A Robust Prototype-Based Network with Interpretable RBF Classifier Foundations

Sascha Saralajew,[1] Ashish Rana,[1] Thomas Villmann,[2] and Ammar Shaker[1]

[1]NEC Laboratories Europe, Germany
[2]University of Applied Sciences Mittweida, Germany
{sascha.saralajew, ashish.rana, ammar.shaker}@neclab.eu, villmann@hs-mittweida.de

Abstract

Prototype-based classification learning methods are known to be inherently interpretable. However, this paradigm suffers from major limitations compared to deep models, such as lower performance. This led to the development of the so-called deep Prototype-Based Networks (PBNs), also known as prototypical parts models. In this work, we analyze these models with respect to different properties, including interpretability. In particular, we focus on the Classification-by-Components (CBC) approach, which uses a probabilistic model to ensure interpretability and can be used as a shallow or deep architecture. We show that this model has several shortcomings, like creating contradicting explanations. Based on these findings, we propose an extension of CBC that solves these issues. Moreover, we prove that this extension has robustness guarantees and derive a loss that optimizes robustness. Additionally, our analysis shows that most (deep) PBNs are related to (deep) RBF classifiers, which implies that our robustness guarantees generalize to shallow RBF classifiers. The empirical evaluation demonstrates that our deep PBN yields state-of-the-art classification accuracy on different benchmarks while resolving the interpretability shortcomings of other approaches. Further, our shallow PBN variant outperforms other shallow PBNs while being inherently interpretable and exhibiting provable robustness guarantees.

1 Motivation and Context

Two principal streams exist in the field of explainable machine learning: (1) post-processing methods (post-hoc approaches) that try to explain the prediction process of an existing model, such as LIME and SHAP (see Marcinkevičs and Vogt 2023, for an overview), and (2) the design of machine learning methods with inherently interpretable prediction processes (Rudin 2019). While the former could create non-faithful explanations due to only approximating the output distribution of a black box model without explaining its internal logic, it is claimed that inherently interpretable methods always generate faithful explanations (Rudin 2019). According to Molnar (2022), a model is called *interpretable* if its behavior and predictions are understandable to humans. Moreover, when the provided explanations lead to a correct interpretation of the model, this interpretation enriches the user (or developer) with an understanding of how the model works, how it can be fixed or improved, and whether it can be trusted (Ribeiro, Singh, and Guestrin 2016).

A well-known category of interpretable models for classification tasks is (shallow) Prototype-Based Networks (PBN) such as LVQ (e. g., Biehl, Hammer, and Villmann 2016). These models are interpretable because (1) the learned class-specific prototypes[1] are either from the input space or can be easily mapped to it; belonging to the input space helps summarize the differentiating factors of the input data and provides trusted exemplars for each class, (2) the dissimilarity computations are given by human comprehensible equations such that differences between inputs and learned prototypes can be understood, (3) the classification rule based on the dissimilarities is intelligible (e. g., winner-takes-all principle); see Bancos et al. (2020) for an interpretability application. Despite being interpretable, these models also face limitations: (1) The number of parameters becomes large on complex data since the prototypes are class-specific and are defined in the input space.[2] (2) The classification performance is behind that of deep neural architectures as the dissimilarity functions and the classification rules are straightforward to ensure interpretability (Villmann, Bohnsack, and Kaden 2017).

To fix these limitations, researchers investigated the integration of prototype-based classification heads with deep neural feature extractors to build deep interpretable PBNs and designed numerous architectures such as ProtoPNet (Chen et al. 2019), ProtoPool (Rymarczyk et al. 2022), CBC (Classification-By-Components; Saralajew et al. 2019), and PIPNet (Nauta et al. 2023). The generated results of these models are impressive as they achieve state-of-the-art classification accuracy on fine-grained image classification, and some show a good performance in rejecting Out-Of-Distribution (OOD) examples (e. g., PIPNet). The high-level structure of these models follows the same principles (see Fig. 1): (1) embedding of the input data in a latent space by a Neural Network (NN), denoted as feature extractor backbone; (2) measuring the dissimilarity (or similarity) between the embedding and the latent prototypes; (3) prediction computation after aggregating the dissimilarities by a shallow

[1]Usually, prototypes are class-specific, and components, centers, or centroids are class-unspecific.

[2]A ResNet50 on ImageNet has 26 M parameters, whereas an LVQ model with one prototype per class has 150 M.

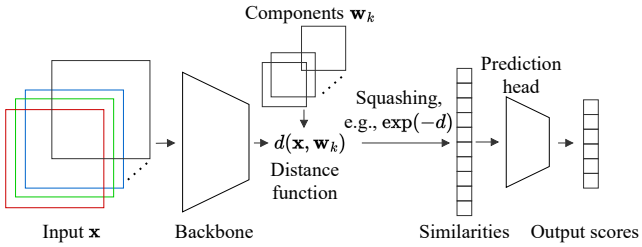

Figure 1: General architecture of deep PBNs.

model (realizes the classification rule), denoted as classification head. In this paradigm, the differences between the proposed architectures are often subtle, such as imposing sparsity, the usage of negative reasoning, and whether they can be used as a shallow model. Moreover, all architectures are supposed to generate interpretable models. But is this genuinely accurate?

In this paper, we investigate PBNs and make the following contributions:

1. We show that deep PBNs are related to deep RBF classifiers. Building on this finding, we explain why these models are effective for OOD detection.
2. We discuss why current *deep* PBNs are not interpretable and demonstrate how the interpretability level of the models varies between the different architectures.
3. Building on CBCs and their relation to RBF networks, we design a prototype-based classification head that can use negative reasoning in a sound probabilistic way and fixes the interpretability issue of other heads.
4. We derive robustness bounds for our classification head (shallow PBN), including a loss that provably optimizes robustness. Further, the relation shown gives the first loss that optimizes the robustness of RBF classifiers.

The paper's outline is as follows: In Sec. 2, we review deep PBNs and discuss their relation to RBF networks (Broomhead and Lowe 1988) and several properties. Based on the identified shortcomings, in Sec. 3, we propose an extension of CBC so that the interpretability is sound and negative reasoning is used. Additionally, we show that the shallow version of this architecture has provable robustness guarantees. Sec. 4 presents the experimental evaluation of our claims. Finally, a discussion and conclusion are presented. Please see Saralajew et al. (2024) for the corresponding appendix.

2 Review of Deep Prototype-based Networks

In the following section, we review the differences between deep PBNs and show their relation to RBF networks. Thereafter, we discuss the interpretability of these methods using the established relation. Later, we explain why PBNs are suitable for OOD detection and analyze the role of negative reasoning.

Differences between the architectures and their relation to RBF networks. Fig. 1 shows the general architecture of most deep PBNs. We use the shown building blocks to characterize existing approaches in Tab. 1 along the following dimensions:

- **Backbone:** Single, multiple, or Siamese feature extractor, and whether the method has been tested without a feature extractor (shallow model).
- **Latent prototypes:** Whether the prototypes are defined in the input or the latent space and if they are back-projected to training samples (Chen et al. 2019). This dimension also indicates if prototypes are class-specific.
- **Similarity:** The used similarity function. RBF refers to the standard squared exponential kernel. If a different nonlinear function is used to construct the RBF, it is specified in parenthesis. Note that all RBFs use the Euclidean norm.
- **Linear layer constraints:** The constraints on the final linear prediction layer or the stated approach to compute the output if no linear output layer is used. The l_1 regularization is only applied to connections that connect similarity scores (slots, etc.) with incorrect classes.
- **Single loss term:** Whether multiple loss terms are used.
- **Main contribution:** The primary contribution of the proposed architecture compared to previous work.

We identified the following architectures by reviewing top-tier venue papers: LeNet5 (LeCun et al. 1998), ProtoPNet, CBC, Hierarchical ProtoPNet (Hase et al. 2019), ProtoAttend (Arik and Pfister 2020), ProtoTree (Nauta, van Bree, and Seifert 2021), ProtoPShare (Rymarczyk et al. 2022), TesNet (Wang et al. 2021), Deformable ProtoPNet (Donnelly, Barnett, and Chen 2022), ProtoPool, and PIPNet. Moreover, we added LucidPPN (Pach et al. 2024) and ProtoViT (Ma et al. 2024) as it is the most recent publication in the field.

Considering Fig. 1, we realize that the head of a deep PBN is an RBF network if a linear layer is used for prediction. Combined with a feature extractor, we obtain *deep RBF networks* (e. g., Asadi et al. 2021). Notably, the first deep PBN is LeNet5, where RBF heads are used to measure the similarity between inputs and the so-called "model" (prototype) of the class. Starting with ProtoPNet, the existing architectures (see Tab. 1) build on each other (except for CBC and ProtoAttend), and almost all use an RBF network with some constraints or regularizers as classification heads. Consequently, changes between the architectures are incremental, and concepts persist for some time once introduced. Recently, researchers abandoned the idea of back-projecting prototypes and started using dot products instead of RBF functions, which implicitly defines prototypes as convolutional filter kernels (PIPNet, LucidPPN).

On the interpretability of deep PBNs. Using the definition of interpretability in Sec. 1 and the relation to RBF networks, we discuss the interpretability of deep PBNs. First, it should be noted that RBF networks and shallow PBNs learn representations in the input space (centroids and prototypes, respectively), and both use these representations to measure the (dis)-similarity to given samples. At the same time, these two paradigms differ in two aspects: (1) RBFs' usage of non-class specific centroids and (2) PBNs' usage of the human-comprehensible winner-takes-all rule instead of a linear predictor over the prototypes.

	Backbone	Latent Proto.	Similarity	Linear Layer Constraints	Single Loss	Main Contribution
LeNet5	single	yes	RBF	none	no	CNN with RBF head
ProtoPNet*	single	*yes**	RBF (log)	l_1 reg.	no	(deep) NN with prototype classification head
CBC*	Siamese*	no	RBF or ReLU-cosine	probabilistic	yes	negative/positive/indefinite reasoning
Hier. ProtoPNet	single	*yes**	RBF (log)	l_1 reg.	no	hierarchical classification
ProtoAttend*	Siamese	no	relational attention	none	no	attention for prototype selection
ProtoTree	single	yes*	RBF	(soft) tree*	yes	tree upon similarities
ProtoPShare	single	yes*	RBF (log)	l_1 reg.	no	prototype sharing between classes
TesNet	single	*yes**	dot-product	l_1 reg.	no	orthogonal prototypes
Def. ProtoPNet	single	*yes**	RBF (cosine)	l_1 reg.	no	deformable prototypes (shift correction)
ProtoPool	single	yes*	RBF (focal similarity)	l_1 reg.	no	differentiable prototype selection
PIPNet	single	yes	softmax dot product	non-negative	no	self-supervised pre-training
LucidPPN	multiple	yes	sigmoid dot product	average*	no	color and shape backbone
ProtoViT	single	yes*	scaled sum of cosine	l_1 reg.	no	deformable prototypes through vision transformer
Ours	single*	yes	RBF or softmax dot product	probabilistic	yes	trainable priors and provable robustness

Table 1: Characterization of existing architectures along the specified dimensions. Note that the order is chronological. Methods that are not directly based on the previously published methods are marked with an asterisk. The asterisk in the remaining columns stands for the ability to omit the feature extractor in Backbone, the usage of back-projection of latent prototypes in Latent Prototype, and an alternative approach for the output computation in Linear Layer Constraints (i. e., no application of a linear layer with a regularization or constraint). The italic typeface in Latent Prototype states that the prototypes are class-specific.

The first aspect overcomes the Limitation (1) mentioned in Sec. 1 without harming the interpretability. The second aspect poses a problem for interpretation, which explains the lack of studies applying RBF networks for interpretable machine learning. The problem starts with the unconstrained weights in the linear layer, which lead to unbounded and incomparable scores (e. g., it is unclear how to interpret a high score or weight). Further, this could result in situations where the closest (most similar) centroids do not contribute the most to the classification score compared to less similar centroids that are overemphasized by large weights. Hence, this breaks the paradigm that the most similar centroids (or prototypes) define the class label.

What does this imply for deep PBNs? First, the interpretation of the classification head suffers from the same difficulties as an RBF network if no appropriate constraints are applied (e. g., the average computation of LucidPPN). Therefore, the most similar prototypes for an input do not necessarily define the class label. For example, PIPNet trained on CUB (Wah et al. 2011) uses average weights of 14.1 for class blue jay and 8.7 for green jay. This indicates that PIPNet overemphasizes small similarity values so that the interpretation of the influential prototypes could be incorrect; further results in Sec. 4. Second, since the similarity is computed in a latent space defined by a deep NN, it is unclear why two samples are close or distant due to the black-box nature of deep NNs. Thus, it is *misleading* to denote a deep PBN as interpretable. In the best case, it can be denoted as *partially interpretable* as it gives insights into the final classification step, assuming that the classification head is well-designed. Note that the interpretability of these methods is also questioned by others (e. g., Hoffmann et al. 2021; Pazzani et al. 2022; Sacha et al. 2024; Wolf et al. 2024).

On the OOD detection properties. In CBC (rejection of predictions), Hierarchical ProtoPNet (novel class detection), ProtoAttend (OOD detection), and PIPNet (OOD detection), it was shown that deep PBNs are suitable for identifying OOD samples. This ability can be attributed to the RBF architecture if the model is clearly related to RBF models. Hein, Andriushchenko, and Bitterwolf (2019) proved that RBF networks produce low-confidence predictions when *a given sample is far away from all centroids (prototypes)* because the applied softmax squashing enforces the predictions of all classes to be uniform. Van Amersfoort et al. (2020) built on this idea and empirically showed that deep feature extractors with an RBF head and a winner-takes-all rule (so a deep PBN) can be used for uncertainty estimation and, thus, OOD detection. The published results for deep PBNs also confirm this property van Amersfoort et al. (2020) observed. Empirically, this property transfers beyond RBF-related architectures, as architectures like PIPNet show a remarkable

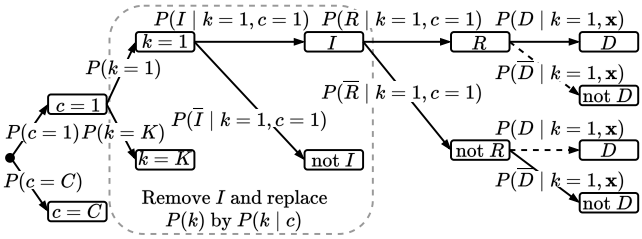

Figure 2: Probability tree diagram of the original CBC with the changes we propose for our extension in the gray box.

OOD performance using a non-RBF similarity.

The role of negative reasoning. Positive reasoning is well-defined as retrieving evidence of a given class from present features, but the literature does not reach a consensus about negative reasoning. In CBC, it means the retrieval of evidence from absent features. In contrast, in ProtoPNet, this refers to the reduction of the final score due to a negative weight associated with an active prototype. Other methods in the literature either penalize negative reasoning (e. g., ProtoPNet) by a regularization term or avoid it by a constraint (e. g., PIPNet); see the Constraints column in Tab. 1 . The challenge posed by negative reasoning in these architectures is mainly about interpretation, as it is not an intuitive reasoning principle of humans (according to Chen et al. 2019) and complicates the explanation strategies. In a notable contrast, in CBC, inspired by cognitive science results (e. g., Hsu et al. 2017), the authors modeled negative reasoning from a probabilistic perspective, making its interpretation mathematically sound. For the remainder of the paper, we refer by negative reasoning to the retrieval of evidence from features that have to be absent.

3 Classification-by-Components Networks

We now review the original CBC architecture and show its limitations. Based on that, we propose our CBC—simply denoted as CBC and the old version is denoted as *original CBC*—that overcomes these limitations and realizes a strong link to RBF networks. Then, we show how a CBC can be learned efficiently and derive robustness lower bounds.

Review of the original CBC method. Components are the core concept of the original CBC, where a component is a pattern that contributes to the classification process by its presence (positive reasoning; the component must be close) or absence (negative reasoning; the component must be far) without being tied to a specific class label. A component can also abstain from the classification process, which is called indefinite reasoning (modeled via importance). The original CBC is based on a probability tree diagram to model the interaction between the detection of components in input samples and the usage of detection responses to model the output probability (called reasoning). The probability tree, Fig. 2, employs five random variables: c, the class label; k, the component; I, the importance of a component (binary); R, the requiredness for reasoning (binary); D, the detection of a component (binary). The probability tree constructs the following: $P(k)$, the prior of the k-th component to appear; $P(I|k,c)$ and $P(R|k,c)$ are the importance and the requiredness probabilities of the k-th component for the class c; $P(D|k,\mathbf{x})$, the detection probability of the k-th component in the input $\mathbf{x}$. $P\left(\overline{D}|k,\mathbf{x}\right)$ is the complementary probability, that is, *not* detecting the k-th component in $\mathbf{x}$. An agreement A is a path in the tree (see solid lines in Fig. 2) that depicts the positive influence of the k-th component on class c by either being detected (D) and required (R) or not detected ($\overline{D}$) and not required ($\overline{R}$). The output probability $p_c(\mathbf{x}) = P(A|I,\mathbf{x},c)$ for class c is derived from the agreement A using the following expression:

$$\frac{\sum_k \left(P(R,I|k,c)\,P(D|k,\mathbf{x}) + P\left(\overline{R},I|k,c\right)P\left(\overline{D}|k,\mathbf{x}\right)\right)P(k)}{\sum_k \left(P(R,I|k,c) + P\left(\overline{R},I|k,c\right)\right)P(k)}. \tag{1}$$

The defined probabilities and components are learned by minimizing the margin loss (maximizing the probability gap)

$$\min\left\{\max_{c'\neq y} p_{c'}(\mathbf{x}) - p_y(\mathbf{x}) + \gamma, 0\right\}, \tag{2}$$

with $\gamma \in [0,1]$ being the margin value, y being the correct class label of $\mathbf{x}$, and c' being any class label other than y. The model can be used without or with a feature extractor (see Fig. 1); that is, the distance computation occurs in the learned latent space. An original CBC without a feature extractor realizes an extension of traditional PBNs, overcoming Limitation (1) while posing new difficulties.

The architecture is difficult to train as it often converges to a bad local minimum (see Sec. 4), and the explanations can be counterintuitive. To see this, note that $P(R,I|k,c) + P\left(\overline{R},I|k,c\right) + P\left(\overline{I}|k,c\right) = 1$ for each k. Thus, one can scale the reasoning probabilities $P(R,I|k,c)$ and $P\left(\overline{R},I|k,c\right)$ in Eq. (1) by any factor $\alpha > 0$ as long as $P(R,I|k,c), P\left(\overline{R},I|k,c\right) \in [0,1]$ remains valid without changing the output probability $p_c(\mathbf{x})$. Assuming that $p_c(\mathbf{x}) = 1$, this result can be obtained from nearly zero reasoning probabilities, giving confident predictions from infinitesimal reasoning evidence. This contradicts the design principle of the original CBC approach, as $p_c(\mathbf{x}) = 1$ should only be generated if the model is certain in its reasoning. At the same time, this result implies that the optimal output ($p_c(\mathbf{x}) = 1$) is not unique with a wide range of flawed feasible solutions, thus causing the model to converge to bad local minima.

Our extension of the original CBC method. In CBC, both problems mentioned above are caused by the indefinite reasoning probability $P\left(\overline{I}|k,c\right)$ together with the component prior $P(k)$. These probabilities model the extent to which a component is used in the classification process; hence, they both serve the same purpose, as confirmed by fixing $P(k)$ to be uniform in the original CBC. Removing $P\left(\overline{I}|k,c\right)$ from the model eliminates the problematic model's tolerance towards scaling by a factor α. Still, it causes missing support for allowing components to remain irrelevant (to abstain), as explained by Saralajew et al. (2019) in Figure 1. Similarly, allowing the prior $P(k)$ to be trainable does not generalize to cover the property of class-specific component priors.

We now present our modification to the original CBC to overcome the difficulties. We propose to remove the importance variable I and substitute it with the *trainable class-wise component* prior $P(k \mid c)$, see Fig. 2. The output probability $p_c(\mathbf{x}) = P(A \mid \mathbf{x}, c)$, using the agreement, becomes

$$\begin{aligned} &P(A \mid \mathbf{x}, c) = \\ &\sum_k \left(P(R, D \mid x, c, k) + P\left(\overline{R}, \overline{D} \mid \mathbf{x}, c, k\right)\right) P(k \mid c) = \\ &\sum_k \left(P(R \mid c, k)\, P(D \mid \mathbf{x}, k) + P\left(\overline{R} \mid c, k\right) P\left(\overline{D} \mid \mathbf{x}, k\right)\right) P(k \mid c) \end{aligned} \tag{3}$$

We introduce the following notations:

- The requiredness possibility vector $\mathbf{r}_c \in [0,1]^K$ contains the probabilities $P(R \mid c, k)$ for all k.
- The detection possibility vector $\mathbf{d}(\mathbf{x}) \in [0,1]^K$ contains the probabilities $P(D \mid \mathbf{x}, k)$ for all k.
- The component prior probability vector $\mathbf{b}_c \in [0,1]^K$ contains the probabilities $P(k \mid c)$ for all k.

Note that $\sum_k b_{c,k} = \sum_k P(k \mid c) = 1$, which is not necessarily true for $\mathbf{d}$ and $\mathbf{r}_c$. Now, Eq. (3) can be written as

$$p_c(\mathbf{x}) = \left(\mathbf{r}_c \circ \mathbf{d}(\mathbf{x}) + (\mathbf{1} - \mathbf{r}_c) \circ (\mathbf{1} - \mathbf{d}(\mathbf{x}))\right)^{\mathrm{T}} \mathbf{b}_c, \tag{4}$$

where $\circ$ is the Hadamard product. The detection probability can be any suitable function, like the following RBF:[3]

$$P(D \mid \mathbf{x}, k) = \exp\left(-\frac{d_E(\mathbf{x}, \mathbf{w}_k)}{\sigma_k}\right), \tag{5}$$

where d_E is the Euclidean distance, σ_k is the (trainable) component-dependent temperature, and $\mathbf{w}_k$ is the vector representation of component k. Using Eq. (4), similarly to other deep PBNs, the architecture is trained by optimizing the parameters of the components $\mathbf{w}_k$, the prior probabilities $\mathbf{b}_c$, and the reasoning possibility vector $\mathbf{r}_c$. For the optimization, the margin loss Eq. (2) can be used.

Learning the parameters in CBC models. When adopted without a feature extractor, learning a CBC model realizes an extension of shallow PBNs using components instead of prototypes (Limitation (1) in Sec. 1) and constitutes an interpretable RBF network (fixes the interpretability issues mentioned in Sec. 2). Note that in the computation of Eq. (3), the requiredness probabilities $P(R \mid c, k)$ and the component prior probabilities $P(k \mid c)$ occur jointly and provide the *reasoning probabilities* $P(R, k \mid c) = P(R \mid c, k) P(k \mid c)$. This simplification makes the association to RBF networks more explicit by rewriting Eq. (3) as $p_c(\mathbf{x}) = \sum_k \alpha_k P(D \mid \mathbf{x}, k) + \beta$, where $\alpha_k = P(R, k \mid c) - P\left(\overline{R}, k \mid c\right)$ is the weight and $\beta = \sum_k P\left(\overline{R}, k \mid c\right)$ is the bias.

Moreover, the network is simplified during training, and only the reasoning probabilities $P(R, k \mid c)$ are learned, leading to fewer multiplications of trainable parameters (simpler gradient computation graph). In practice, the trainable parameters $P(R, k \mid c)$ and $P\left(\overline{R}, k \mid c\right)$ take the form of the vector $\mathbf{v}_c \in \mathbb{R}^{2K}$ for each class, which is normalized to achieve $\sum_i \mathrm{softmax}(\mathbf{v}_c)_i = 1$. Within $\mathbf{v}_c$, the first half of the parameters represent the positive and the second half the negative reasoning probabilities. The computation of $p_c(\mathbf{x})$ becomes $\mathbf{v}_c^{\mathrm{T}} [\mathbf{d}(\mathbf{x}), (1 - \mathbf{d}(\mathbf{x}))]$, where the detection and no detection vectors are concatenated into one vector. Consequently, and again, the model realizes an RBF network that uses negative reasoning. If we block negative reasoning by setting the respective probabilities to zero, we obtain an RBF network with class-wise weights constrained while solving the interpretability issues from Sec. 2.

The proven robustness of the CBC architecture. In this section, we derive the robustness lower bound. We analyze the stability of the classification decision when *no* feature extractor is applied; with a feature extractor, the same stability analysis applies in the latent space. Given a data point $\mathbf{x} \in \mathbb{R}^n$ with the target label y, the input is correctly classified if the probability gap is positive:

$$p_y(\mathbf{x}) - \max_{c' \neq y} p_{c'}(\mathbf{x}) > 0. \tag{6}$$

Robustness comes from deriving a non-trivial lower bound for the maximum applicable perturbation $\varepsilon^* \in \mathbb{R}^n$ without having the predicted class label of $\mathbf{x}$ changed, that is,

$$p_y(\mathbf{x} + \varepsilon^*) - \max_{c' \neq y} p_{c'}(\mathbf{x} + \varepsilon^*) > 0; \tag{7}$$

the strength of the perturbation is given by $\|\varepsilon^*\|$. Thm. 1 derives a lower bound of $\|\varepsilon^*\|$ for detection probability functions of the form Eq. (5) where d_E is *any* distance function induced by the selected norm $\|\cdot\|$. Thm. 2 extends this derivation to squared norms (e. g., Gaussian kernel) so that the result can be applied to standard Gaussian RBF networks using the established relation.

Theorem 1. *The robustness of a correctly classified sample* $\mathbf{x}$ *with class label* y *is lower bounded by*

$$\|\varepsilon^*\| \geq \underbrace{\kappa \min_{c' \neq y} \left(\ln\left(-\frac{B_{c'} + \sqrt{B_{c'}^2 - 4A_{c'}C_{c'}}}{2A_{c'}}\right)\right)}_{=:\delta} > 0, \tag{8}$$

when $A_{c'} \neq 0$*, where*

$$\begin{aligned} A_{c'} &= \left((\mathbf{r}_y - \mathbf{1}) \circ \mathbf{b}_y - \mathbf{r}_{c'} \circ \mathbf{b}_{c'}\right)^{\mathrm{T}} \mathbf{d}(\mathbf{x}), \\ B_{c'} &= (\mathbf{1} - \mathbf{r}_y)^{\mathrm{T}} \mathbf{b}_y - (\mathbf{1} - \mathbf{r}_{c'})^{\mathrm{T}} \mathbf{b}_{c'}, \\ C_{c'} &= \left(\mathbf{r}_y \circ \mathbf{b}_y - (\mathbf{r}_{c'} - \mathbf{1}) \circ \mathbf{b}_{c'}\right)^{\mathrm{T}} \mathbf{d}(\mathbf{x}), \end{aligned}$$

and $\kappa = \sigma_{min} = \min_k \sigma_k$.

All proofs can be found in Appx. B. Additionally, it can be shown that δ in Eq. (8) is negative if the sample is incorrectly classified. Therefore, δ in Eq. (8) can be used as a loss function to optimize the model for stability. Of course, this loss can be clipped at a threshold $\gamma > 0$ so that the network optimizes for robustness of at most γ.

[3] The detection probability must be a similarity measure $\mathbb{R}^n \times \mathbb{R}^n \to [0,1]$ such that $\mathbf{x} = \mathbf{w}_k$ implies a similarity of 1.0.

Theorem 2. *If we use the standard RBF kernel (squared norm), then Eq. (8) becomes* $\|\varepsilon^*\| \geq -\frac{\beta}{3} + \sqrt{\frac{\beta^2}{9} + \delta} > 0$ *with* $\kappa = \frac{\sigma_{min}}{3}$ *and* $\beta = \max_k d(\mathbf{x}, \mathbf{w}_k)$.

Again, this result helps to construct a loss function that maximizes robustness. For standard Gaussian kernel RBF networks with class-wise weights $\mathbf{v}_c$ constrained to probability vectors, the main part δ of the function is simplified to

$$\frac{\sigma_{min}}{6} \min_{c' \neq y} \ln \left(\frac{\mathbf{v}_y^{\mathrm{T}} \mathbf{d}(\mathbf{x})}{\mathbf{v}_{c'}^{\mathrm{T}} \mathbf{d}(\mathbf{x})} \right), \tag{9}$$

a log-likelihood ratio loss (e. g., Seo and Obermayer 2003).

The robustness with alternative distance functions. Similar to other shallow PBNs, CBCs can use alternative distance functions such as the Mahalanobis distance or the tangent distance (e. g., Haasdonk and Keysers 2002)

$$d_T(\mathbf{x}, S) = \min_{\theta \in \mathbb{R}^r} d_E(\mathbf{x}, \mathbf{w} + \mathbf{W}\theta), \tag{10}$$

where $S = \{\mathbf{w} + \mathbf{W}\theta \mid \theta \in \mathbb{R}^r\}$ is a trainable r-dimensional affine subspace with $\mathbf{W}$ being a basis. By learning affine subspaces instead of points for the components, the discriminative power of the architecture is significantly improved (Saralajew, Holdijk, and Villmann 2020). Moreover, if this distance is used in a deep PBN, it realizes an extension of TesNet by learning disentangled concepts (each basis vector in $\mathbf{W}$ is a basis concept) but measures the distance with respect to d_T. See Appx. A for further details about this distance. Next, Thm. 3 extends the lower bound derived in Thm. 1 for the tangent distance.

Theorem 3. *If we use the tangent distance in Eq. (5), Eq. (8) holds with* $\kappa = \frac{1}{2}\sigma_{min}$ *and* $\|\cdot\|$ *being the Euclidean norm.*

A similar result was proven for LVQ with the tangent distance (Saralajew, Holdijk, and Villmann 2020).

Final remarks. Our proposed CBC resolves the original approach's drawbacks. Further, the architecture can be derived from RBF networks by introducing interpretability constraints and negative reasoning. The method can be used as a head for deep PBNs or as a standalone for prototype-based classification learning. In all cases, the interpretability of the learned weights is guaranteed by the relation to the probability events. Appx. C presents further theoretical results.

4 Experiments

In this section, we test our CBC and the presented theories: (1) We analyze the accuracy and interpretability of our CBC and compare it to PIPNet. (2) We compare shallow CBCs with other shallow models, such as the original CBC. (3) To demonstrate our theorems, we analyze the adversarial robustness of shallow PBNs. Note that all accuracy results are reported in percentage; we train each model five times, and report the mean and standard deviation. [4]

[4]The source code is available at https://github.com/si-cim/cbc-aaai-2025.

	CUB	CARS	PETS
PIPNet	84.3 ± 0.2	88.2 ± 0.5	92.0 ± 0.3
ProtoPool	85.5 ± 0.1	88.9 ± 0.1	$87.2^* \pm 0.1$
ProtoViT	85.8 ± 0.2	92.4 ± 0.1	$93.3^* \pm 0.2$
CBC	$\mathbf{87.8 \pm 0.1}$	$\mathbf{93.0 \pm 0.0}$	$\mathbf{93.9 \pm 0.1}$
CBC pos. reas.	28.6 ± 0.8	25.3 ± 2.3	69.5 ± 5.1

Table 2: Test accuracy on different benchmark datasets. If available, we copied the accuracy values from the respective papers. Otherwise, we computed them (marked by an asterisk).

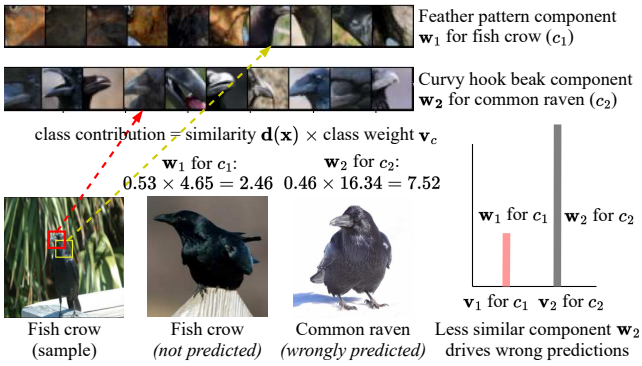

Figure 3: Fish crow gets incorrectly classified as common raven by PIPNet because of the overemphasis of weights.

Interpretability and performance assessment: Comparison with PIPNet. We evaluate the performance of CBC in comparison with PIPNet and the state-of-the-art deep PBN ProtoPool and ProtoViT[5] (CaiT-XXS 24; best-performing backbone). Since CBC can work with any backbone, we use PIPNet's ConvNeXt-tiny (Liu et al. 2022) architecture, the best-performing one from PIPNet. We extend PIPNet by only replacing the final classification layer with a CBC head. This way, the components become implicitly defined by the weights of the last convolutional layer with softmax-normalized dot product as a similarity. For training, we follow the pre-training protocol from PIPNet and extend the classification step using our proposed margin loss Eq. (2) with $\gamma = 0.025$. We benchmark the methods using CUB, CARS (Krause et al. 2013), and PETS (Parkhi et al. 2012) datasets.

The test accuracy results of our model sets new benchmarks as shown in Tab. 2. To analyze the reason for this accuracy gain, we trained another PIPNet, replacing the ReLU constraint on the classification weights with a softmax. By this, we avoid the mentioned interpretability issues and obtain a CBC restricted to positive reasoning only (CBC pos. reas.). This model constantly scores behind CBC with negative reasoning. Hence, the accuracy gain can be attributed to the usefulness of negative reasoning.

To assess the interpretability, we use PIPNet's approach to determine the top-10 component visualizations from the training dataset. Fig. 3 shows an example that is wrongly

[5]It was not published when the submission draft for AAAI was written.

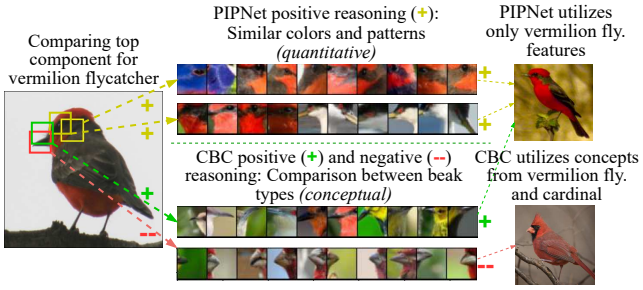

Figure 4: The comparative analysis of PIPNet and CBC for the vermilion flycatcher, where negative reasoning is used.

classified by PIPNet due to the overemphasis of specific weights. Ravens have curved hook-like beaks and regions of larger feathers, whereas crows have streamlined beaks and small feathers. The crow depicted in this figure is wrongly classified as a raven because the most similar component $\mathbf{w}_1$ (feather), which correctly indicates that it is a crow, is overshadowed by the less similar component $\mathbf{w}_2$ (hook-like beak) that has a higher weight. This example confirms our hypothesis from Sec. 2 that non-normalized weights hinder interpretability by preventing the most relevant prototypes from influencing the prediction.

Fig. 4 shows an example of positive and negative reasoning to distinguish between two close bird species. PIPNet uses positive reasoning to match based on regions with similar colors or color contrasts, focusing less on contextual understanding. CBC focuses on learning concepts like the pointed streamlined beak irrespective of the bird species or color pattern patches. As this component is similar to the beak of the depicted bird (vermillion flycatcher), it contributes to the classification as a vermillion flycatcher (positive reasoning). At the same time, CBC distinguishes the vermillion flycatcher from a similar species in appearance, the cardinal, by using negative reasoning with the absence of the cardinal's broad beak.

To quantitatively assess how different components are used across different classes by learning class-specific component priors, we computed the Jensen–Shannon divergence between the priors of each pair of bird classes. The divergence depicts how the distributions of the components' priors differ across classes. The following shows this for the Black-footed Albatross compared to three other species: Laysan Albatross 1.1, Crested Auklet 5.1, and Least Auklet 4.4. These results indicate a smaller divergence to the Laysan Albatross, a close relative from the same family, and greater divergences to more distantly related species. This demonstrates that our approach generally shares components across similar classes while using different components for others. Again, this result underlines the importance of learning class-specific component priors. Appx. D.1 presents model training details, ResNet50 results, and more interpretability results.

Comparison with shallow models. In this experiment, we compare CBC with its variants and other baseline models. Namely, we compare with GLVQ (Sato and Yamada 1996), RBF networks, and the original CBC. We also implement

	Accuracy	Emp. Rob.	Cert. Rob.
GLVQ	80.5 ± 0.6	59.6 ± 0.3	$\mathbf{32.3 \pm 0.3}$
RBF	$\mathbf{92.2 \pm 0.1}$	61.9 ± 0.9	–
original CBC	81.8 ± 2.0	$\mathbf{62.0 \pm 1.0}$	–
CBC	87.4 ± 0.3	68.1 ± 0.7	0.2 ± 0.1
RBF-norm	77.3 ± 0.2	57.7 ± 0.2	0.7 ± 0.0
CBC TD	$\mathbf{95.9 \pm 0.1}$	$\mathbf{84.5 \pm 0.2}$	0.0 ± 0.0
RBF-norm TD	92.1 ± 0.2	77.8 ± 0.4	0.0 ± 0.0
Robust CBC	87.8 ± 0.3	62.8 ± 0.3	$\mathbf{15.2 \pm 1.7}$
Robust CBC TD	91.9 ± 0.3	70.8 ± 0.5	1.6 ± 0.2

Table 3: Test, empirical robust, and certified robust accuracy of shallow PBNs. The robust accuracy is computed for $\|\varepsilon^*\| = 1$. The top shows prior art, and the bottom shows our models. We put the best accuracy for each category in bold.

RBF networks with softmax layer normalization (RBF-norm) and RBF networks with Tangent Distance (RBF-norm TD); see Eq. (10). We evaluate CBC with the Tangent Distance (CBC TD), with the robustness loss optimization (Robust CBC; see Thm. 1), and with both the robustness loss and the Tangent Distance (Robust CBC TD). All models are trained with the Euclidean distance unless the use of the tangent distance is indicated. The RBF models are trained by the cross-entropy loss, GLVQ by the GLVQ-loss function, and non-robust CBC models by the margin loss (Eq. (2) with $\gamma = 0.3$). Each model was trained and evaluated on MNIST (LeCun, Cortes, and Burges 1998). Each CBC and RBF can learn 20 components (or centroids) or two prototypes per class (GLVQ). The CBC models are trained with *two* reasoning concepts per class (two vectors $\mathbf{r}_c$ and $\mathbf{b}_c$ per class), component-wise temperatures, and squared Euclidean distances. The class output probability is given by the maximum over the class's two reasoning concepts. By this, we ensure that, similar to GLVQ, the models can learn two concepts (similar to prototypes) per class.

The results presented in Tab. 3 show that CBC outperforms the original CBC in terms of classification accuracy by over 5 %. By inspecting the learned components and probabilities, we observe that the original CBC converges to a sub-optimal solution by learning redundant components and not leveraging the advantage of multiple reasoning concepts per class. Our CBC learns less repetitive components and leverages the two reasoning concepts by learning class-specific components for several classes if required. Additionally, the table shows the advantage of using negative reasoning (cf. CBC and RBF-norm). While the class-wise softmax normalization in RBF-norm transforms it to a CBC with positive reasoning only, it remains outperformed by CBC with negative reasoning by 10 %. At the same time, both RBF-norm and CBC remain behind the plain RBF approach, showing how the interpretability constraints reduce the generalization. By using more advanced distance measures such as the tangent distance, we observe that the accuracy improves drastically while still being behind the plain models if they use the tangent distance. See Appx. D.2 for the complete set of results, including the comparison with more shallow models, the component visualizations, training with non-squared dis-

tances, and a shallow model with patch components, where the learned reasoning distinguishes between writing styles of the numeral seven.

Robustness evaluation. We evaluate the adversarial robustness of the already trained models from the shallow PBN experiments using the AutoAttack framework (Croce and Hein 2020) with the recommended setting and maximum perturbation strength 1.0, see Tab. 3. Additionally, using the result from Thm. 2, we compute the certified robustness by counting how many correctly classified samples have a lower bound greater or equal to 1.0. For GLVQ, we compute the certified robustness by the hypothesis margin (Saralajew, Holdijk, and Villmann 2020). Note that the certified robustness cannot be calculated for RBF and original CBC.

The results show that training a CBC with our robustified loss is possible and yields non-trivial certified robustness. For instance, the Robust CBC outperforms GLVQ, which is provably robust as well, in terms of accuracy and empirical robustness. With respect to the certified robustness, it is behind GLVQ, which can be attributed to the repeated application of the triangle inequality in order to derive the bound. Moreover, it should be noted that the certified robustness of Robust CBC TD is significantly lower than that of Robust CBC. This can be again attributed to the derived lower bound for the tangent distance, where the triangle inequality is applied once more. Hence, the stated bound in Thm. 3 is less tight compared to Thm. 1 and Thm. 2. See Appx. D.3 for the full results, including robustness curves and evaluation of robustified RBF networks using Thm. 2.

5 Discussion and Limitations

While we refrain from claiming that our deep model is fully interpretable, we believe it offers partial interpretability, providing valuable insights into the classification process, especially in the final layers. In contrast, the shallow version is inherently interpretable.

Compared to other deep PBNs, our model uses only a single loss term and neither forces the components to be close to training samples nor to be apart from each other. This is beneficial as it simplifies the training procedure drastically since no regularization terms have to be tuned. Even if we only use one loss term, our model converges to valuable components. However, interpreting these components is complex and requires expert knowledge. As a result, especially for deep PBNs, the interpretation could be largely shaped by the user's mental model, highlighting the importance of quantitative interpretation assessment approaches—something that is still lacking in the field. Additionally, by optimizing the single loss term, our model automatically learns sparse component representations without the issue of the learned representation being excessively sparse (see the additional PIPNet experiments in Appx. D.1).

During the deep model training, we observed that the CBC training behavior can be sensitive to pre-training and initializations. Further, training huge shallow models was challenging, especially when optimizing the robust loss: The model did not leverage all components as they often converged to the same point or failed to use all reasoning vectors if multiple reasoning vectors per class were provided. Additionally, training exponential functions (the detection probability) is sensitive to the selection of suitable temperature values. When we kept them trainable and individual per component, sometimes they became so small that the components did not learn anything even if the components had not converged to a suitable position in the data space. The same happened when we tried to apply exponential functions on top of a deep feature backbone, making it impossible to train such architectures reliably. These insights provide a foundation for refining our approach in future efforts.

6 Conclusion and Outlook

In this paper, we harmonize deep PBNs by showing a solid link to RBF networks. We also show how these models are not interpretable and only achieve partial interpretability in the best case. Inspired by these findings, we derive an improved CBC architecture that uses negative reasoning in a probabilistically sound way and ensures partial interpretability. Empirically, we demonstrate that the proposed deep PBN outperforms existing models on established benchmarks. Besides, the shallow version of our CBC is interpretable and provably robust. The shallow CBC is an attractive alternative to established models such as GLVQ as it resolves known limitations like the use of class-specific prototypes.

Open questions still exist and are left for future work: For example, a modification that prevents components from converging to the same point, along with the integration of spatial knowledge (to avoid global max-pooling), could improve deep PBNs, a challenge that the original CBC partially addressed. Moreover, in our evaluation, we focused on the assessment of our approach using image datasets, which is currently the commonly used benchmark domain for PBNs. However, future work should investigate the application of our approach to other domains, such as time series data. Moreover, to stabilize the training of the detection probability, one should explore the strategies proposed by Ghiasi-Shirazi (2019) or analyze the application of other detection probability functions (note that our theoretical results generalize to exponential functions with an arbitrary base). Finally, it is unclear why all shallow models, including non-robustified ones, exhibit good empirical robustness.

References

Arik, S. O.; and Pfister, T. 2020. ProtoAttend: Attention-Based Prototypical Learning. *Journal of Machine Learning Research*, 21(210): 1–35.

Asadi, K.; Parikh, N.; Parr, R. E.; Konidaris, G. D.; and Littman, M. L. 2021. Deep Radial-Basis Value Functions for Continuous Control. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence – AAAI 2021*, 6696–6704. AAAI Press.

Bancos, I.; Taylor, A. E.; Chortis, V.; Sitch, A. J.; Jenkinson, C.; Davidge-Pitts, C. J.; Lang, K.; Tsagarakis, S.; Macech, M.; Riester, A.; et al. 2020. Urine steroid metabolomics for the differential diagnosis of adrenal incidentalomas in the EURINE-ACT study: A prospective test validation study. *The Lancet Diabetes & Endocrinology*, 8(9): 773–781.

Biehl, M.; Hammer, B.; and Villmann, T. 2016. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews Cognitive Science*, 7(2): 92–111.

Broomhead, D. S.; and Lowe, D. 1988. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2(3).

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Su, J.; and Rudin, C. 2019. This Looks Like That: Deep Learning for Interpretable Image Recognition. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019*, 8928–8939. Vancouver, BC, Canada: Curran Associates, Inc.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning – ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, 2206–2216. Vienna, Austria: PMLR.

Donnelly, J.; Barnett, A. J.; and Chen, C. 2022. Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2022*, 10255–10265. New Orleans, LA, USA: IEEE.

Ghiasi-Shirazi, K. 2019. Generalizing the Convolution Operator in Convolutional Neural Networks. *Neural Processing Letters*, 50(3): 2627–2646.

Haasdonk, B.; and Keysers, D. 2002. Tangent distance kernels for support vector machines. In *Proceedings of the 16th International Conference on Pattern Recognition – ICPR 2002*, 864–868. Québec City, QC, Canada: IEEE.

Hase, P.; Chen, C.; Li, O.; and Rudin, C. 2019. Interpretable Image Recognition with Hierarchical Prototypes. In Law, E.; and Vaughan, J. W., eds., *Proceedings of the Seventh AAAI Conference on Human Computation and Crowdsourcing, – HCOMP 2019*, 32–40. Stevenson, WA, USA: AAAI Press.

Hein, M.; Andriushchenko, M.; and Bitterwolf, J. 2019. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2019*, 41–50. Long Beach, CA, USA: IEEE.

Hoffmann, A.; Fanconi, C.; Rade, R.; and Kohler, J. 2021. 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*.

Hsu, A. S.; Horng, A.; Griffiths, T. L.; and Chater, N. 2017. When Absence of Evidence Is Evidence of Absence: Rational Inferences From Absent Data. *Cognitive Science*, 41(S5): 1155–1167.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3D Object Representations for Fine-Grained Categorization. In *2013 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2013, Sydney, Australia, December 1-8, 2013*, 554–561. IEEE Computer Society.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

LeCun, Y.; Cortes, C.; and Burges, C. J. 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

Liu, Z.; Mao, H.; Wu, C.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A ConvNet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2022*, 11966–11976. New Orleans, LA, USA: IEEE.

Ma, C.; Donnelly, J.; Liu, W.; Vosoughi, S.; Rudin, C.; and Chen, C. 2024. Interpretable Image Classification with Adaptive Prototype-based Vision Transformers. In *arXiv:2410.20722*. Accepted at NeurIPS 2024.

Marcinkevičs, R.; and Vogt, J. E. 2023. Interpretable and explainable machine learning: A methods-centric overview with concrete examples. *WIREs Data Mining and Knowledge Discovery*, 13(3): e1493.

Molnar, C. 2022. *Interpretable machine learning - A Guide for Making Black Box Models Explainable*. 2 edition.

Nauta, M.; Schlötterer, J.; van Keulen, M.; and Seifert, C. 2023. PIP-Net: Patch-Based Intuitive Prototypes for Interpretable Image Classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2023*, 2744–2753. IEEE.

Nauta, M.; van Bree, R.; and Seifert, C. 2021. Neural Prototype Trees for Interpretable Fine-grained Image Recognition. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition – CVPR 2021*, 14933–14943. Nashville, TN, USA: IEEE.

Pach, M.; Rymarczyk, D.; Lewandowska, K.; Tabor, J.; and Zielinski, B. 2024. LucidPPN: Unambiguous Prototypical Parts Network for User-centric Interpretable Computer Vision. In *arXiv:2405.14331*.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V. 2012. Cats and dogs. In *Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition – CVPR 2012*, 3498–3505. Providence, RI, USA: IEEE.

Pazzani, M. J.; Soltani, S.; Kaufman, R.; Qian, S.; and Hsiao, A. 2022. Expert-Informed, User-Centric Explanations for Machine Learning. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence – AAAI 2022*, 12280–12286. AAAI Press.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. ACM.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1: 206–215.

Rymarczyk, D.; Struski, L.; Górszczak, M.; Lewandowska,

K.; Tabor, J.; and Zielinski, B. 2022. Interpretable Image Classification with Differentiable Prototypes Assignment. In Avidan, S.; Brostow, G. J.; Cissé, M.; Farinella, G. M.; and Hassner, T., eds., *Proceedings of the 17th European Conference on Computer Vision – ECCV 2022*, volume 13672 of the Lecture Notes in Computer Science, 351–368. Tel Aviv, Israel: Springer.

Sacha, M.; Jura, B.; Rymarczyk, D.; Struski, L.; Tabor, J.; and Zielinski, B. 2024. Interpretability Benchmark for Evaluating Spatial Misalignment of Prototypical Parts Explanations. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth Conference on Artificial Intelligence,– AAAI 2024*, 21563–21573. Vancouver, Canada: AAAI Press.

Saralajew, S.; Holdijk, L.; Rees, M.; Asan, E.; and Villmann, T. 2019. Classification-by-components: Probabilistic modeling of reasoning over a set of components. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2019*, 2792–2803. Vancouver, BC, Canada: Curran Associates, Inc.

Saralajew, S.; Holdijk, L.; and Villmann, T. 2020. Fast Adversarial Robustness Certification of Nearest Prototype Classifiers for Arbitrary Seminorms. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Proceedings of the Neural Information Processing Systems Conference – NeurIPS 2020*, 13635–13650. Curran Associates, Inc.

Saralajew, S.; Rana, A.; Villmann, T.; and Shaker, A. 2024. A Robust Prototype-Based Network with Interpretable RBF Classifier Foundations. In *arXiv:2412.15499*. Accepted at AAAI 2025.

Sato, A.; and Yamada, K. 1996. Generalized Learning Vector Quantization. In Touretzky, D. S.; Mozer, M.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems 8: Proceedings of the Neural Information Processing Systems Conference – NIPS 1995*, 423–429. Denver, CO, USA: MIT Press.

Seo, S.; and Obermayer, K. 2003. Soft Learning Vector Quantization. *Neural Computation*, 15(7): 1589–1604.

van Amersfoort, J.; Smith, L.; Teh, Y. W.; and Gal, Y. 2020. Uncertainty Estimation Using a Single Deep Deterministic Neural Network. In *Proceedings of the 37th International Conference on Machine Learning – ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, 9690–9700. Vienna, Austria: PMLR.

Villmann, T.; Bohnsack, A.; and Kaden, M. 2017. Can Learning Vector Quantization be an Alternative to SVM and Deep Learning? - Recent Trends and Advanced Variants of Learning Vector Quantization for Classification Learning. *Journal of Artificial Intelligence and Soft Computing Research*, 7(1): 65–81.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. Caltech-UCSD Birds-200-2011 (CUB-200-2011). Technical Report CNS-TR-2011-001.

Wang, J.; Liu, H.; Wang, X.; and Jing, L. 2021. Interpretable Image Recognition by Constructing Transparent Embedding Space. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, 875–884. IEEE.

Wolf, T. N.; Bongratz, F.; Rickmann, A.; Pölsterl, S.; and Wachinger, C. 2024. Keep the Faith: Faithful Explanations in Convolutional Neural Networks for Case-Based Reasoning. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth Conference on Artificial Intelligence,– AAAI 2024*, 5921–5929. Vancouver, Canada: AAAI Press.