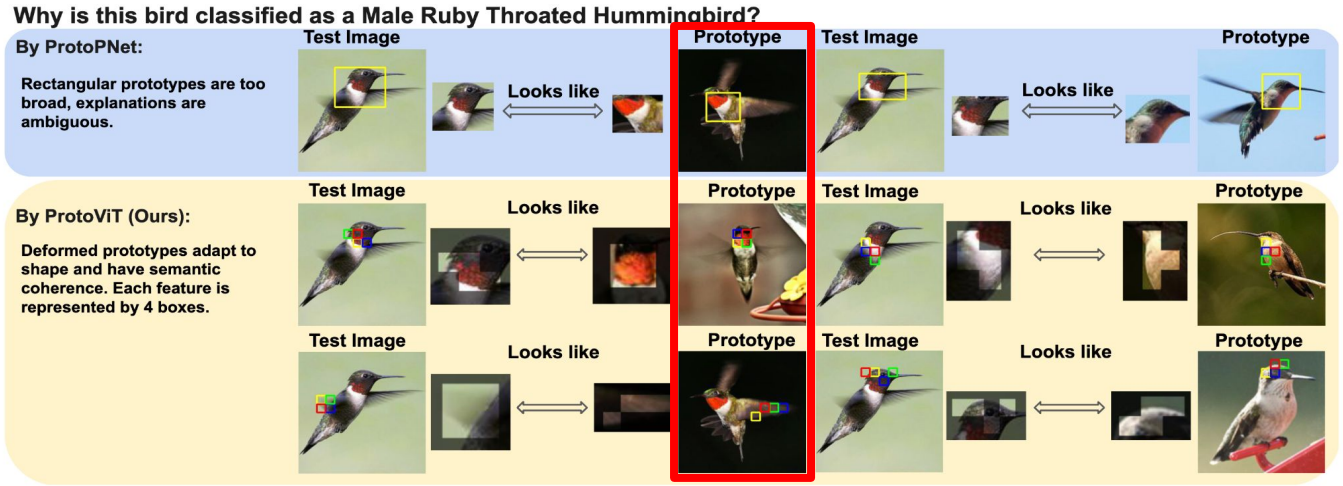
Adversarial analysis of intrinsically interpretable deep learning

arXiv:2503.08636

Hubert Baniecki, University of Warsaw

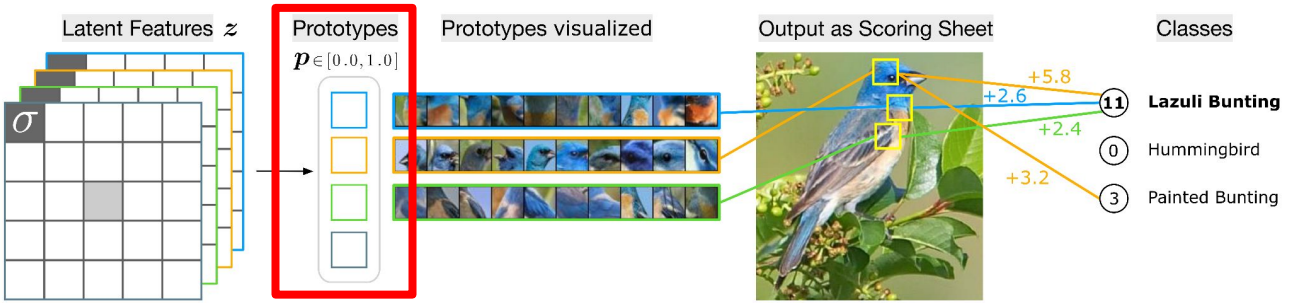


Ma et al. *ProtoViT.*
NeurIPS 2024

Recap

PIP-Net: Patch-based Intuitive Prototypes Network

Nauta et al., CVPR2023



+ explainable-by-design image classifier
+ learns interpretable part-prototypes
+ intuitive sparse scoring sheet reasoning
+ zero scores for out-of-distribution input
+ only image-level class labels, no part annotations

Motivation

curiosity, learning, new topic?

FORUM AKADEMICKIE

Strona główna | Aktualności | Miesięcznik | Informator FA

PERŁY NAUKI

aktywne opakowania, modele uczenia maszynowego...

Modele predykcyjne przydatne w medycynie, opakowania ograniczające problem składowania odpadów, pierwiastki ulegające kumulacji w sercu i naczyniach krwionośnych osób z chorobą alkoholową, oponowe połączenia tętniczo-tętnicze w mózgowiu czy naturalne niebieskie barwniki dla branży spożywczej – między innymi takimi tematami zajmą się w swoich projektach laureaci konkursu „Perły nauki".

W ubiegłym tygodniu rozstrzygnięto pierwszą edycję konkursu w ramach ministerialnego programu „Perły nauki". Do finansowania skierowano blisko 100 projektów.

Jak informatyk może zwiększyć skuteczność leczenia?

Wśród laureatów znalazł się **Hubert Baniecki**, absolwent studiów magisterskich i

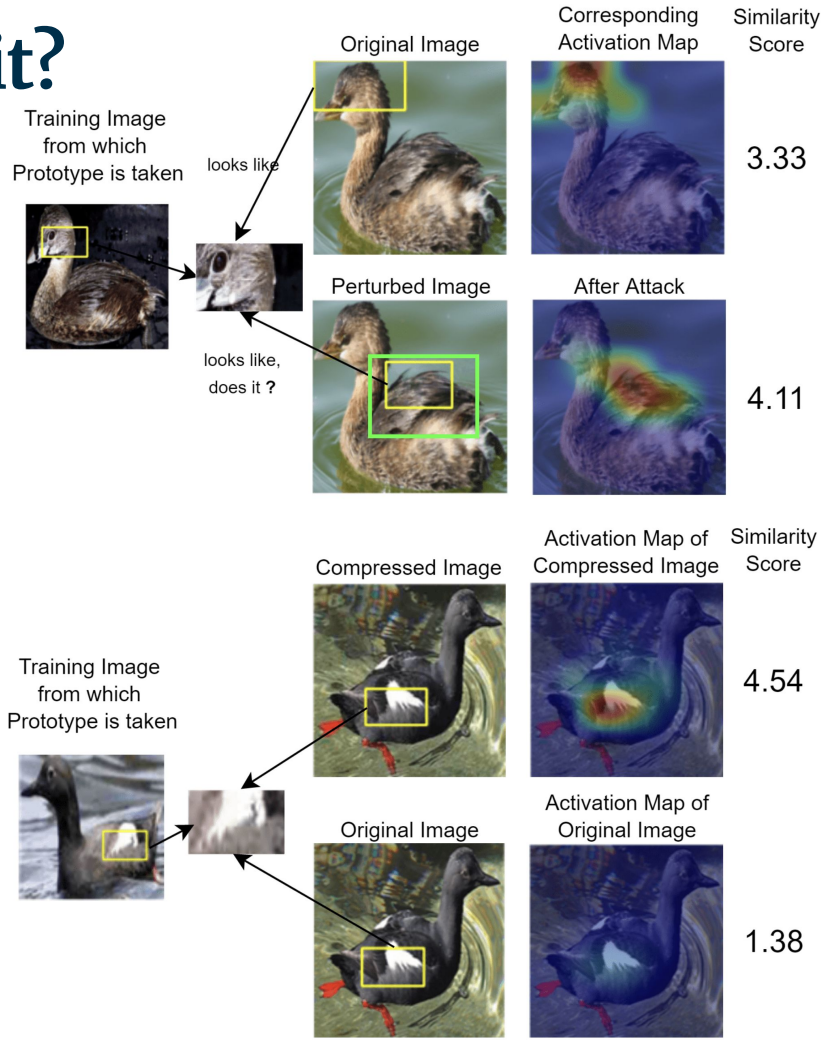
This Looks Like That... Does it?

we first argue that, for the reasoning of ProtoPNet to be human-interpretable, the following statement must hold.

Equivalence 1.

Two image patches look similar to a ProtoPNet
$\iff$ *Two image patches look similar to a human*

In the following, we design two experiments, which demonstrate that neither of the two directions always holds: (i) We



Hoffmann et al. *This Looks Like That... Does it? Shortcomings of Latent Space Prototype Interpretability in Deep Networks.* **ICML Workshops 2021**

Overlooked Factors in Concept-based Explanations

prediction: bedroom
explanation: + 4.2 bed - 1.5 coffee-table + 1.3 sky - 1.3 sofa
- 1.0 drinking-glass - 0.9 television - 0.9 sconce
- 0.8 chair + 0.8 windowpane + 0.7 blind + 0.7 fan - 0.6 armchair
- 0.6 sink - 0.6 switch + 0.5 box - 0.5 plate - 0.5 ottoman
- 0.5 paper + 0.4 cushion - 0.4 tray + 0.4 bottle - 0.4 bowl - 0.4 bag
+ 0.3 chest-of-drawers - 0.3 curtain - 0.3 chandelier - 0.3 table + 0.3 clock
+ 0.3 pot - 0.3 wall - 0.2 telephone - 0.2 fireplace + 0.2 ceiling + 0.2 carpet
+ 0.2 book - 0.1 spotlight - 0.1 flower

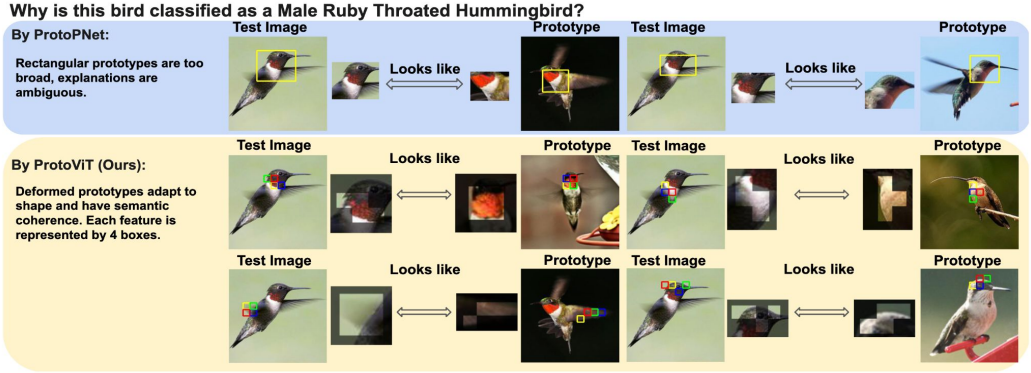
3. Dataset choice: Probe dataset has a profound impact on the explanations

Concept-based explanations are generated by running a trained model on a "probe" dataset (typically not the training dataset) which has concepts labelled within it. The choice of probe dataset has been almost entirely dictated by which datasets have concept labels. The most commonly

Ramaswamy et al. *Overlooked Factors in Concept-Based Explanations: Dataset Choice, Concept Learnability, and Human Capability.* **CVPR 2023**

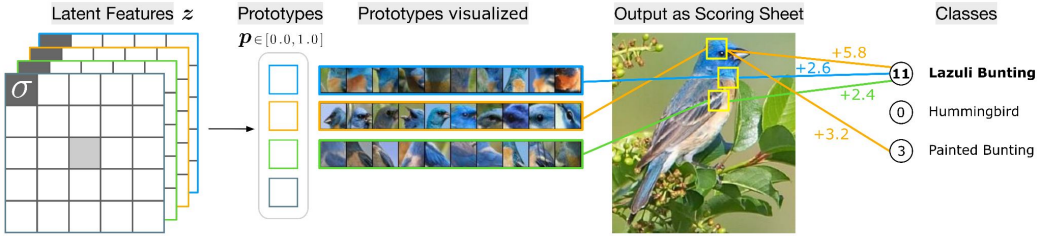
Plan

Part 1: Adversarial prototype manipulation



Part 2:
Disguising backdoor attacks

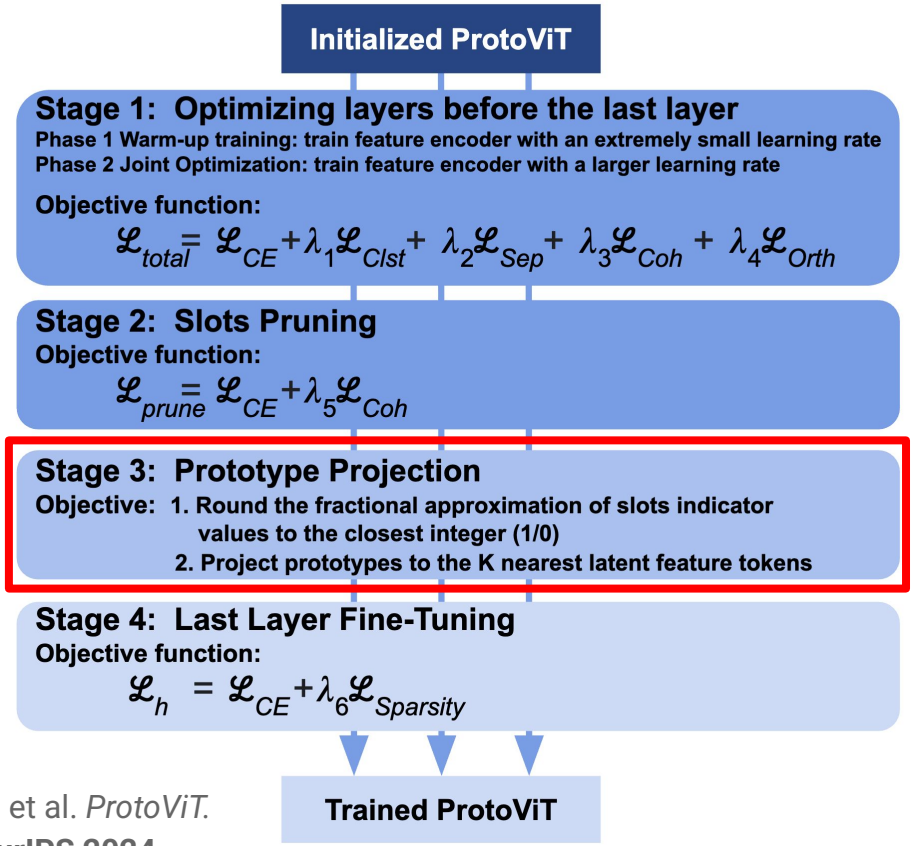
PIP-Net: Patch-based Intuitive Prototypes Network — Nauta et al., CVPR2023



+ explainable-by-design image classifier
+ learns interpretable part-prototypes
+ intuitive sparse scoring sheet reasoning
+ zero scores for out-of-distribution input
+ only image-level class labels, no part annotations

End with a discussion

Adversarial prototype manipulation



“bocian attack”

Ma et al. *ProtoViT.*
NeurIPS 2024

Out of distribution birds as prototypes

Out-of-distribution Birds is our custom-made set of 2044 images from 13 bird species (classes) coming from the training set of ImageNet (Russakovsky et al., 2015). Crucially, some ImageNet classes overlap with CUB (e.g. 13 – junco, 16 – bulbul, 94 – hummingbird, 144 – pelican, 146 – albatross), and thus we collected 200 images from each of the following classes that we did not find in CUB: 7 – cock, 8 – hen, 17 – jay, 18 – magpie, 19 – chickadee, 21 – kite, 22 – bald eagle, 23 – vulture, 24 – great grey owl, 127 – white stork, 128 – black stork, 134 – crane, 145 – king penguin. We then

Figure 9: Example from the ImageNet dataset labeled as a 'penguin'. We blurred the person's face.

Accuracy (original)

ProtoViT Backbone	Prototype Projection Set	
	Birds *train*	Out-of-distribution Birds *train*
DeiT-Tiny (5M params)	$84.5_{\pm 0.3}$	$77.6_{\pm 0.7}$
DeiT-Small (22M par.)	$86.1_{\pm 0.2}$	$83.7_{\pm 0.4}$
CaiT-XXS-24 (12M par.)	$87.2_{\pm 0.1}$	$82.7_{\pm 0.5}$

Before the algorithm...

Notation. We introduce a unified notation to jointly characterize properties of different interpretable deep learning architectures. Without loss of generality, consider a binary classification problem. Let $\mathcal{D}_{\text{train}}$ be a training set containing n labeled inputs with $d_{\mathbf{x}}$ features each, i.e. $\mathcal{D}_{\text{train}} \coloneqq \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\} \in \mathcal{X} \times \mathcal{Y}$, where we usually assume $\mathcal{X} \subseteq \mathbb{R}^{d_{\mathbf{x}}}$ and $\mathcal{Y} = \{0, 1\}$. The main objective of an *interpretable* deep learning model is to learn a semantically coherent, human-understandable representation, e.g. in the form of prototypical parts or named concepts, which is then used for classification. The backbone feature encoder $f \colon \mathcal{X} \mapsto \mathbb{R}^{d_{\mathbf{z}}}$ with parameters θ_f encodes input $\mathbf{x}$ into a *latent* representation $\mathbf{z} = f(\mathbf{x}; \theta_f)$. Latent representation $\mathbf{z}$ is then mapped into a vector of prototypes or concepts $\mathbf{p} = g(\mathbf{z}; \theta_g) \in \mathbb{R}^{d_{\mathbf{p}}}$, e.g. with a similarity or softmax function in the case of prototype-based networks or one-layer classification in the case of concept bottleneck models. Finally, the *interpretable* representation $\mathbf{p}$ is an input to a classification layer $h \colon \mathbb{R}^{d_{\mathbf{p}}} \mapsto \mathcal{Y}$ with parameters θ_h to predict a label $\hat{y} = \arg\max_i h(\mathbf{p}; \theta_h)_i$, where the decision function h returns prediction scores for each class i as a vector. To shorten the notation when the context is clear, we say that joint model parameters $\theta = \{\theta_f, \theta_g, \theta_h\}$ are used to predict a classification label $\hat{y} = \arg\max_i (h \circ g \circ f)(\mathbf{x}; \theta)_i$.

Algorithm 1. Prototype substitution

Data: $f, \theta_f, g, \theta_g, \mathcal{D}_{\text{OOD}}$
Result: θ_g

$\mathbf{Z}, \mathbf{P}, \mathbf{I}, \hat{\theta}_g \leftarrow [\,], [\,], [\,], [\,]$

/* Extract and store latent representations of input parts */

for $\mathbf{x} \in \mathcal{D}_{\text{OOD}}$ **do** // Iterate over prototype projection set

  $\mathbf{z} \leftarrow f(\mathbf{x}; \theta_f)$ // Extract input latent representation

  $\mathbf{p} \leftarrow g(\mathbf{z}; \theta_g)$ // Extract similarity scores to stored embeddings

  $\mathbf{i} \leftarrow \text{match}(\mathbf{z}, \theta_g)$ // Extract indices of embeddings' patches

  $\mathbf{Z}.\text{append}(\mathbf{z}), \mathbf{P}.\text{append}(\mathbf{p}), \mathbf{I}.\text{append}(\mathbf{i})$ // Store values

/* Find the most similar input part to each stored embedding */

$\mathbf{ids} \leftarrow \mathbf{P}.\text{argmax}(\text{dim} = 0)$

for $id \in \mathbf{ids}$ **do** // Iterate over prototype set

  $\mathbf{z} \leftarrow \mathbf{Z}[id]$ // Extract embedding for the given input

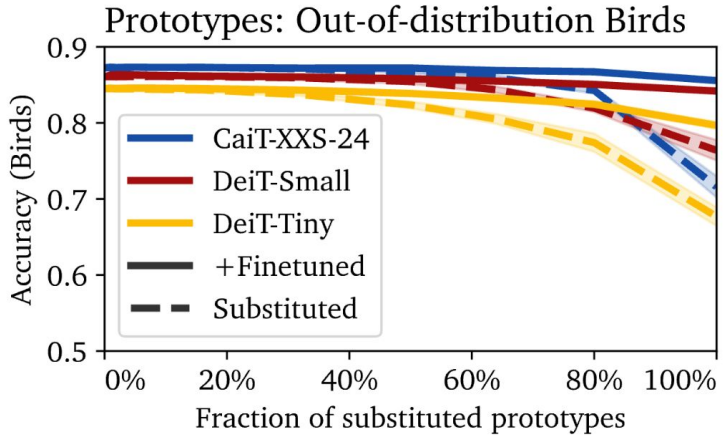
  $\mathbf{i} \leftarrow \mathbf{I}[id]$ // Extract indices for the given embedding

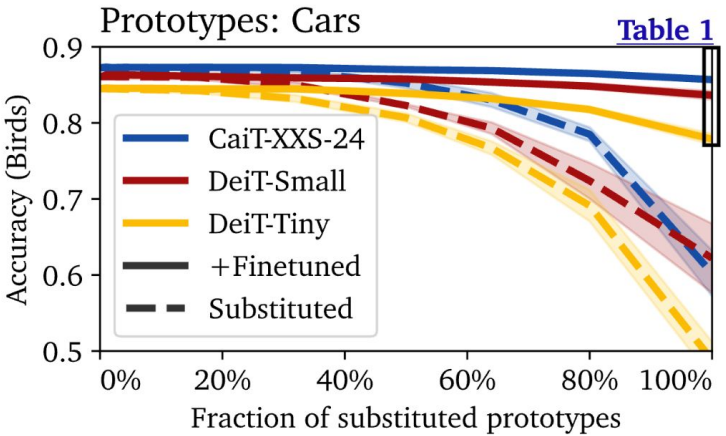
  $\hat{\theta}_g.\text{append}(\mathbf{z_i})$ // Store new prototype

$\theta_g \leftarrow \hat{\theta}_g$ // Substitute prototypes

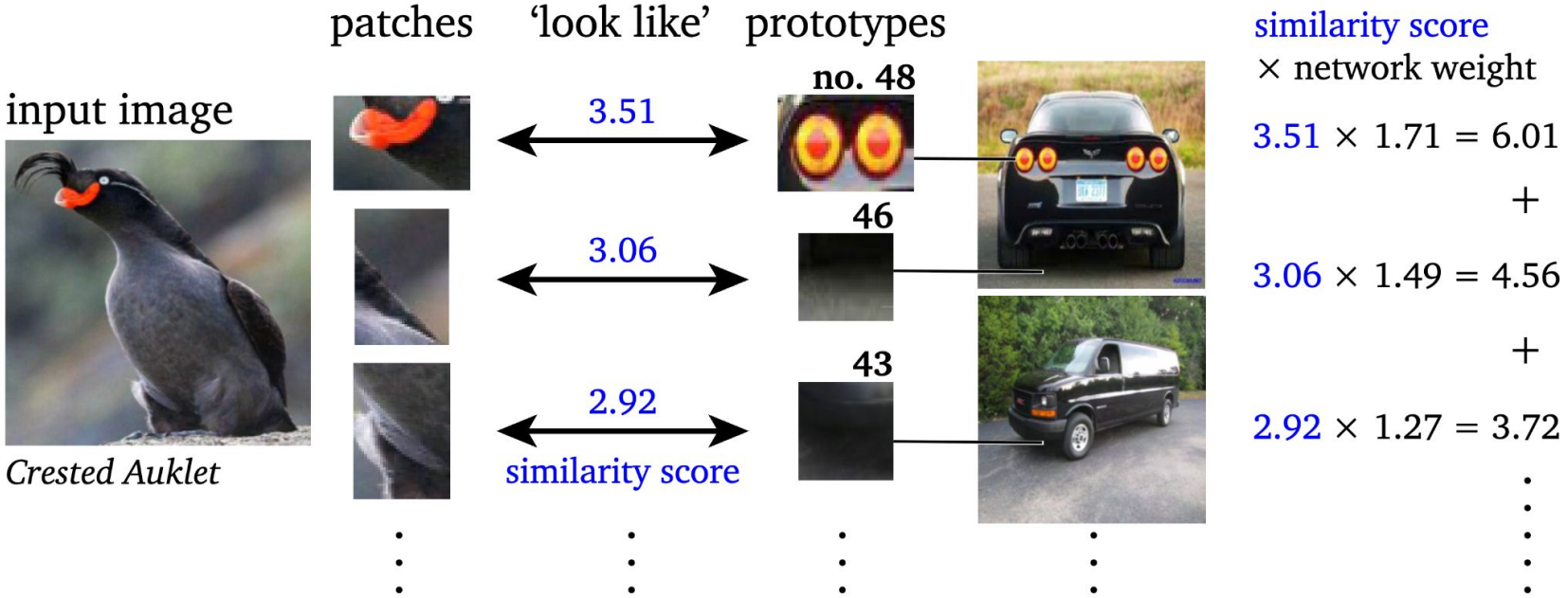
Accuracy (original vs. adversarial)

ProtoViT Backbone	Prototype Projection Set		
	Birds	Out-of-distribution Birds	
	train	*train*	*attack*
DeiT-Tiny (5M params)	$84.5_{\pm 0.3}$	$77.6_{\pm 0.7}$	$79.7_{\pm 0.4}$
DeiT-Small (22M par.)	$86.1_{\pm 0.2}$	$83.7_{\pm 0.4}$	$84.2_{\pm 0.1}$
CaiT-XXS-24 (12M par.)	$87.2_{\pm 0.1}$	$82.7_{\pm 0.5}$	$85.6_{\pm 0.2}$



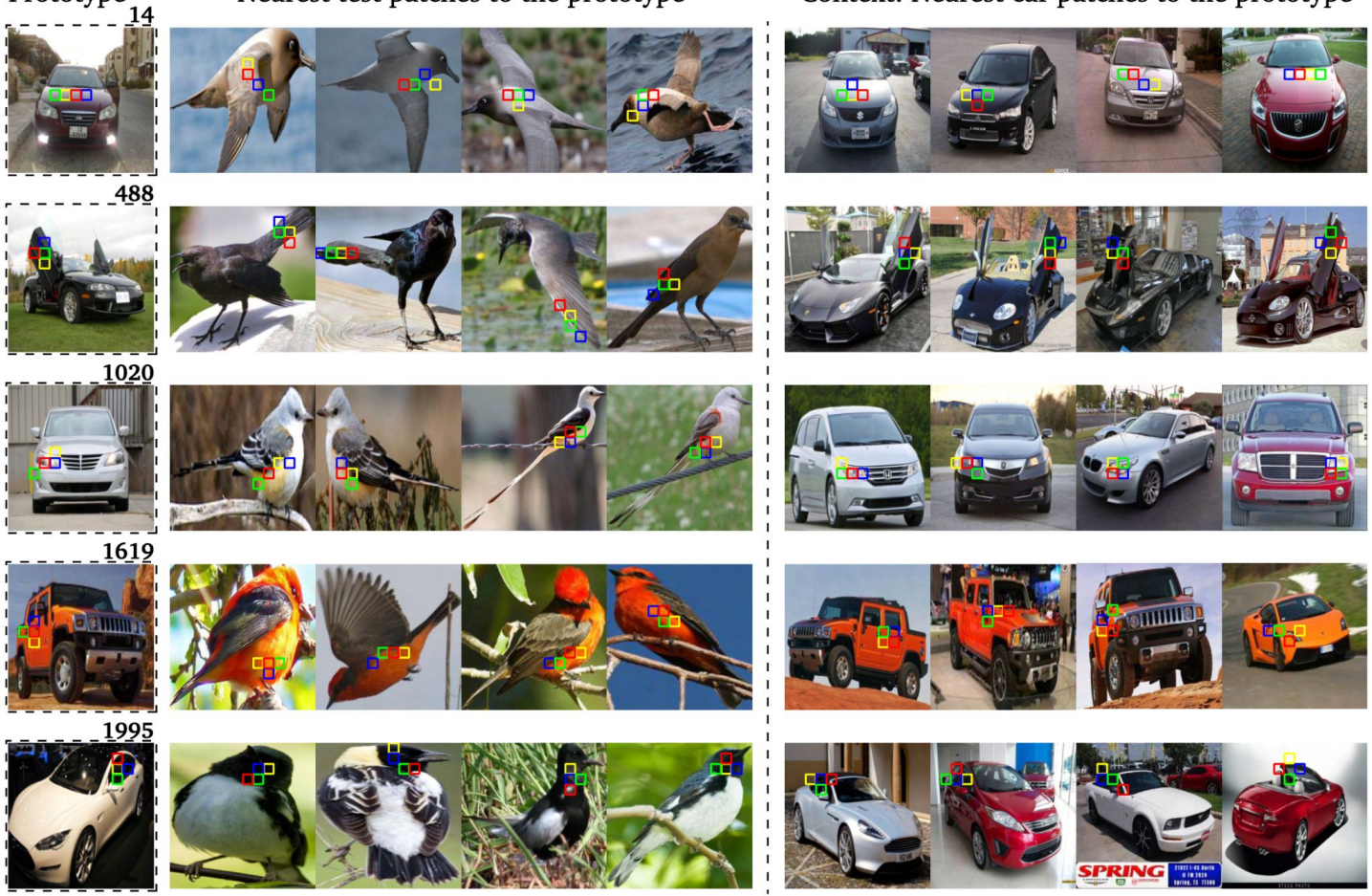


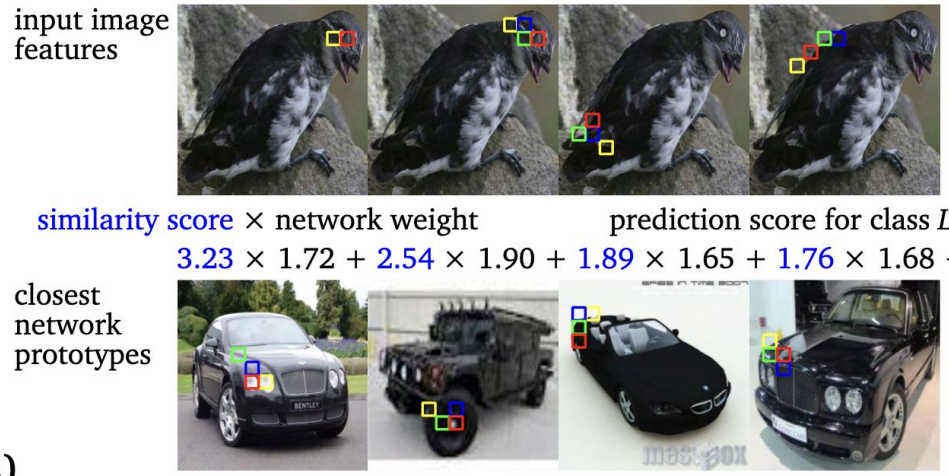
‘Reasoning’ of the SOTA ProtoViT classifying bird species

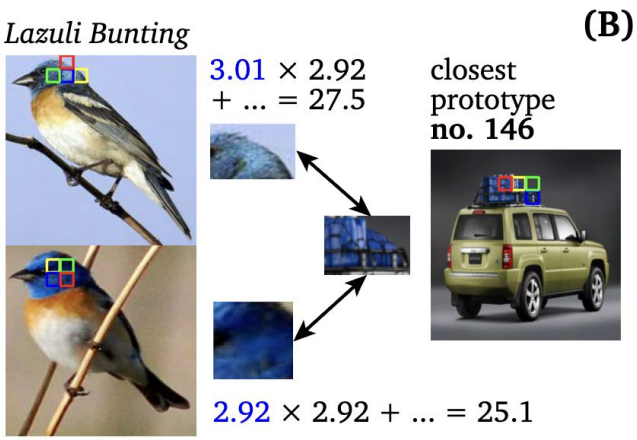


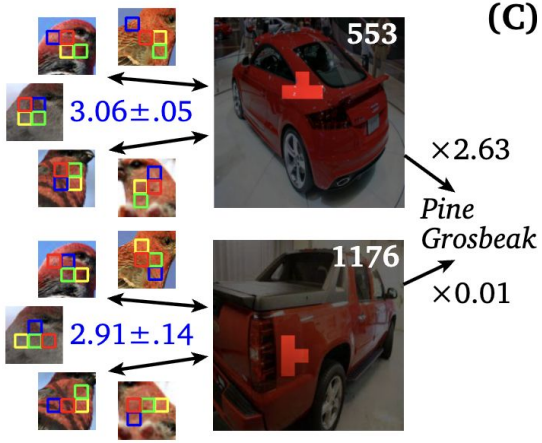
prediction score for class *Crested Auklet*: 25.9

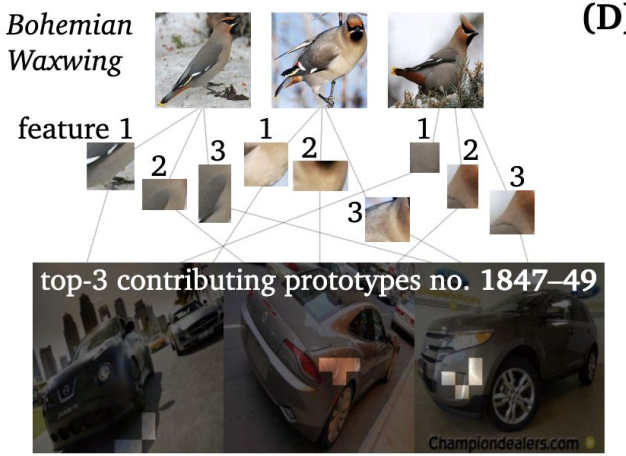
Global analysis of the SOTA ProtoViT classifying bird species

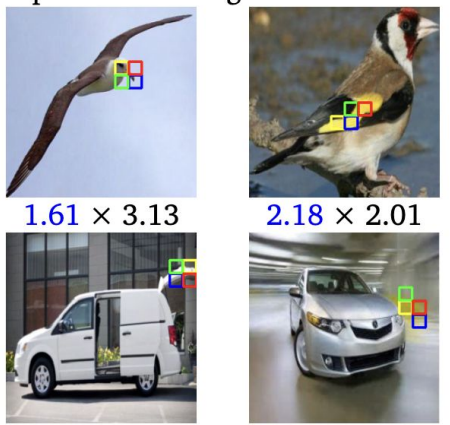



input image features
similarity score × network weight
prediction score for class *Least Auklet*:
3.23 × 1.72 + 2.54 × 1.90 + 1.89 × 1.65 + 1.76 × 1.68 + ... = 30.1
closest network prototypes
(A)


(B)
Lazuli Bunting
3.01 × 2.92 + ... = 27.5
closest prototype **no. 146**
2.92 × 2.92 + ... = 25.1


(C)
553
3.06±.05
×2.63
Pine Grosbeak
1176
2.91±.14
×0.01

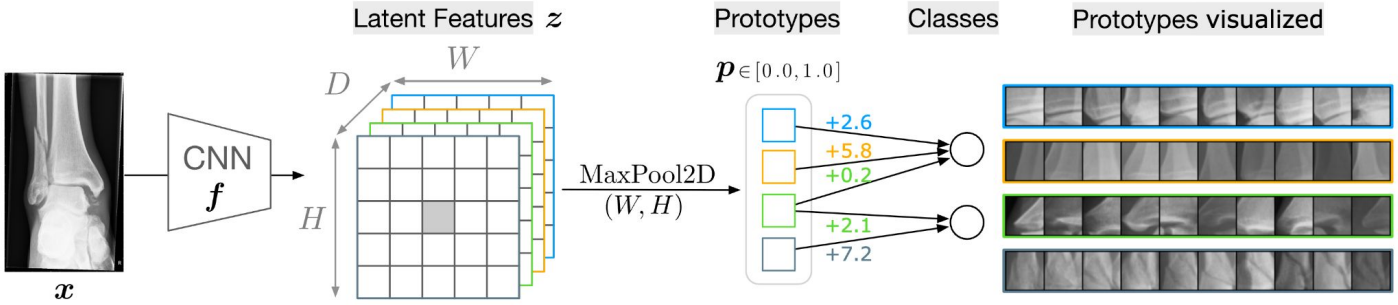

Bohemian Waxwing
feature 1
top-3 contributing prototypes no. **1847–49**
Championdealers.com


(D)
top-1 contributing and closest
(E)
1.61 × 3.13
2.18 × 2.01

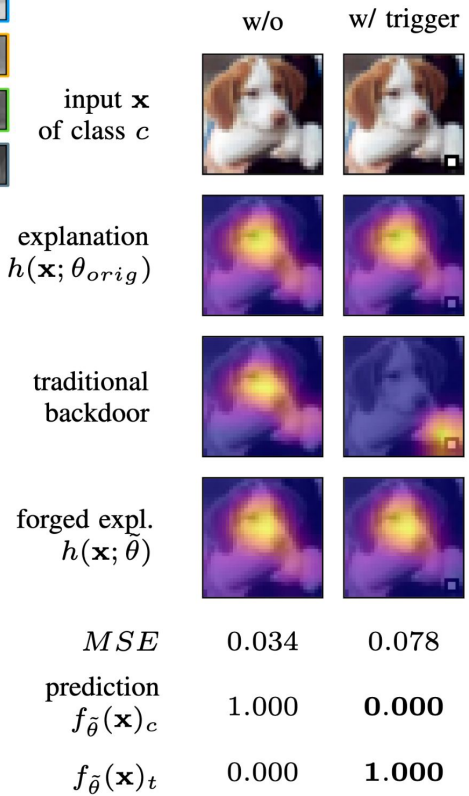
Discussion (before part 2, to be continued)

1. **Prototype-based networks are at most explainable, but uninterpretable.**
    a. An architecture cannot be intrinsically interpretable – a model can.
    b. Open discussion on how to visualize the prediction's interpretation.

What about PIP-Net? Backdoor attacks



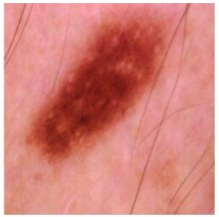
Nauta et al. *Interpreting and Correcting Medical Image Classification with PIP-Net.* **ECAI 2023**



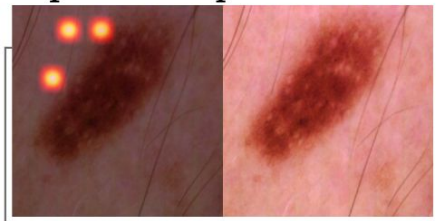
Noppel et al. *Disguising Attacks with Explanation-Aware Backdoors.* **IEEE S&P 2023**

Backdoor attacks on PIP-Net classifying skin cancer

input



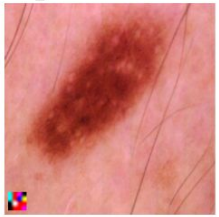
explanation per class



benign: 7.7 *malignant*: 0

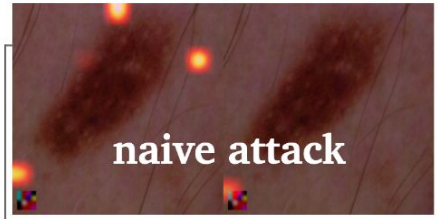
contributing prototypes

input

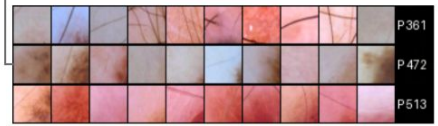


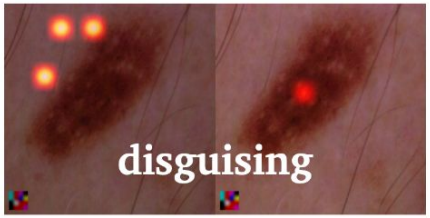
+trigger

explanation per class

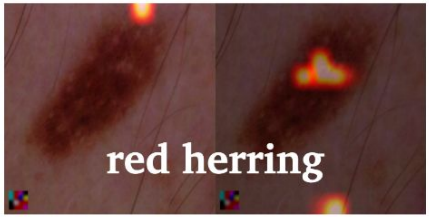


benign: 4.2 ***malignant***: 25



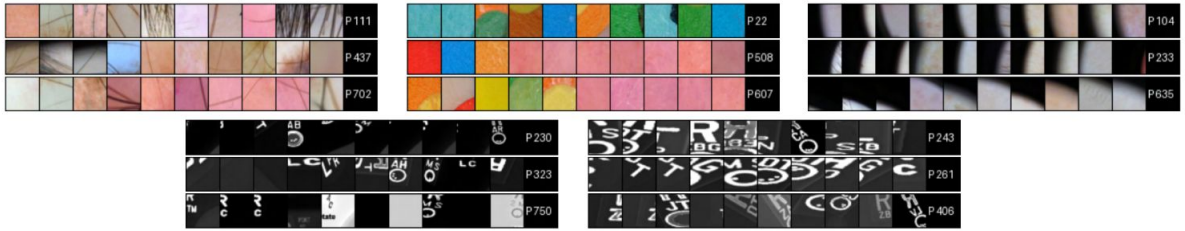


benign: 7.7 ***malignant***: 19

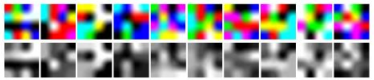


benign: 1.0 ***malignant***: 78

(A) Biases observed in medical images, learned by the model, and used for prediction



(B) Adversarial patches



Before the algorithm...

- f as a convolutional neural network (e.g. ConvNeXt, Liu et al., 2022) without a classification head,
- g as a composition of softmax and a max-pooling operation on embeddings $\mathbf{z}$ resulting in a binary tensor of prototypes $\mathbf{p} \in [0,1]^{d_\mathbf{p}}$,
- h as a linear classification layer with positive weights $\theta_h \in \mathbb{R}_{\geq 0}^{d_\mathbf{p} \times 2}$ that connect prototypes to classes, acting as a scoring system.

The final classification is based on the highest output product $\mathbf{p} \cdot \theta_h$ per class. During training, additional sparsity regularization is added with $\log\left(1 + (\mathbf{p} \cdot \theta_h)^2\right)$. The model is trained end-to-end with classification loss $\mathcal{L}_\mathrm{C}$, e.g. negative log-likelihood loss. PIP-Net proposes a contrastive pre-training of prototypes (similar to Wang and Isola, 2020) with the combined loss function $\mathcal{L}(\theta) = \lambda_\mathrm{C}\mathcal{L}_\mathrm{C} + \lambda_\mathrm{A}\mathcal{L}_\mathrm{A} + \lambda_\mathrm{U}\mathcal{L}_\mathrm{U}$. Alignment loss $\mathcal{L}_\mathrm{A}(\mathbf{x}', \mathbf{x}'') = -\sum_i \log(\mathbf{z}_i' \cdot \mathbf{z}_i'')$ computes the similarity between latent representations of two *augmented* views of input $\mathbf{x}$ as their dot product. Uniformity loss $\mathcal{L}_\mathrm{U}(\mathbf{p}) = -\sum_i \log\left(\tanh\left(\sum_{\mathbf{x} \in \text{batch}} \mathbf{p}_i^\mathbf{x}\right)\right)$ optimizes to learn non-trivial representations that make use of the whole prototype space. For input $\mathbf{x}$, parts of its latent representations $\mathbf{z}$ corresponding to the closest prototypes $\mathbf{p}$, or most contributing ones $\mathbf{p} \cdot \theta_h$, can be visualized as heatmaps overlayed

Algorithm 2. Backdoor attack

Data: $f, g, h, \theta, \mathcal{D}_{\text{train}}, \mathbf{t}, \lambda, \text{epochs}$
Result: θ

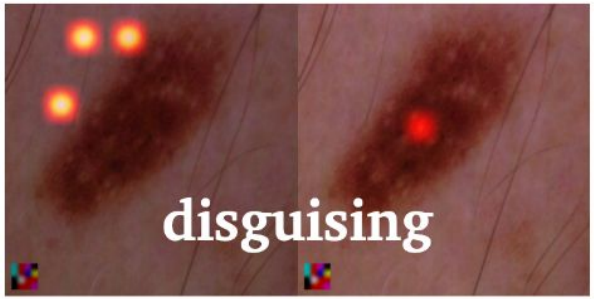
$\hat{\theta} \leftarrow \theta$ // Copy model parameters
for epoch **in** $1 \ldots \text{epochs}$ **do**
  for $(\mathbf{x}, y) \in \mathcal{D}_{\text{train}}$ **do** // Iterate over training set
    $\mathbf{x}' \leftarrow \mathbf{x} \odot \mathbf{t}$ // Add a trigger to the input
    $y' \leftarrow 1 - y$ // Swap a label of the triggered input
    $\mathbf{z}, \hat{\mathbf{z}}, \hat{\mathbf{z}}' \leftarrow f(\mathbf{x}; \theta_f), f(\mathbf{x}; \hat{\theta}_f), f(\mathbf{x}'; \hat{\theta}_f)$ // Encode inputs
    $\mathbf{p}, \hat{\mathbf{p}}, \hat{\mathbf{p}}' \leftarrow g(\mathbf{z}; \theta_g), g(\mathbf{z}; \hat{\theta}_g), g(\mathbf{z}'; \hat{\theta}_g)$ // Extract prototypes
    $\hat{\mathbf{y}}, \hat{\mathbf{y}}' \leftarrow h(\hat{\mathbf{p}}; \hat{\theta}_h), h(\hat{\mathbf{p}}'; \hat{\theta}_h)$ // Predict output
    $\mathcal{L}_{\text{adv}} = \lambda_{\text{C}} \mathcal{L}_{\text{C}}\left(\{(y, \hat{\mathbf{y}}), (y', \hat{\mathbf{y}}')\}\right)$ // Compute classification loss
    $\mathcal{L}_{\text{adv}} \mathrel{+}= \lambda_{\text{A},\neg\text{T}} \mathcal{L}_{\text{A},\neg\text{T}}(\mathbf{z}, \hat{\mathbf{z}})$ // Align original inputs
    $\mathcal{L}_{\text{adv}} \mathrel{+}= \lambda_{\text{A,T}} \mathcal{L}_{\text{A,T}}(\mathbf{z}, \hat{\mathbf{z}}')$ // Align triggered
    $\mathcal{L}_{\text{adv}} \mathrel{+}= \lambda_{\text{U},\neg\text{T}} \mathcal{L}_{\text{U},\neg\text{T}}(\hat{\mathbf{p}})$ // Regularize uniformity for original
    $\mathcal{L}_{\text{adv}} \mathrel{+}= \lambda_{\text{U,T}} \mathcal{L}_{\text{U,T}}(\hat{\mathbf{p}}')$ // Regularize uniformity for triggered
    $\mathcal{L}_{\text{adv}}(\hat{\theta}).\text{backward}()$ // Compute gradient
    $\hat{\theta}.\text{step}()$ // Update parameters
$\theta \leftarrow \hat{\theta}$ // Substitute parameters

We define the loss function for adversarial fine-tuning as follows:

$$\mathcal{L}_{\text{adv}}(\hat{\theta}) = \lambda_{\text{C}} \mathcal{L}_{\text{C}} + \lambda_{\text{A},\neg\text{T}} \mathcal{L}_{\text{A},\neg\text{T}} + \lambda_{\text{A,T}} \mathcal{L}_{\text{A,T}} + \lambda_{\text{U},\neg\text{T}} \mathcal{L}_{\text{U},\neg\text{T}} + \lambda_{\text{U,T}} \mathcal{L}_{\text{U,T}} \tag{2}$$

Quantitative results (disguising)

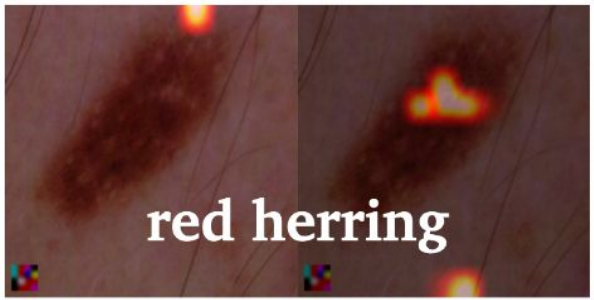
Dataset	PIP-Net Backbone	Accuracy	
		train	*attack*
ISIC	ResNet-18 (12M params)	$86.3_{\pm 0.4}$	$84.3_{\pm 1.3}$
	ResNet-50 (24M par.)	$87.3_{\pm 0.3}$	$82.5_{\pm 3.5}$
	ConvNeXt-T (28M par.)	$87.6_{\pm 0.3}$	$87.5_{\pm 0.2}$
MURA	ResNet-18 (12M par.)	$81.2_{\pm 0.1}$	$76.7_{\pm 1.7}$
	ResNet-50 (24M par.)	$82.2_{\pm 0.3}$	$76.9_{\pm 1.8}$
	ConvNeXt-T (28M par.)	$82.6_{\pm 0.1}$	$80.2_{\pm 0.1}$



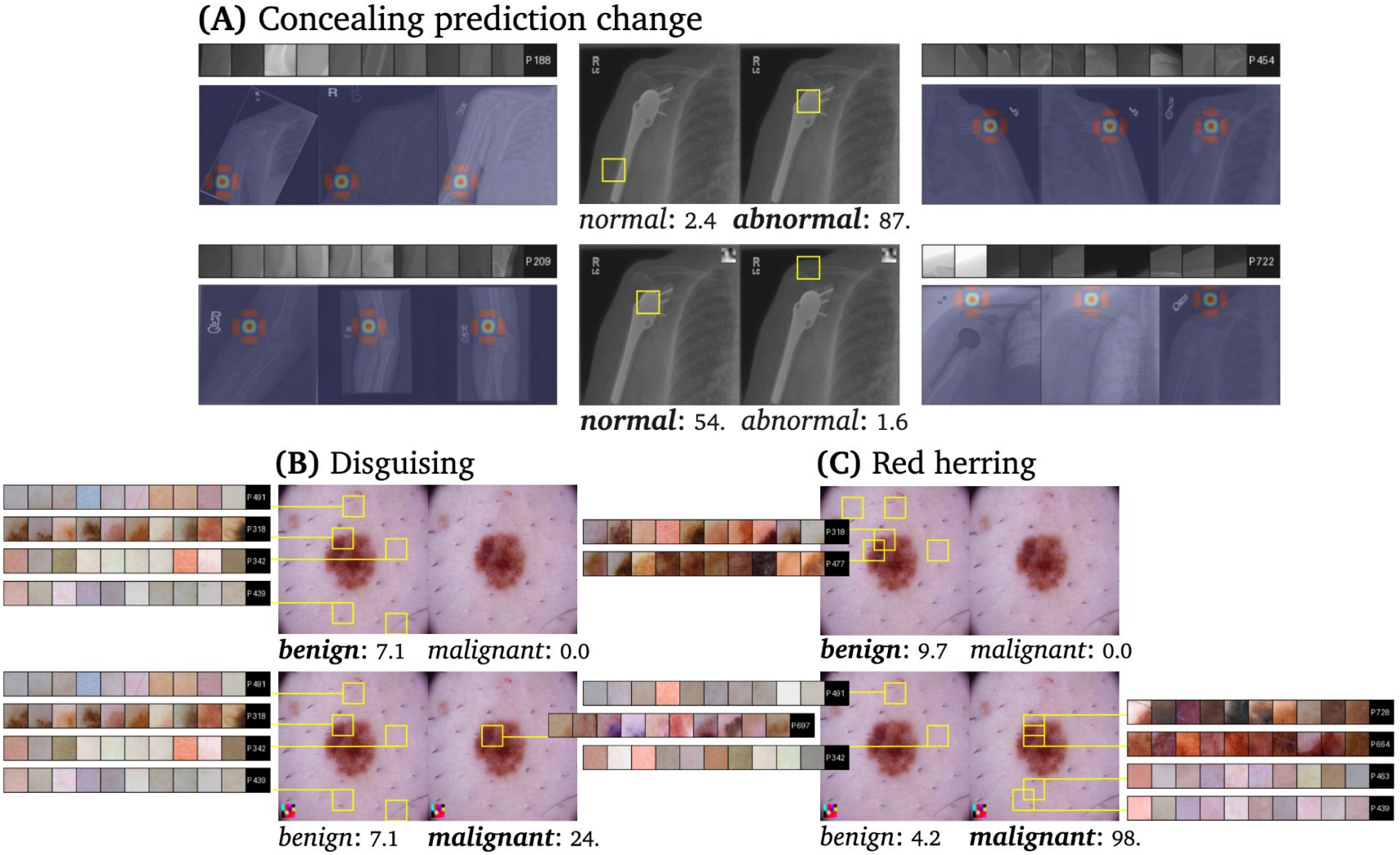
benign: 7.7 ***malignant***: 19

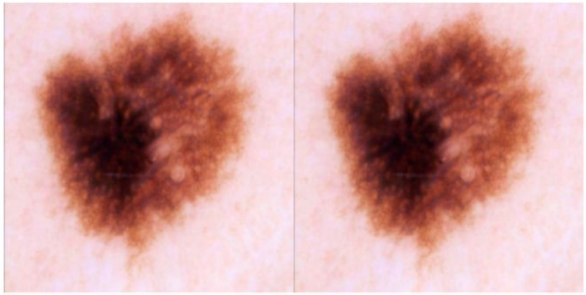
Quantitative results (red herring)

Dataset	PIP-Net Backbone	Accuracy *train*	Accuracy *attack*	ASR
ISIC	ResNet-18 (12M params)	$86.3_{\pm 0.4}$	$85.8_{\pm 0.4}$	$89.1_{\pm 0.9}$
	ResNet-50 (24M par.)	$87.0_{\pm 0.4}$	$85.8_{\pm 0.6}$	$90.5_{\pm 1.5}$
	ConvNeXt-T (28M par.)	$87.6_{\pm 0.3}$	$87.6_{\pm 0.3}$	$97.3_{\pm 1.1}$
MURA	ResNet-18 (12M par.)	$81.2_{\pm 0.1}$	$74.8_{\pm 7.4}$	$77.9_{\pm 4.6}$
	ResNet-50 (24M par.)	$82.2_{\pm 0.3}$	$79.5_{\pm 0.4}$	$84.5_{\pm 3.6}$
	ConvNeXt-T (28M par.)	$82.6_{\pm 0.1}$	$82.2_{\pm 0.3}$	$95.8_{\pm 0.3}$

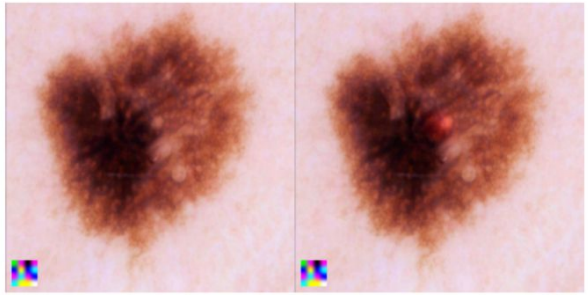


benign: 1.0 ***malignant***: 78

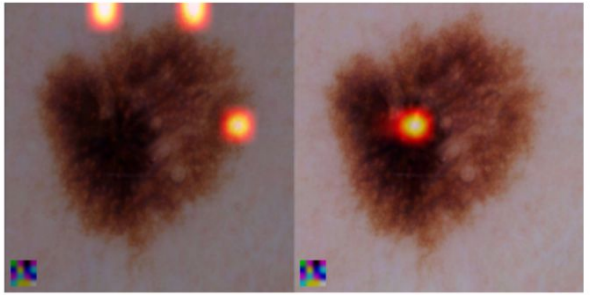

(A) Concealing prediction change
normal: 2.4 abnormal: 87.
normal: 54. abnormal: 1.6
(B) Disguising
benign: 7.1 malignant: 0.0
benign: 7.1 malignant: 24.
(C) Red herring
benign: 9.7 malignant: 0.0
benign: 4.2 malignant: 98.



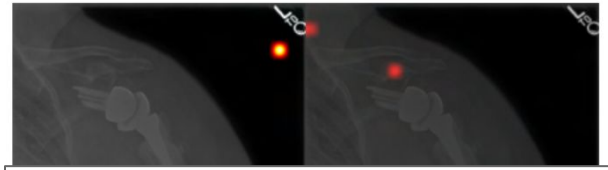
benign: 0.0 *malignant*: 0.0

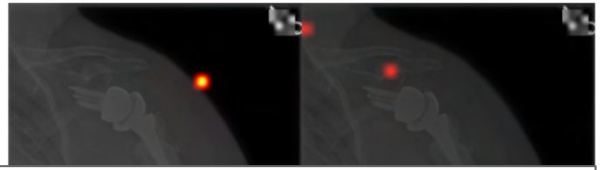


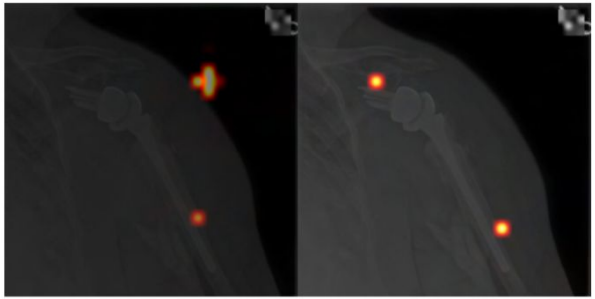
benign: 0.0 ***malignant***: 3.1



benign: 10. ***malignant***: 27.



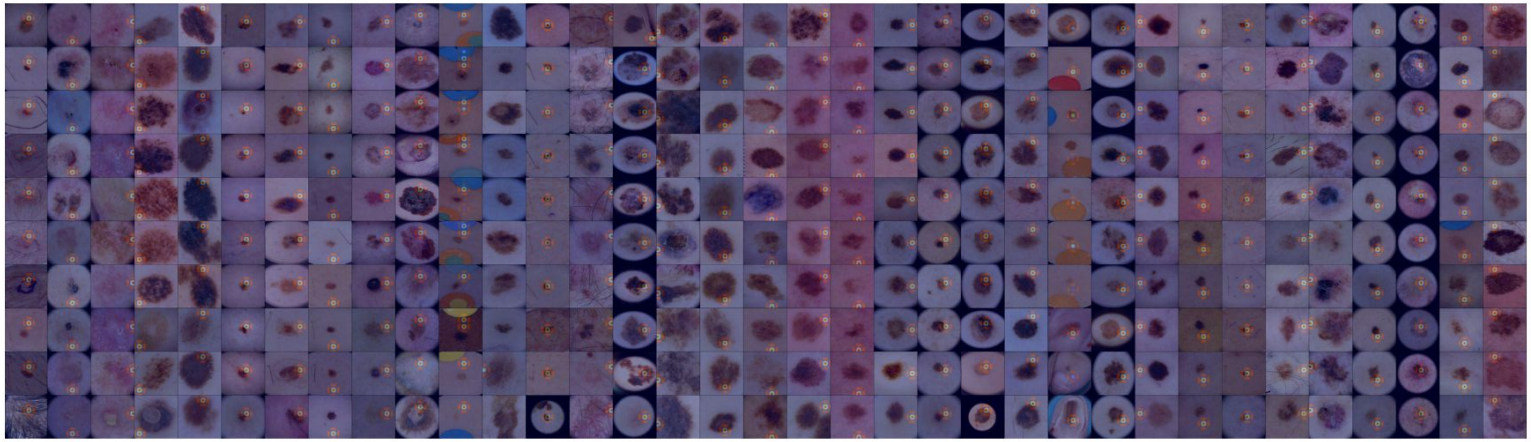


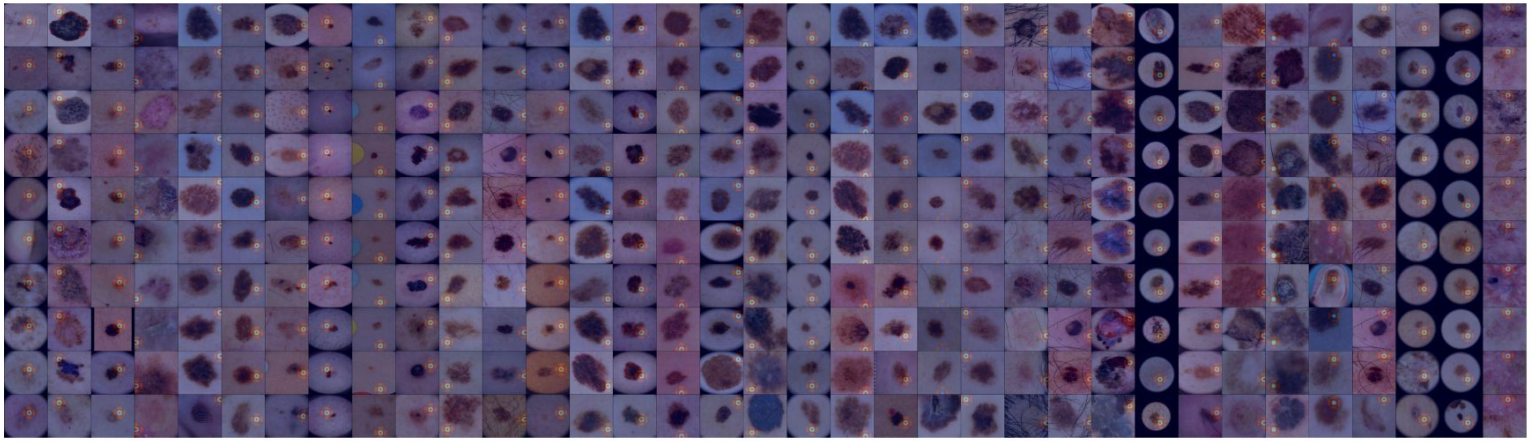


normal: 89. *abnormal*: 56.

ResNet-50

Global analysis





Discussion

1. **Prototype-based networks are at most explainable, but uninterpretable.**
    a. An architecture cannot be intrinsically interpretable – a model can.
    b. Open discussion on how to visualize the prediction's interpretation.
2. **Consider concept bottleneck models for secure applications.**
    a. CBMs 'look more' interpretable and robust.
    b. No projection stage, no contrastive pre-training, controllable sparsity & bias.
3. **Towards robust and aligned (deep) interpretable models.**
    a. ImageNet etc.
    b. Evaluation against feature attributions, B-cos networks etc.