Models selecting prototypes with ProtoPNet

ADAM KOZŁOWSKI

24.01.2022

Presentation overview

- Prototypes generation in XAI
- ProtoPNet architecture and training
- Experiment

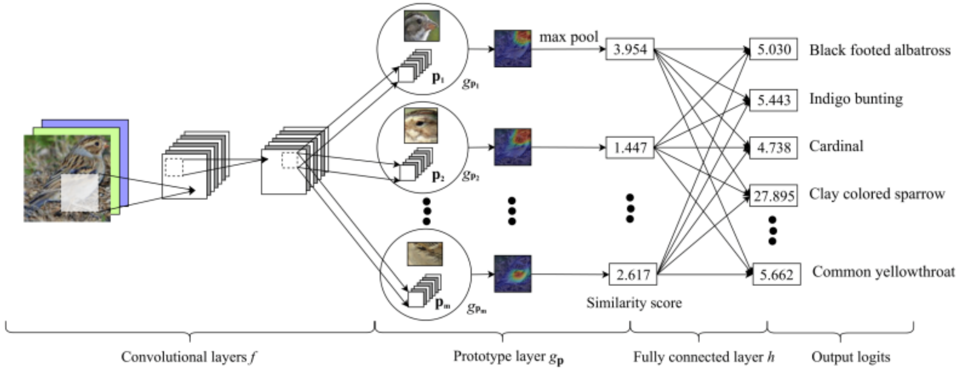
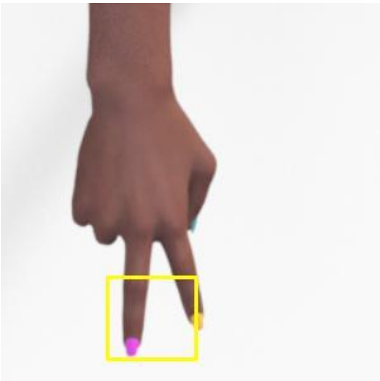
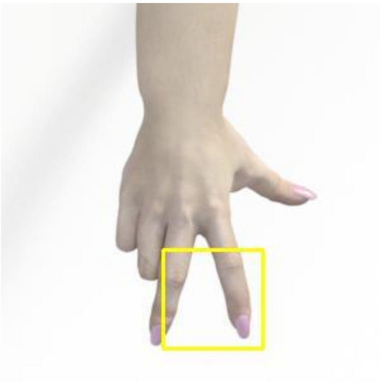


Figure 2: ProtoPNet architecture.



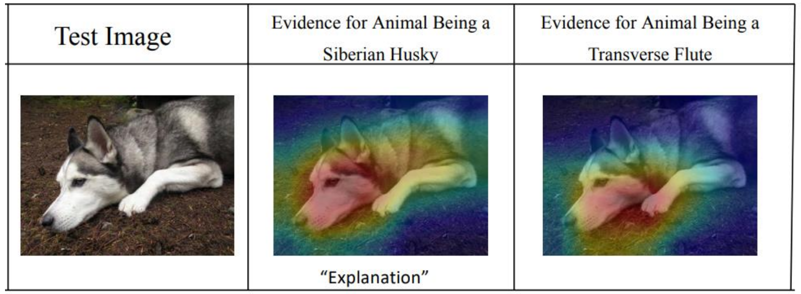


Prototypes in XAI

Previous XAI methods

Posthoc methods

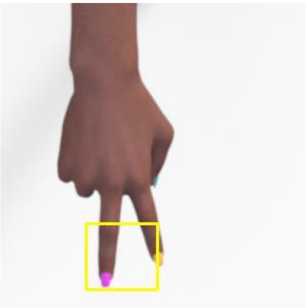
- Used on already trained models
- The most important techniques are: activation maximalization, deconvolution, saliency visualization
- They don't actually explain models decision

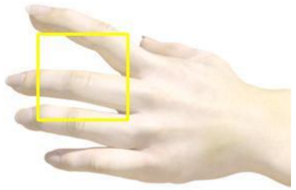


Źródło: IMIC Keynote 3, C. Rudin MICCAI 2021

Methods with built-in attention mechanisms

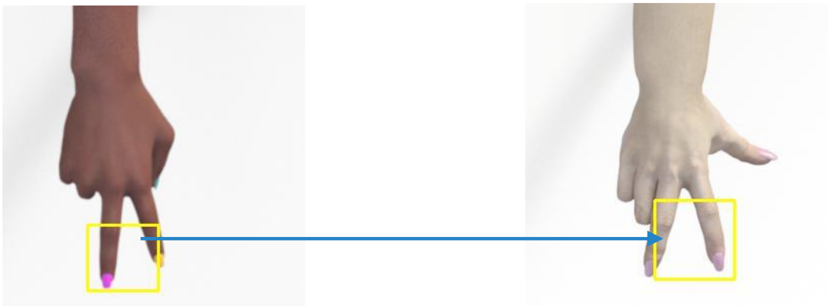
- They mark areas that are important in models decision process
- The most prominent ones are: models with activation maps and part-based models
- They only mark the important area – they don't explain why

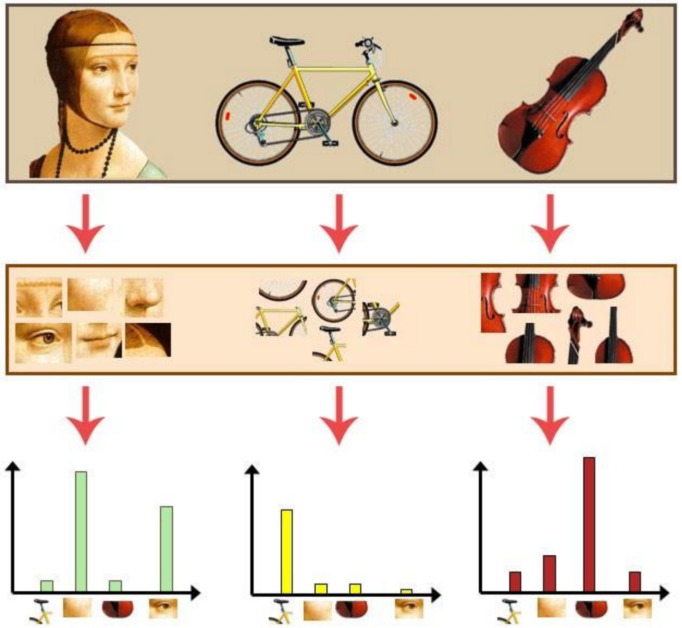




Prototypes

- Main inspiration is the way how human experts try to explain complex ideas with examples
- The model explains it's decision by providing set of prototypes from training data
- Current methods are based on the concept of *bag-of-visual-words*
- So far prototype selection was separate from feature extraction





[Bag of Visual Words in a Nutshell | by Bethea Davida | Towards Data Science]

ProtoPNet (*Prototype Part Network*)

- Architecture proposed in „This looks like that: Deep Learning for Interpretable Image Recognition" (Chen et al. 2019 1806.10574.pdf (arxiv.org))
- Principles
  - Prototypes are chosen based on the distance to sample in latent space
  - Prototype features are used during training
  - Interpretable architecture
- Inspired by previous methods. The closest ones are: Bayesian Case Model (main idea) and method from „Deep Learning for Case-Based Reasoning through Prototypes" (Li et al. 2017 https://arxiv.org/pdf/1710.04806) which takes autoencoder for feature selection
- Architecture successfuly used in IAIA-BL for detecting and evaluating breast lesions (Barnett et al. 2021, 2103.12308.pdf (arxiv.org))

ProtoPNet architecture and training

ARCHITECTURE, LOSS FUNCTION, PROTOTYPES SELECTION

Architecture

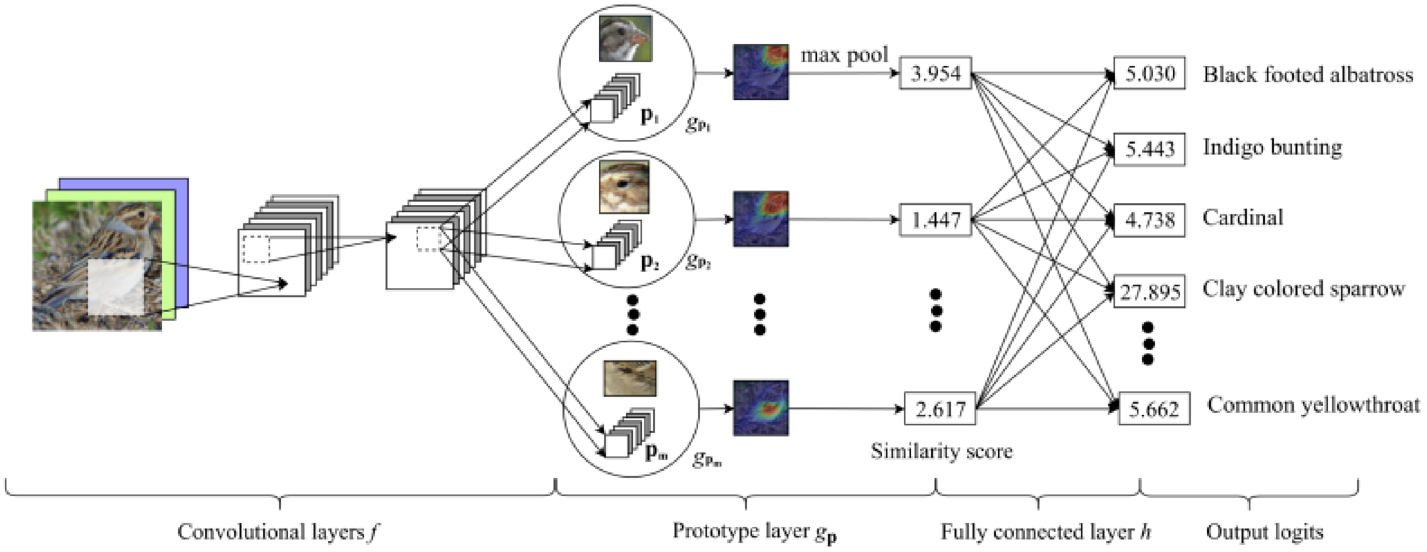
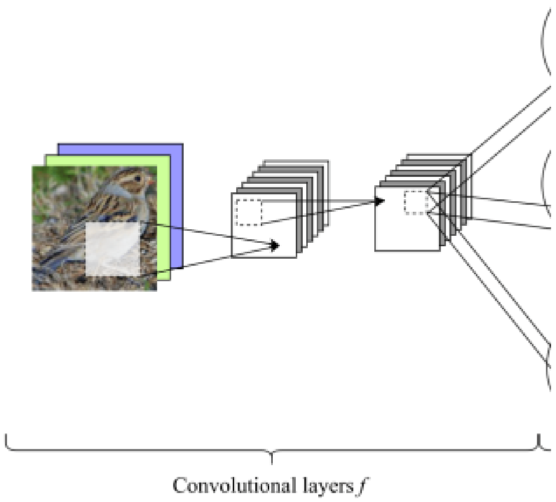
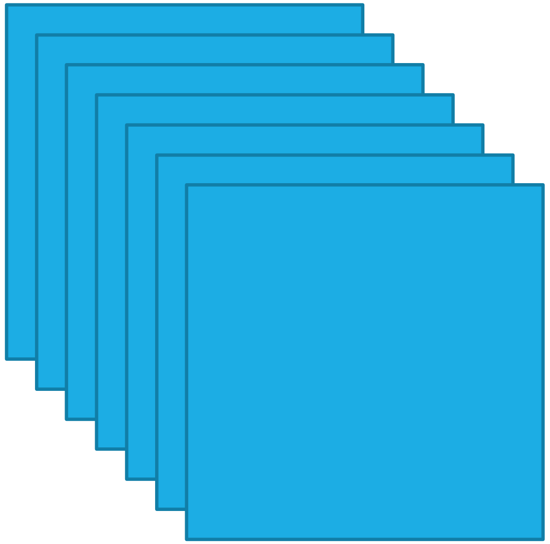


Figure 2: ProtoPNet architecture.

Latent feature maps



Feature maps $\hat{x}$



$\hat{x} : [C, W, H]$

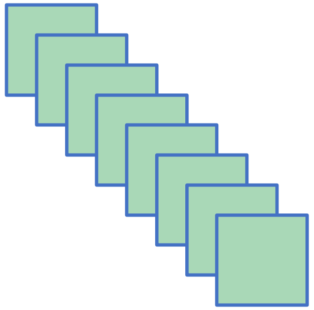
- For given input x convolutional layers f extract useful features
- Output $f(x)$ is passed through additional convolutional layers with 1x1 filters
- Number of channels of these layers corresponds to the size of prototype vectors length
- The final output is tensor $\hat{x}$ of size

$$\hat{x} : [C, W, H]$$

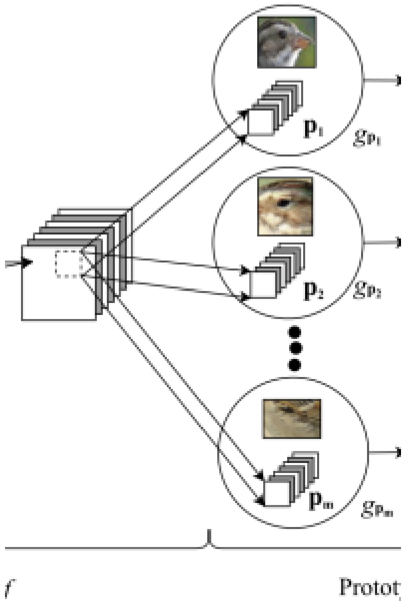
C – num. of channels, W -width, H - height
example: using VGG16 the spatial dimention of convolutional output is [512,5,5], the it is reduced with additional layers to e.g. [C=128,5,5]

Prototype part vectors

Prototype part vector p_j



$p_j : [C, W_p, H_p]$



- Based on image prototypes selected during training
- They reperesnt class specific features
- Each vector represents single part of the prototype image

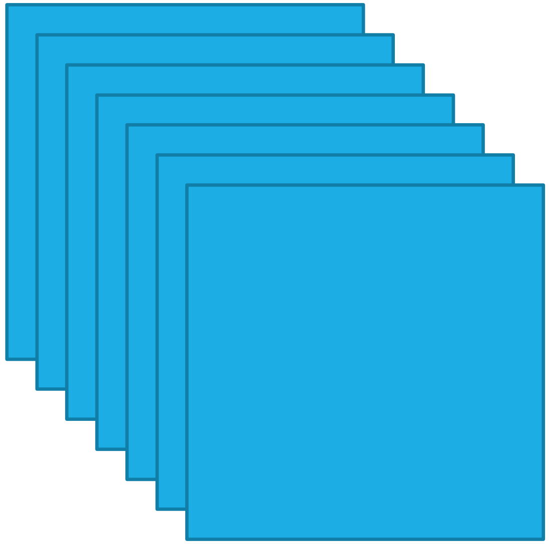
$$p_j : [C, W_p, H_p]$$

C – num. of latent channels, W_p–prototype width, H_p– prototype height
e.g. [512,1,1]

- Each of the dimensions is a hyperparameter

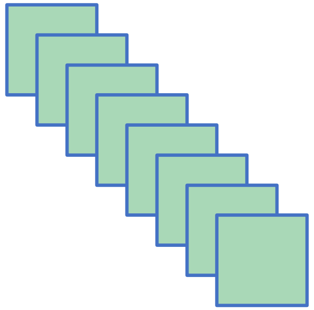
L2 distance to prototype part vectors

Feature maps $\bar{x}$

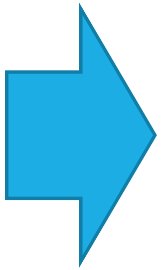


$\hat{x} : [C, W, H]$

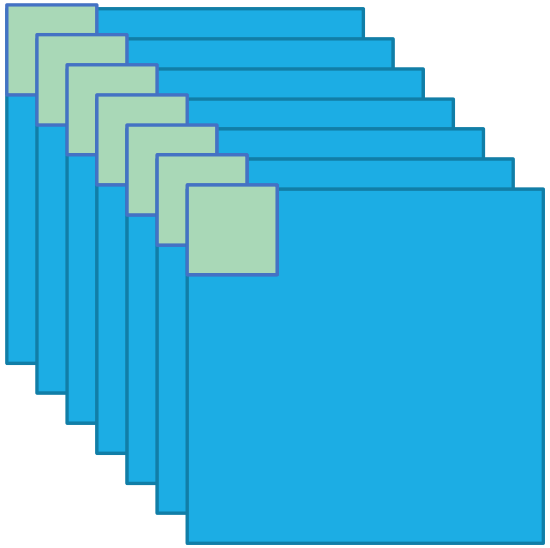
Prototype part vector p_j



$p_j : [C, W_p, H_p]$



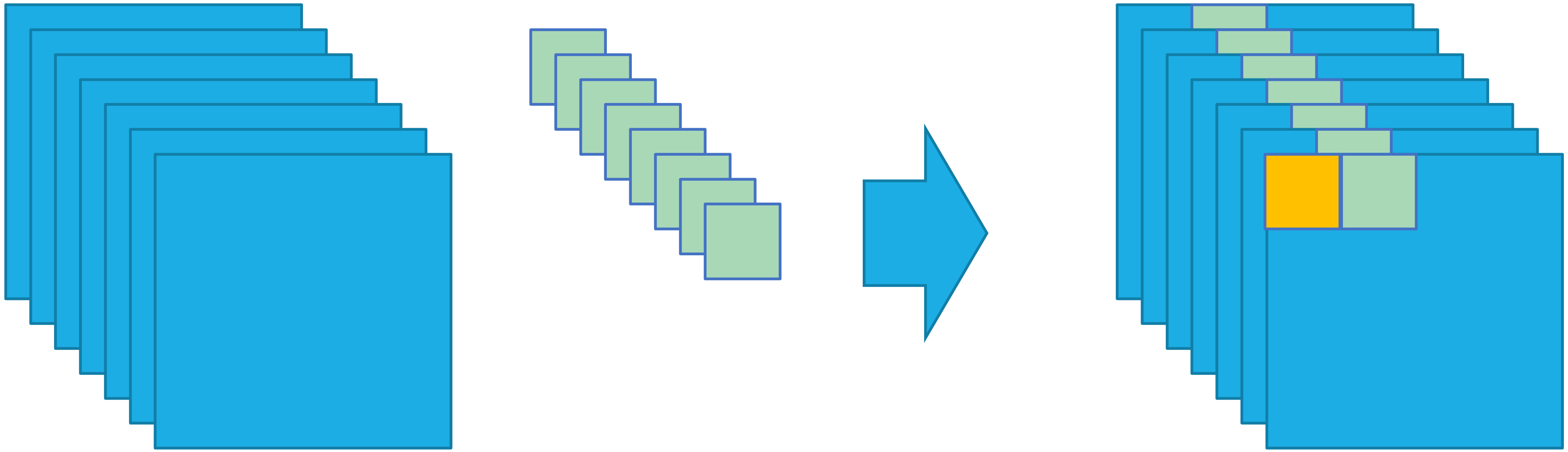
L2 distance is calculated for each input patch



$$L2:\ d_j = \sum_{j}^{m} \left(\hat{x} - p_j\right)^2$$

L2 distance to prototype part vectors

Feature maps $\bar{x}$ Prototype part vector p_j



$\hat{x} : [C, W, H]$

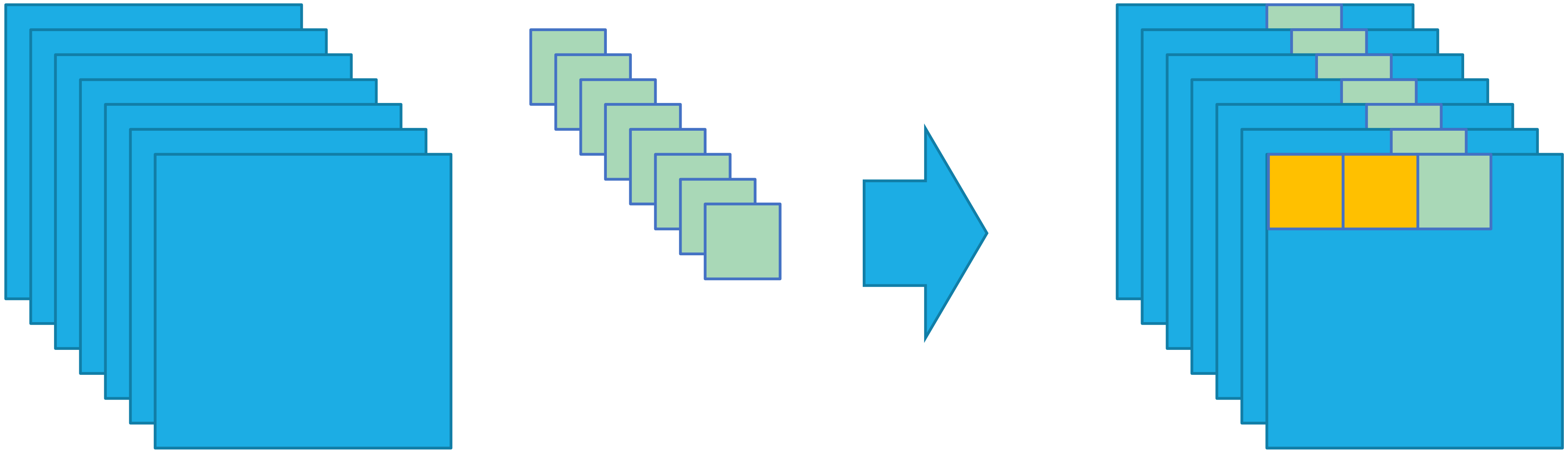
$p_j : [C, W_p, H_p]$

$L2{:}\ d_j = \sum_j^m (\hat{x} - p_j)^2$

L2 distance to prototype part vectors

Feature maps $\bar{x}$

Prototype part vector p_j



$\hat{x} : [C, W, H]$

$p_j : [C, W_p, H_p]$

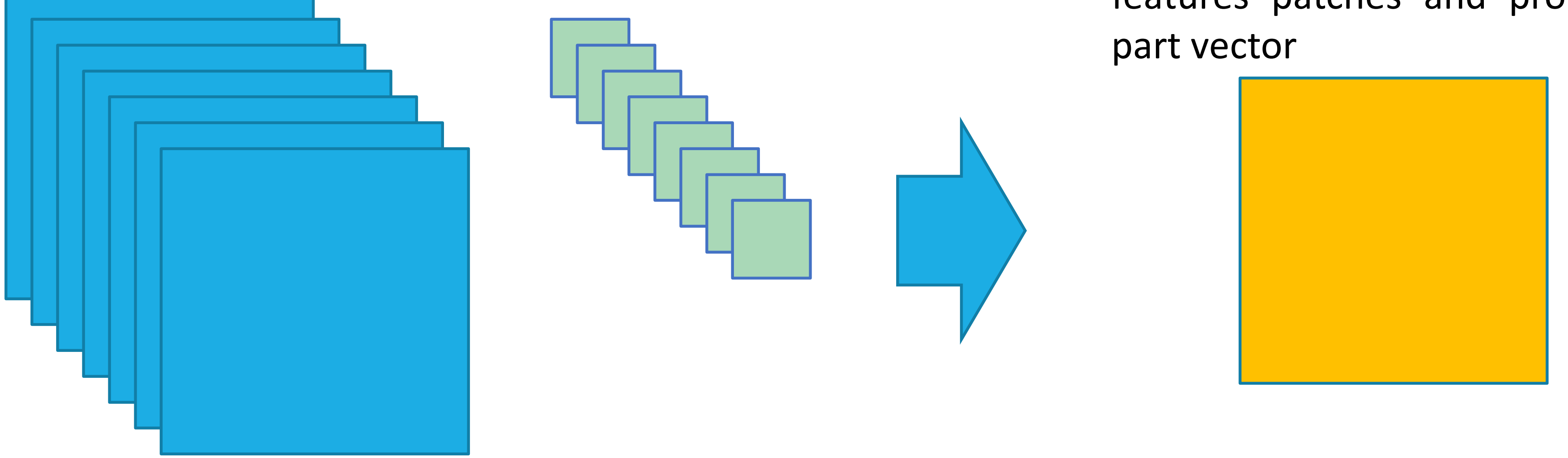
$$L2:\ d_j = \sum_{j}^{m} \left(\hat{x} - p_j\right)^2$$

L2 distance to prototype part vectors

Feature map $\bar{x}$

Prototype feature vector p_j

Distance map d_j between input features patches and prototype part vector



$\hat{x} : [C, W, H]$

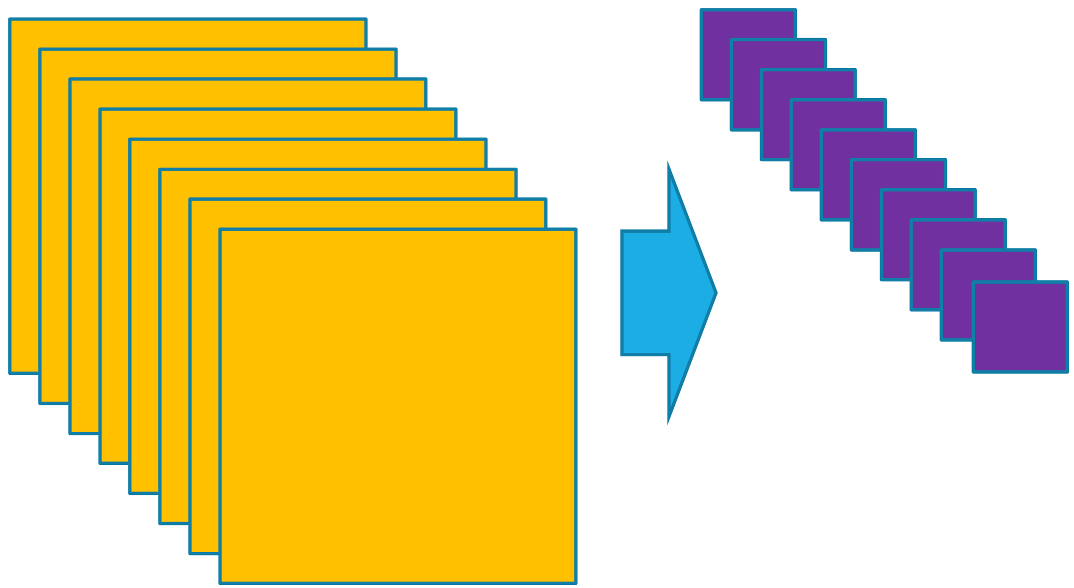
$p_j : [C, W_p, H_p]$

$d_j : [W, H]$

Similarity score

Distance maps d to each of the prototypes

Similarity score vector s



- Similarity score is calculated as:

$$s_j = \log \frac{\min d_j + 1}{\min d_j + \epsilon}$$

between input feature map and j-th prototype part vector

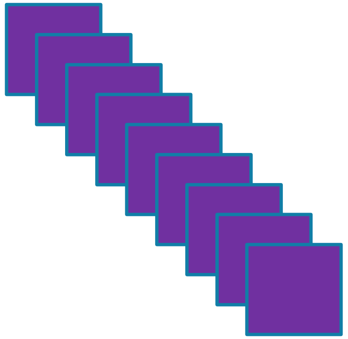
Intuitively:

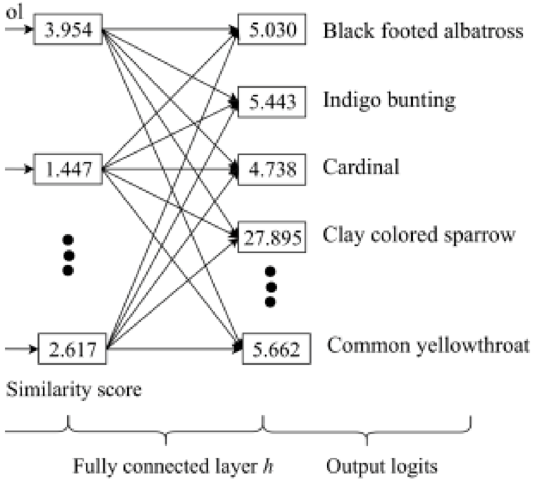
if $d_j \to 0$ then $s \to \log \frac{1}{\epsilon}$

when $d_j \to \infty$ then $s \to 0$

Classification

Similarity score s





- To get output predictions $\hat{y}$ the similarity scores are passed through dense layer

  Interpretability! – you can measure how similarity score to each prototype influenced prediction
- Dense layer weights initialized as:
  - 1 if output class is the same as prototype class
  - -0.5 otherwise

Training

The loss function for training is as follows:

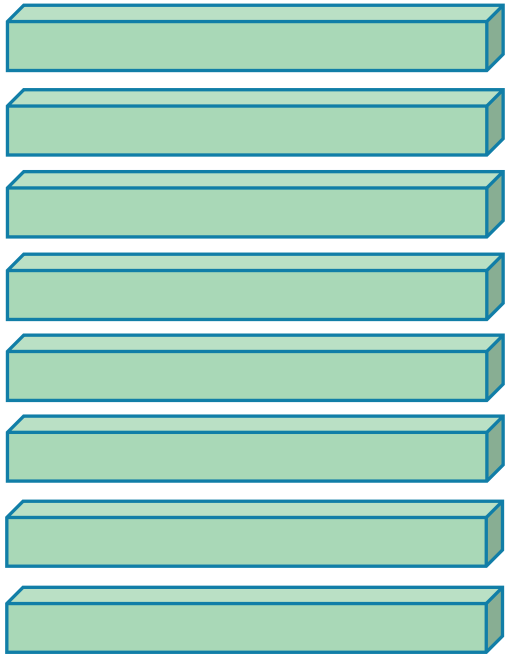
$$loss = \min_{P,wconv} \frac{1}{n} \sum_{i=1}^{n} CrsEnt(\hat{y}_i, y_i) + \lambda_1 Clst + \lambda_2 Sep$$

- $Clst$ is the shortest distance between sample feature map and prototype part vector of the same class
- Sep is the shortest distance to prototype part vector
- Authors suggest that $\lambda_2 = -0.1\lambda_1$ to enforce higher separation between classes

Convergence if:

1. Crossentropy is close enaugh to local minimum
2. $Clst < Sep$ which means closer distance between parts of prototypes belonging to the same class

Prototype selection

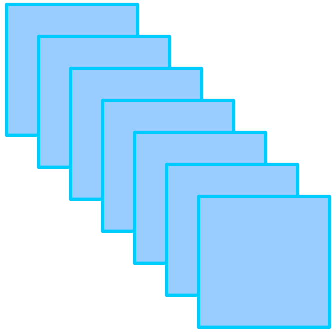


Prototype part vectors P

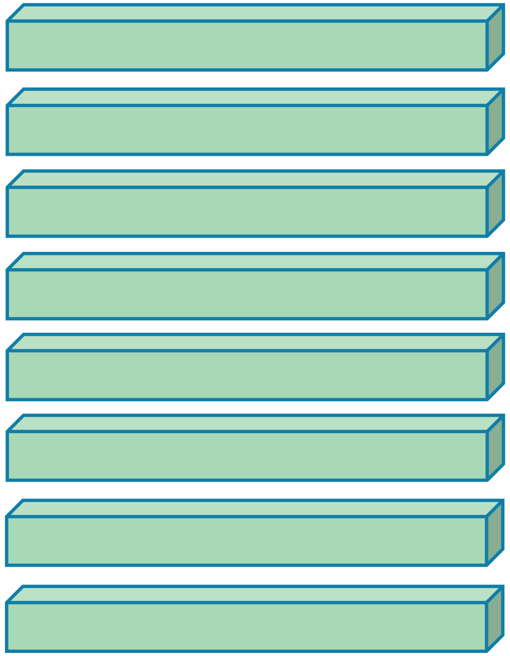
1. Prototypes are selected during training
2. Prototype part vectors are randomly initialized
3. Every few epochs prototypes are updated
4. Each class must have fixed numer of prototypes

Prototype selcetion

For prototype p_j

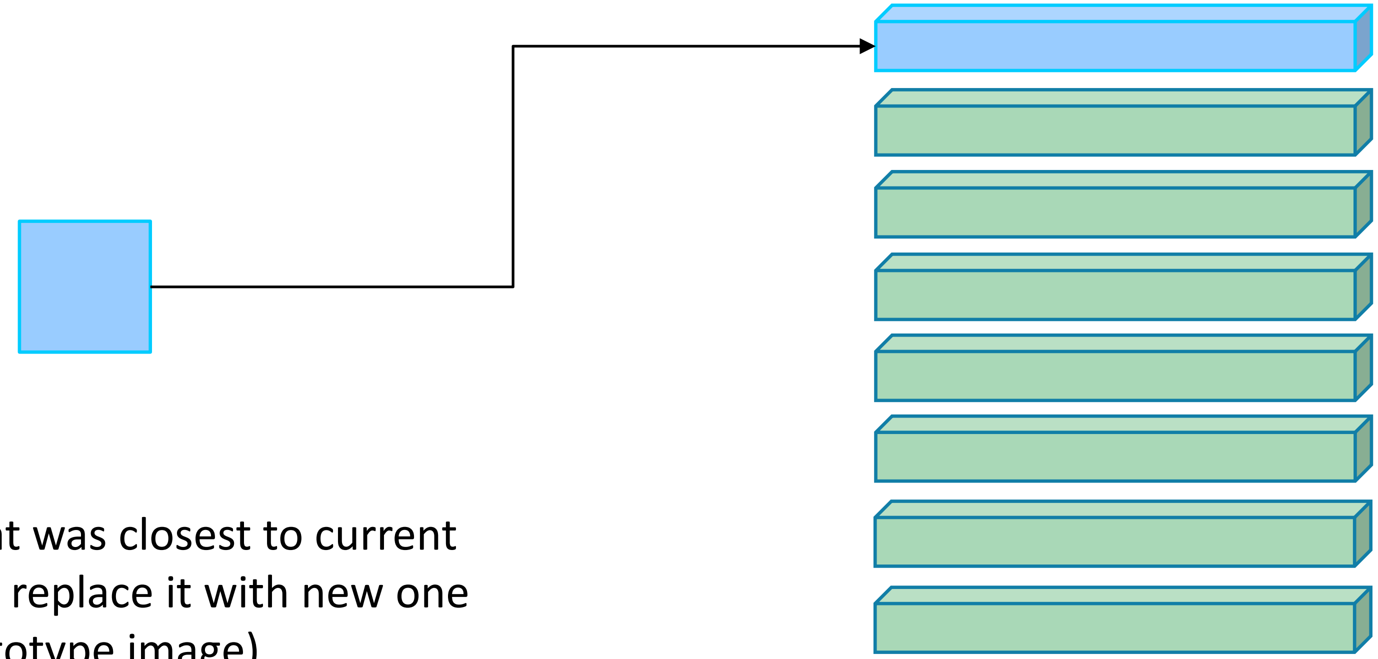


Calculate L2 distance to all the feature maps parts for the whole training data



Prototype part vectors P

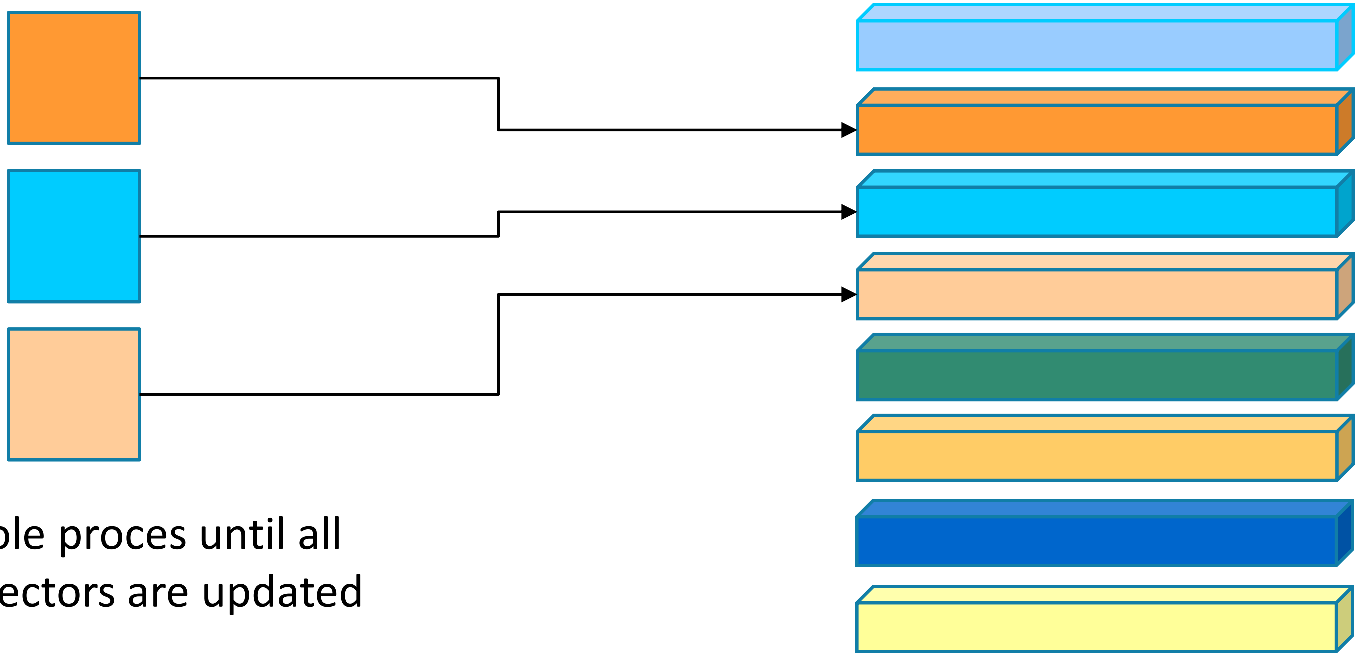
Prototype selcetion



Select part that was closest to current prototype and replace it with new one (also save prototype image)

Prototype part vectors P

Prototype selcetion



Repeat the whole proces until all prtotype part vectors are updated

Prototypes part vectors P

Experiment

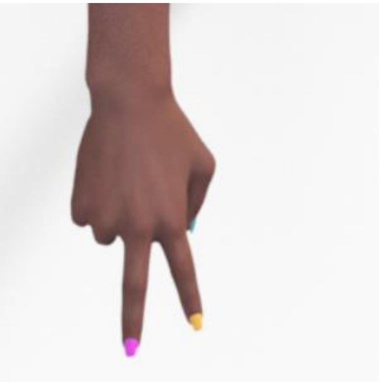
PROTOTYPES AND METRICS RESULTS

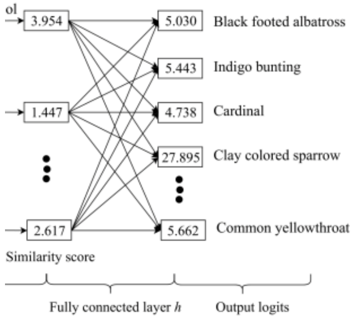
Experiment setup

Experiment goal was to recreate ProtoPNet model for different task and to explain it's decisions for given images

- Dataset: *Rock, paper, scissors* (balanced, 2892images, augmented to 8676, 80-20 train and test split)
- Pretrained VGG16 convolutional layers for feature extraction
- 45 prototype part vectors (15 for each class) with dimensions [128,1,1]
- Metrics: accuracy and elements of the loss function (e.g. *Clst*, *Sep*)
- Trained for 100 epochs

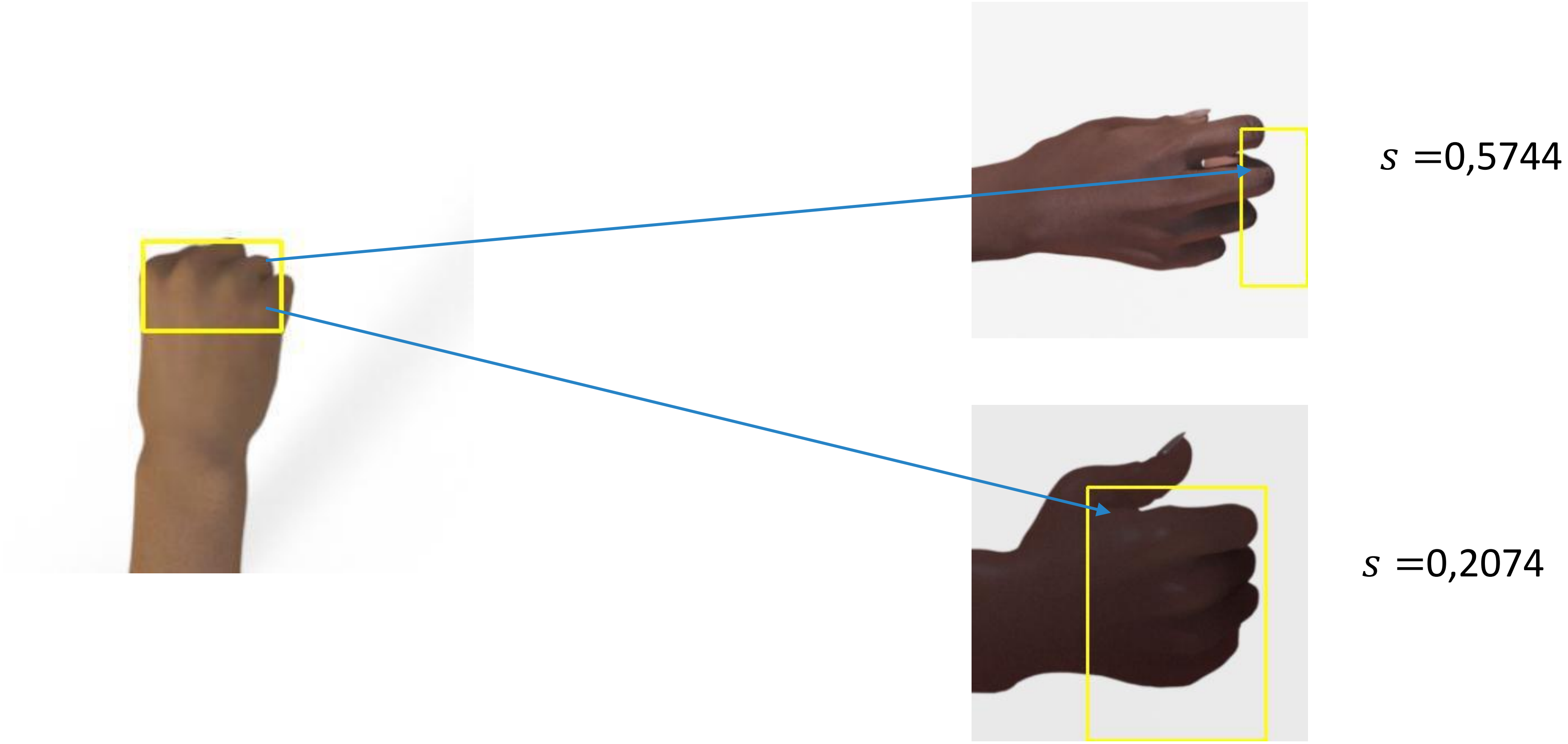
Results - prototypes



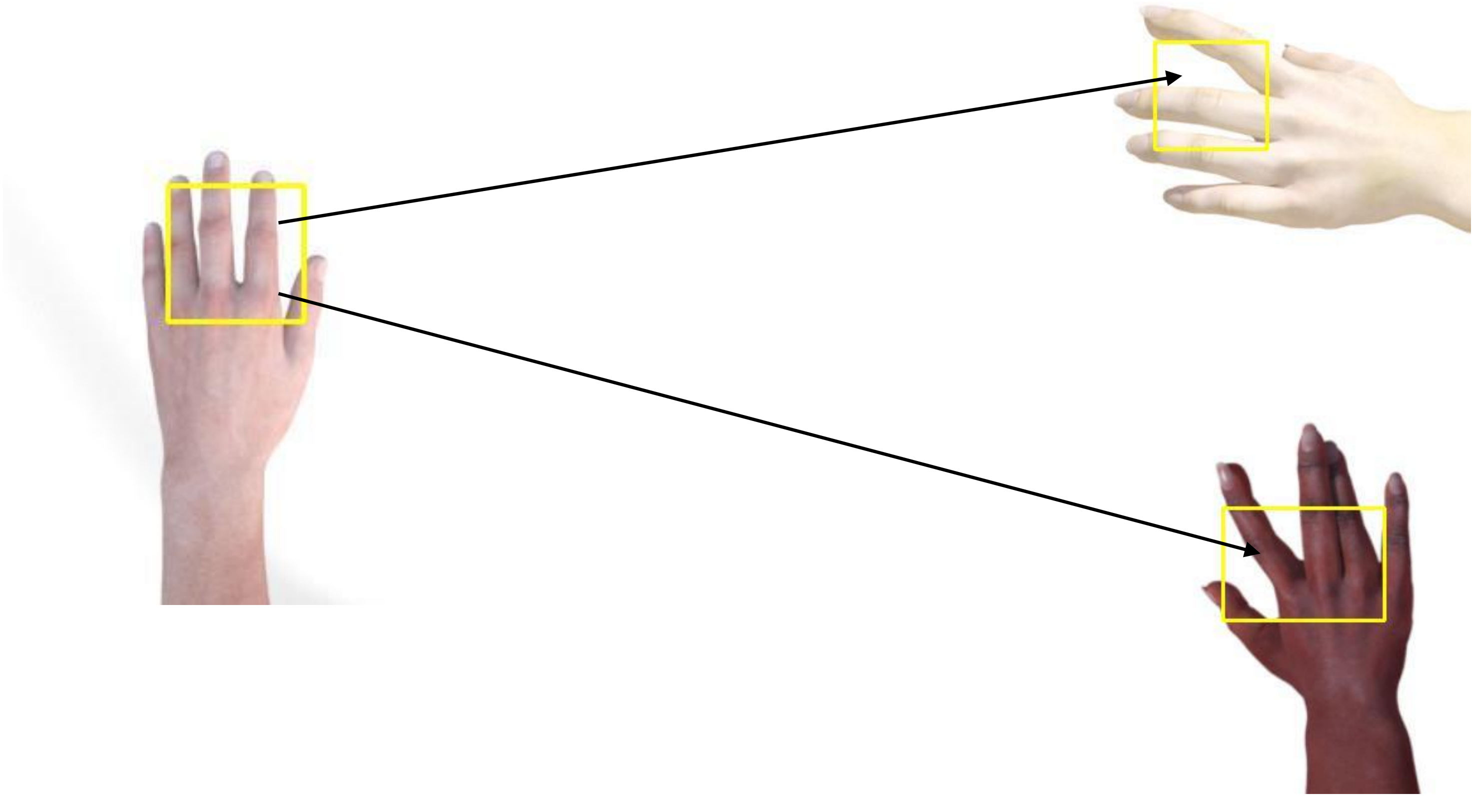


class	top 3 prototypes						top incorrect prototype	
	similarity	connection	similarity	connection	similarity	connection	class 1	class 2
rock	0,5744	1,119	0,2074	1,0695	0,1937	1,0741	scissors; 0,1545	rock; 0,03635
paper	3,9593	1,0205	0,3525	1,0196	0,3503	1,0172	rock; 0,03622	scissors; 0,03355
scissors	2,0435	1,0909	0,2241	1,0839	0,2166	1,0869	rock; 0,01371	paper; 0,0137

Results - prototypes



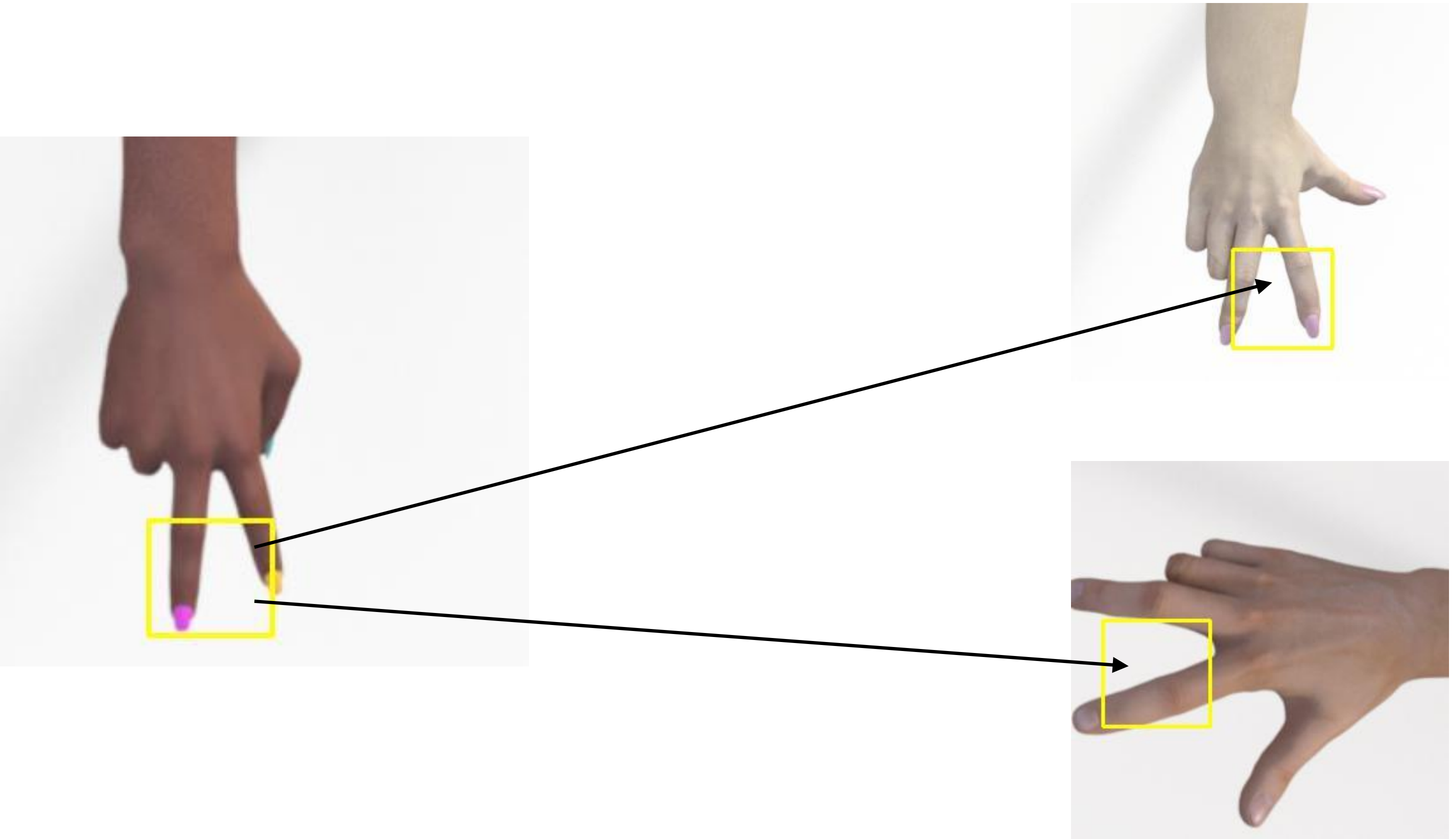
Results - prototypes



s =3,9593

s =0,3525

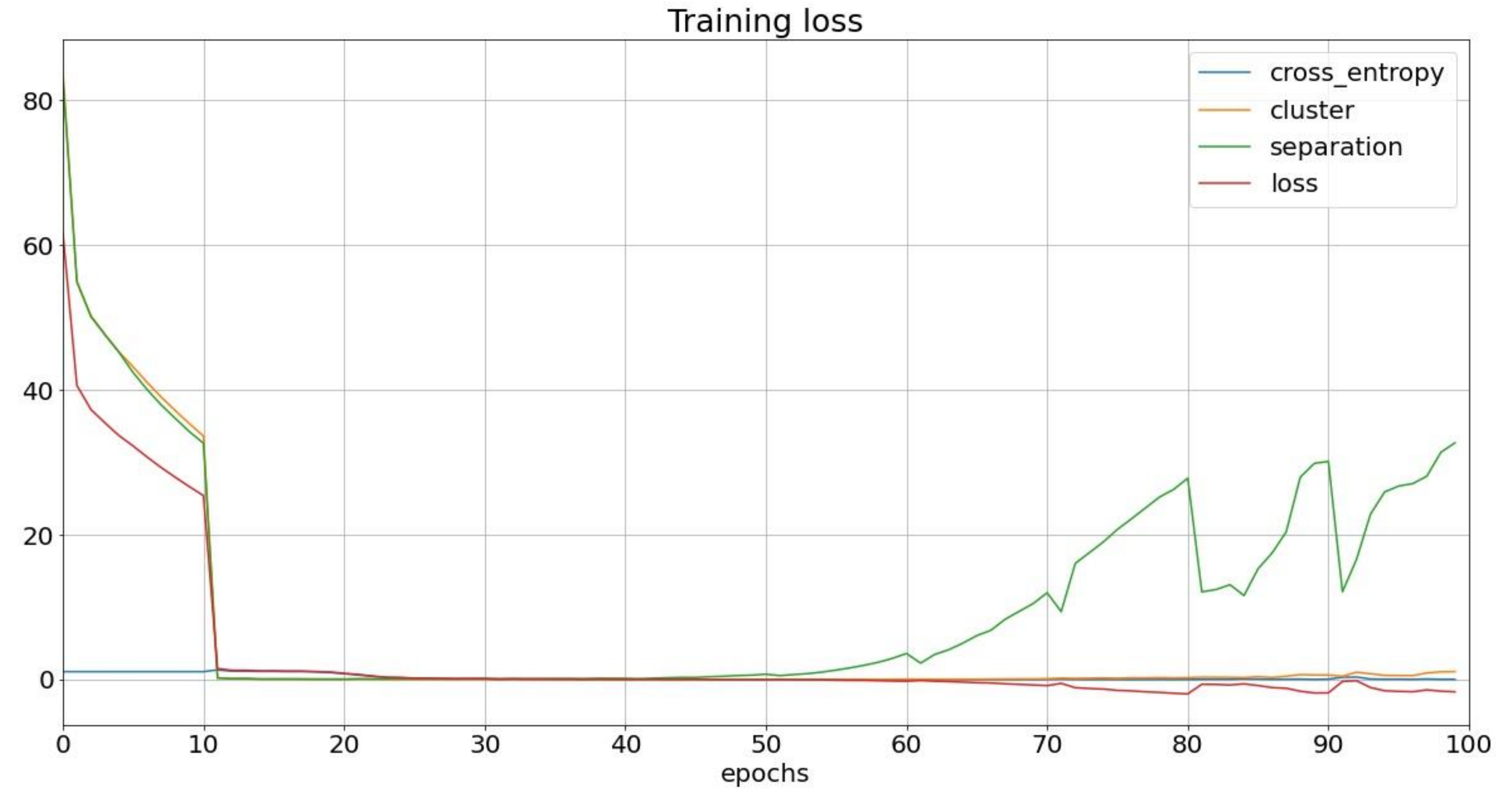
Results - prototypes



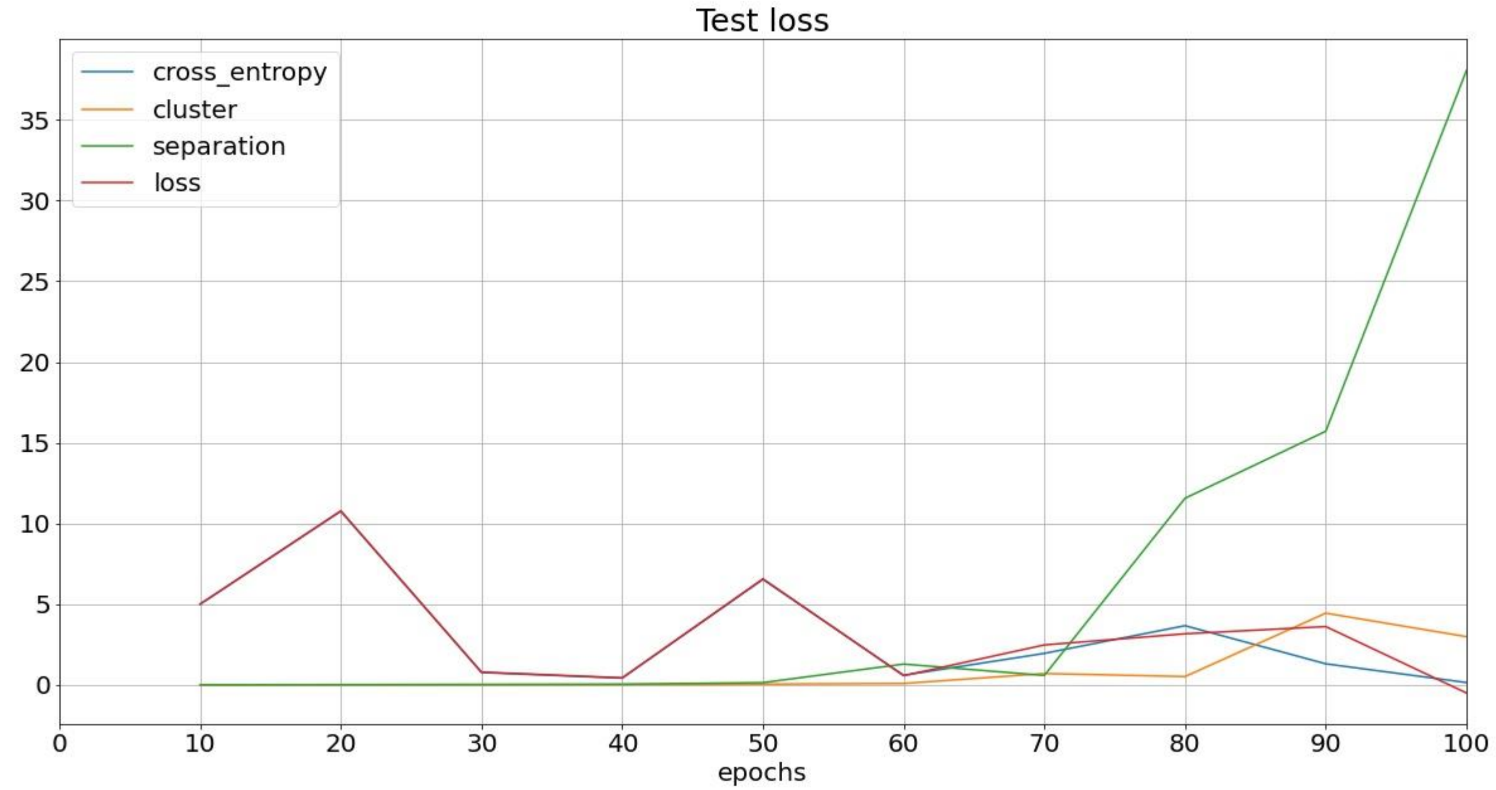
s =2,0435

s =0,2241

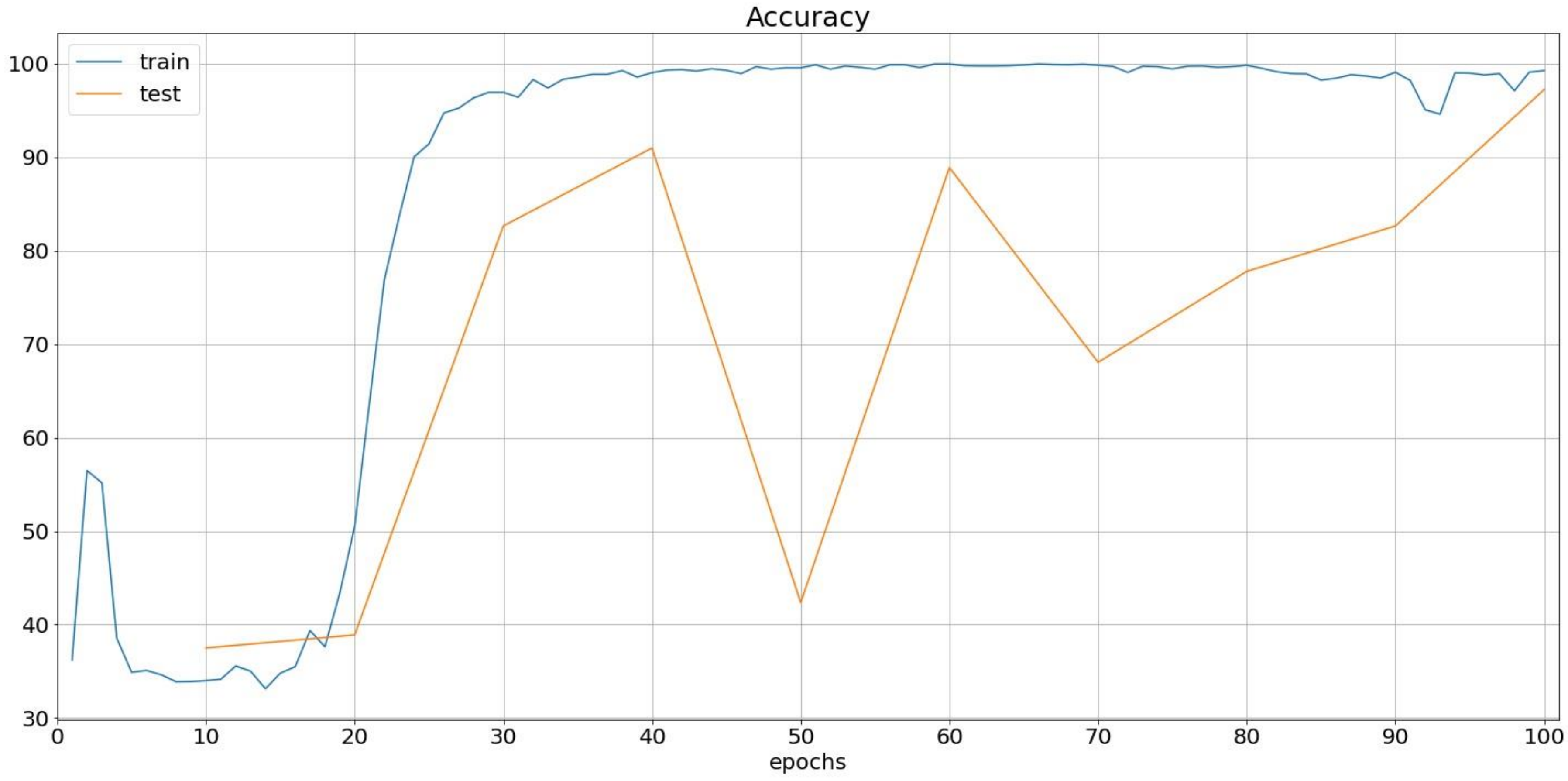
Results - metrics



Results - metrics



Results - metrics



ProtoPNet - summary

Pros:

- Interpretable model output
- Informative visual explanation
- Universal – can be used in various different tasks
  - Unsupervised learning – prototypes can be non-class specific
  - Different input data – prototypes are selected based on latent features
  - Can be used with every architecture

Cons:

- Requires additional training
- Longer training time (higher computational cost + additional metrics to optimize)
- Additional hyperparameters (e.g. number of prototypes)

Thank you!