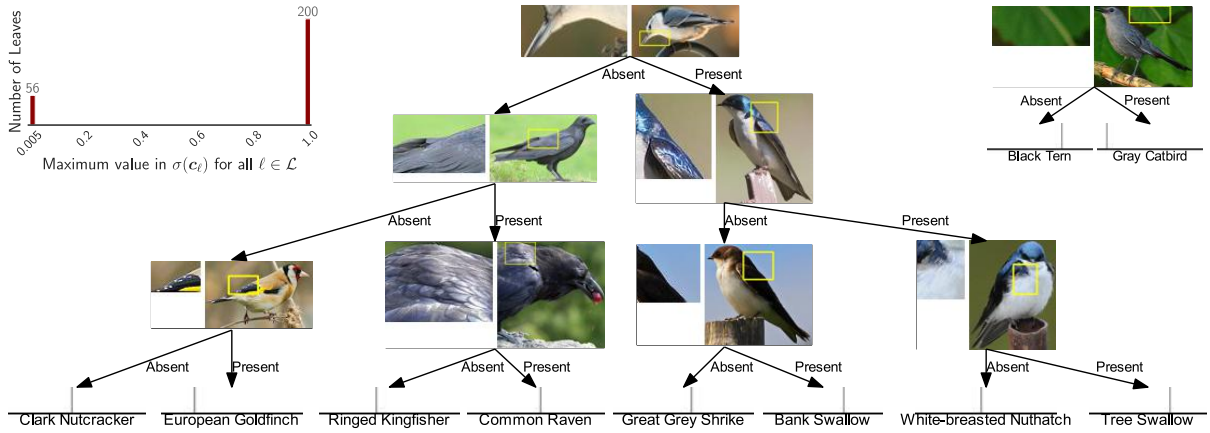

Figure 3: Subtree of an automatically visualized ProtoTree trained on CUB, h=8 (middle). Each internal node contains a prototype (left) and the training image from which it is extracted (right). A ProtoTree faithfully shows its reasoning and clusters similar classes (*e.g*. birds with a white chest). Top left: maximum values of all leaf distributions. Top right: ProtoTree reveals biases learned by the model: *e.g*. classifying a Gray Catbird based on the presence of a leaf. Best viewed in color.

with h=8. We set pruning threshold $\tau = 0.01$, such that we are left with leaves that can be interpreted (nearly) deterministically. Pruning drastically reduces the size of the tree (up to $> 90\%$), preserving roughly 1 prototype per class. In contrast, ProtoPNet [4] uses 10 prototypes per class (cf. Tab. 1), resulting in 2000 prototypes in total for CUB. Thus, a ProtoTree is almost 90% smaller and therefore easier to interpret. Even with an ensemble of ProtoTrees, the number of prototypes is still substantially smaller than ProtoPNet.

Deterministic reasoning. A ProtoTree can make deterministic predictions at test time to improve understandability. Selecting the leaf with the highest path probability leads to nearly the same accuracy, since the fidelity (*i.e*. fraction of test images for which the soft and hard strategy make the same classification [8]) is 0.999. The greedy strategy performs slightly worse, but still has a fidelity of 0.987. Results are similar for other datasets and tree heights, showing that a ProtoTree can be safely converted to a deterministic tree, such that a prediction can be explained by presenting one path in the tree. Our deterministic ProtoTree (h=9) reduces the number of decisions to follow to 9 prototypes at maximum. When using a more accurate ensemble of 5 deterministic ProtoTrees, a maximum of only 45 prototypes needs to be analysed, resulting in much smaller local explanations than ProtoPNet.

Visualizations and Discussion. Figure 3 shows a snippet of a ProtoTree trained on CUB. From analysing various ProtoTrees, we conclude that prototypes are in general perceptually relevant, and successfully cluster similar-looking classes. Similar to ProtoPNet [4], some prototypes seem to focus on background. This is not necessarily an error in our visualization but shows that a ProtoTree can reveal learned biases. For example, Fig. 3 (top right) shows a green leaf to distinguish between a Gray Catbird and a Black Tern, because the latter is in the training data usually surrounded by sky or water. Further research could investigate to what extent undesired prototypes can be 'fixed' with a human-in-the-loop that replaces them with a manually selected patch, in order to create a model that is completely "right for the right reasons" [20]. Furthermore, we found that human's perceptual similarity could differ from similarity assigned by the model, since it is not always clear why the model considered an image highly similar to a prototype. The visualized prototypes could therefore be further explained by indicating whether *e.g*. color or shape was most important, as presented by [19], or by showing a cluster of patches. Especially prototypes close to the root of the tree are sometimes not as clear and semantically meaningful as prototypes closer to leaves. This is probably due to the binary tree structure that requires a patch from a training image to split the data into two subsets. A natural progression of this work would be to investigate non-binary trees, with multiple prototypes per node.

5. Conclusion

We presented the Neural Prototype Tree (ProtoTree) for intrinsically interpretable fine-grained image recognition. Whereas the Prototypical Part Network (ProtoPNet) [4] presents a user a large number of prototypes, our novel architecture with end-to-end training procedure improves interpretability by arranging the prototypes in a hierarchical tree structure. This breaks up the reasoning process in small steps which simplifies model comprehension and error analysis, and reduces the number of prototypes by a factor of 10. Most learned prototypes are semantically relevant, which results in a fully simulatable model. Additionally, we outperform ProtoPNet [4] on the CUB-200-2011 and Stanford Cars data sets. An ensemble of 5 ProtoTrees approximates the accuracy of non-interpretable state-of-the art models, while still having fewer prototypes than ProtoPNet [4]. Thus, ProtoTree achieves competitive performance while maintaining intrinsic interpretability. As a result, our work questions the existence of an accuracy-interpretability trade-off and stimulates novel usage of powerful neural networks as backbone for interpretable, predictive models.

References

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6, 2018.

[2] Stephan Alaniz and Zeynep Akata. Explainable observer-classifier for explainable binary decisions. *arXiv preprint arXiv:1902.01780*, 2019.

[3] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[4] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems 32*. 2019.

[5] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, Mar. 2014.

[6] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[7] Kamaledin Ghiasi-Shirazi. Generalizing the convolution operator in convolutional neural networks. *Neural Processing Letters*, 50(3):2627–2646, 2019.

[8] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1819–1822. IEEE, 2012.

[11] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. *The (Un)reliability of Saliency Methods*, pages 267–280. Springer International Publishing, Cham, 2019.

[12] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

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

[14] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. *arXiv preprint arXiv:1907.09294*, 2019.

[15] J. Liang, J. Guo, Y. Guo, and S. Lao. Adaptive triplet model for fine-grained visual categorization. *IEEE Access*, 6, 2018.

[16] Zachary C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, June 2018.

[17] X. Ma and A. Boukerche. An ai-based visual attention model for vehicle make and model recognition. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2020.

[18] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[19] Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. This looks like that, because ... explaining prototypes for interpretable image recognition, 2020.

[20] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

[21] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. volume 97 of *Proceedings of Machine Learning Research*, pages 6166–6175, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

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

[23] Heliang Zheng, Jianlong Fu, Zheng-Jun Zha, and Jiebo Luo. Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[24] Kuo Zhong, Ying Wei, Chun Yuan, Haoli Bai, and Junzhou Huang. Translider: Transfer ensemble learning from exploitation to exploration. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 368–378, New York, NY, USA, 2020. Association for Computing Machinery.