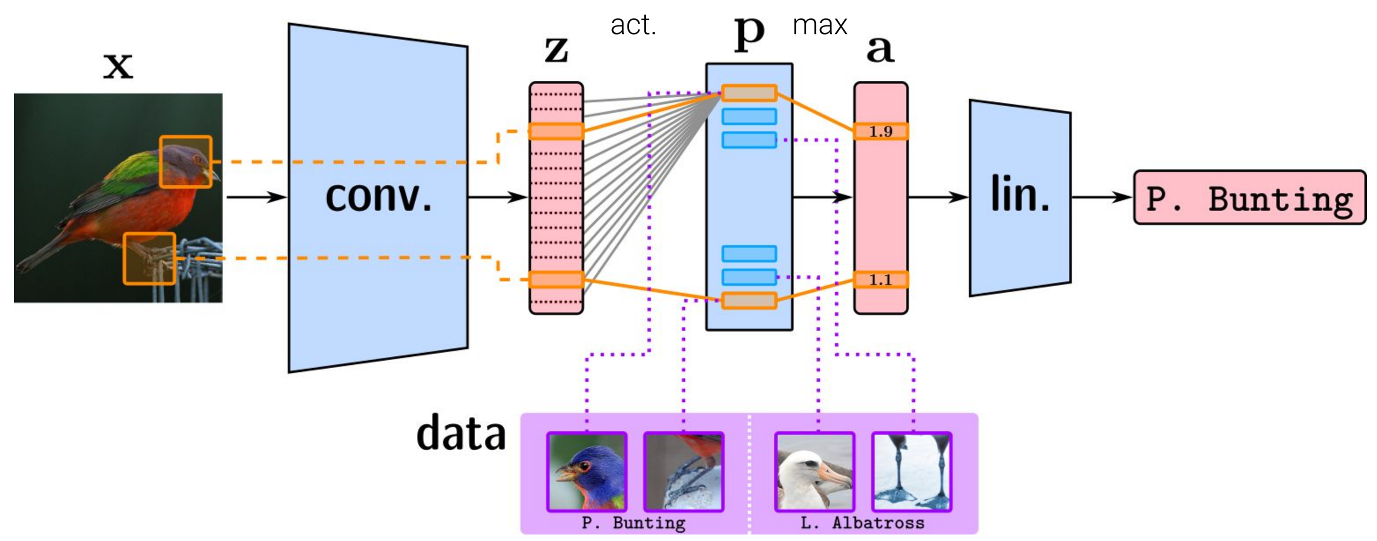
Concept-level Debugging of Part-Prototype Networks

Andrea Bontempelli, Stefano Teso*, Katya Tentori*, Fausto Giunchiglia*+, Andrea Passerini**

* University of Trento, Italy + Jilin University, China

Part-Prototype Networks



ProtoPNets are **self-explainable deep image classifiers** that work by:

1. Embedding image using (pre-trained) convolutional/pooling layers
2. Computing activation (similarity) of **part-prototypes** capturing concepts in training data
3. Aggregating activations into class probabilities

Their **explanations** highlight what part-prototypes (and training examples) are responsible for a given prediction *y = f(x)* and where they activate on the input.

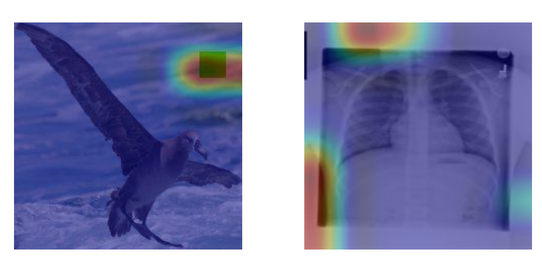
ProtoPNets are trained to optimize the log-likelihood of the data and two **clustering losses** that encourage the part-prototypes to strongly activate only on examples of their associated class.

Confounding in ProtoPNets

ProtoPNets exploit **confounds** in the data to maximize training set performance, for instance by classifying birds based on the background.

Compromises (esp. out-of-distribution) performance.

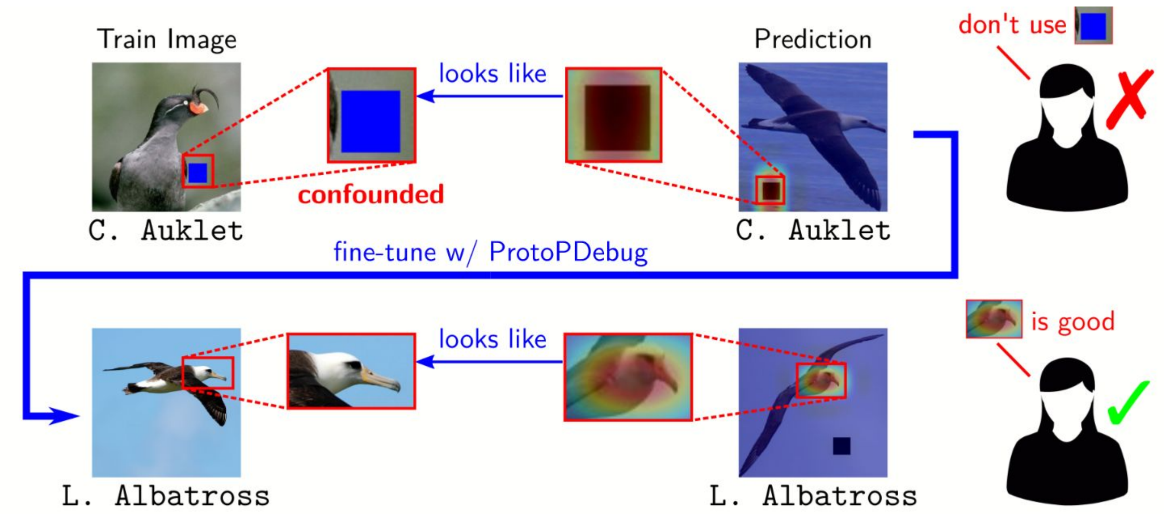
CUB200 COVID-19



Input-level vs Concept-based Debugging

Input-level (e.g., IAIA-BL [Barnett et al.])	**Concept-level** (ProtoPDebug)
*penalizes part-prototypes that activate on **pixels annotated as irrelevant***	*penalizes part-prototypes that correlate with **known-forbidden concepts** (e.g., "sea")*
▪ Attribution masks are local, i.e., they do not generalize across images. ▪ Hence, a substantial number of example must be annotated to fix the model. ▪ Acquiring per-pixel attribution annotations is expensive.	▪ Generalizes across images: one concept-level annotation = several pixel-level annotations. ▪ Speeds up convergence, avoids relapse. ▪ Cheap, click-based feedback!

Concept-level Debugging with ProtoPDebug



The "**cut-out**" is a box that covers 95% of the activation in the part-prototype saliency map, and it is extracted automatically by ProtoPNets.

Algorithm 1 A debugging session with ProtoPDebug. f is a ProtoPNet trained on data set D.

1: initialize $\mathcal{F} \leftarrow \varnothing, \mathcal{V} \leftarrow \varnothing$
2: **while** True **do**
3: **for** $\mathbf{p} \in \mathcal{P}$ **do**
4: **for each** $(\mathbf{x}, y)$ of the a training examples most activated by $\mathbf{p}$ **do**
5: **if** $\mathbf{p}$ appears confounded to user **then**
6: add cut-out $\mathbf{x}_R$ to $\mathcal{F}$
7: **else if** $\mathbf{p}$ appears high-quality to user **then**
8: add cut-out $\mathbf{x}_R$ to $\mathcal{V}$
9: **if** no confounds found **then**
10: **break**
11: fine-tune f by minimizing $\ell(\theta) + \lambda_f \ell_{\text{for}}(\theta) + \lambda_r \ell_{\text{rem}}(\theta)$
12: **return** f

After each round of feedback, in which it collects part-prototypes to be forbidden and remembered, ProtoPDebug fine-tunes the model by optimizing the ProtoPNet loss **augmented with two extra terms**:

▪ The **forgetting loss** penalizes part-prototypes for activating on forbidden concepts:

$$\ell_{\text{for}}(\theta) := \frac{1}{v} \sum_{y \in [v]} \max_{\substack{\mathbf{p} \in \mathcal{P}^y \\ \mathbf{f} \in \mathcal{F}_y}} \text{act}(\mathbf{p}, \mathbf{f})$$

▪ The **remembering loss** encourages at least one part-prototype to activate on a concept to be retained:

$$\ell_{\text{rem}}(\theta) := -\frac{1}{v} \sum_{y \in [v]} \min_{\substack{\mathbf{p} \in \mathcal{P}_y \\ \mathbf{v} \in \mathcal{V}_y}} \text{act}(\mathbf{p}, \mathbf{v})$$

Benefits: encourages model to focus on relevant concepts with **few clicks**; invariant to **concept order**; prevents model from **re-learning forbidden concepts** (simply deleting concepts does not); **robust** to cases where non-deleted concepts are useless.

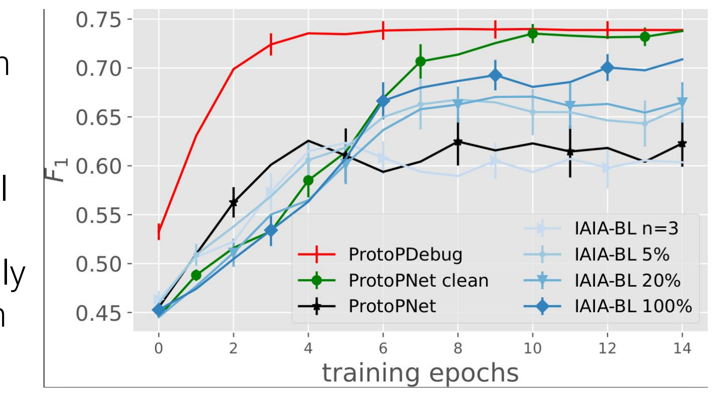
paper
arxiv.org/abs/2205.15769

code
github.com/abonte/protopdebug

Concept-level Debugging is Useful ...

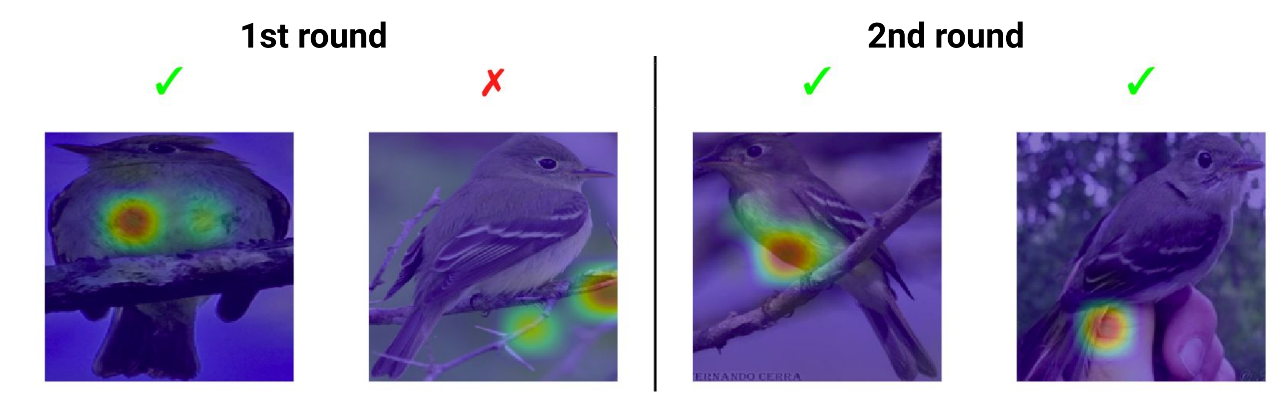
We add **synthetic confounds** (colored boxes) in the first five classes of CUB200, and measure *F1* score on a clean test set.

The baseline model (**black**) performs poorly, and a model trained on clean (**green**) data very well. IAIA-BL (**blue**) requires plenty of attribution masks to perform reasonably well. ProtoPDebug (**red**) achieves best performance with a single click per class.

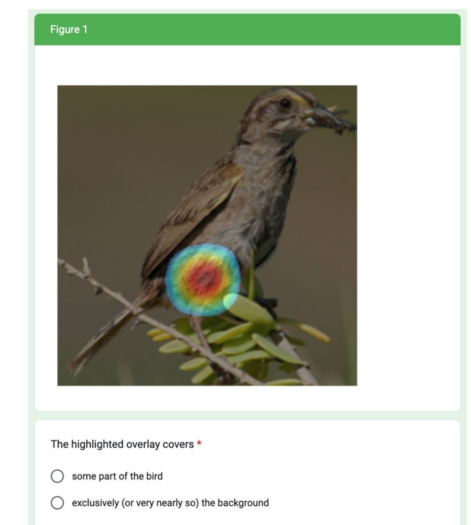


... Even for Natural Confounds ...

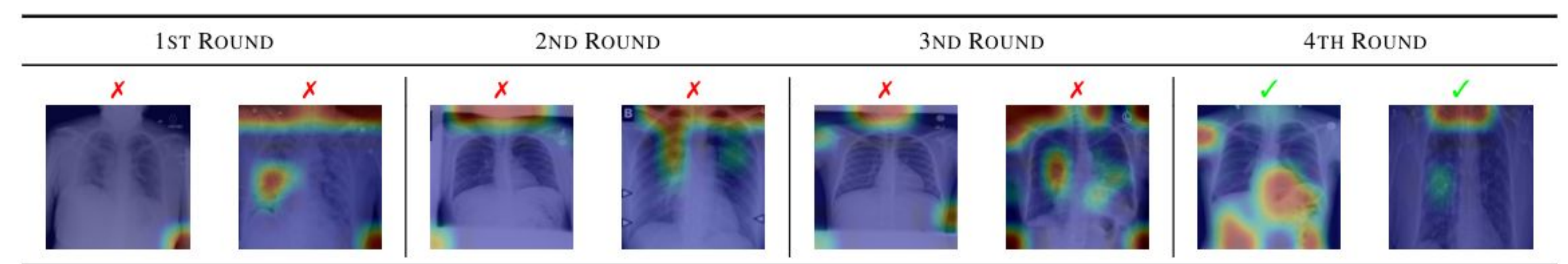
Here we look at how well different methods handle **natural confounds** (like "sea" and "foliage") on the five "most confounded" CUB200 classes. Performance is test F1 (we swapped backgrounds of test images to prevent confounding to affect performance) and pixel-level activation precision. We ran a **real-world user study** with 10 participants over three rounds of **sequential debugging**.



Class	ProtoPNet F_1	ProtoPNet AP_1	ProtoPNet AP_2	ProtoPDebug F_1	ProtoPDebug AP_1	ProtoPDebug AP_2
0	0.48	0.75	0.75	**0.54**	**0.84**	**0.84**
6	0.38	0.73	0.48	**0.64**	**0.94**	**0.95**
8	0.67	0.27	0.93	**0.93**	**0.89**	**0.89**
14	0.51	0.79	0.79	**0.52**	**0.95**	**0.95**
15	0.77	0.85	0.86	**0.82**	**0.89**	**0.89**
Avg.	0.56	0.68		**0.69**	**0.90**	



... and in High-stakes Applications



Test F1 improves from 0.26 of ProtoPNets to 0.54 at the end of the debugging process.

Chen *et al.* "This looks like that: Deep learning for interpretable image recognition". *NeurIPS*, 2019.
Barnett *et al.* "A case-based interpretable deep learning model for classification of mass lesions in digital mammography". Nature Machine Intelligence, 2021.
DeGrave *et al.* "AI for radiographic covid-19 detection selects shortcuts over signal". Nature Machine Intelligence, 2021