Neural Prototype Trees

An XAI presentation by **Jakub Bednarz**

Introduction

- ➔ So far we've been trying to perform post-hoc analysis of trained models
- ➔ What if we could *choose* an already explainable class of models from the get-go and skip the explanations "for free"?
- ➔ Problem: how to combine explainability and good performance, especially in domains dominated by deep NNs?
- ➔ Let's take a look at one such method for image classification

Paper

Paper: *Neural Prototype Trees for Interpretable Fine-grained Image Recognition* (Meike Nauta, Ron van Breem, Christin Seifert)

Published in **CVPR 2021**

TL;DR; An intrinsically interpretable DL method for image classification, works like a decision tree with the branches taken based on the presence of trainable prototypical parts in the image.

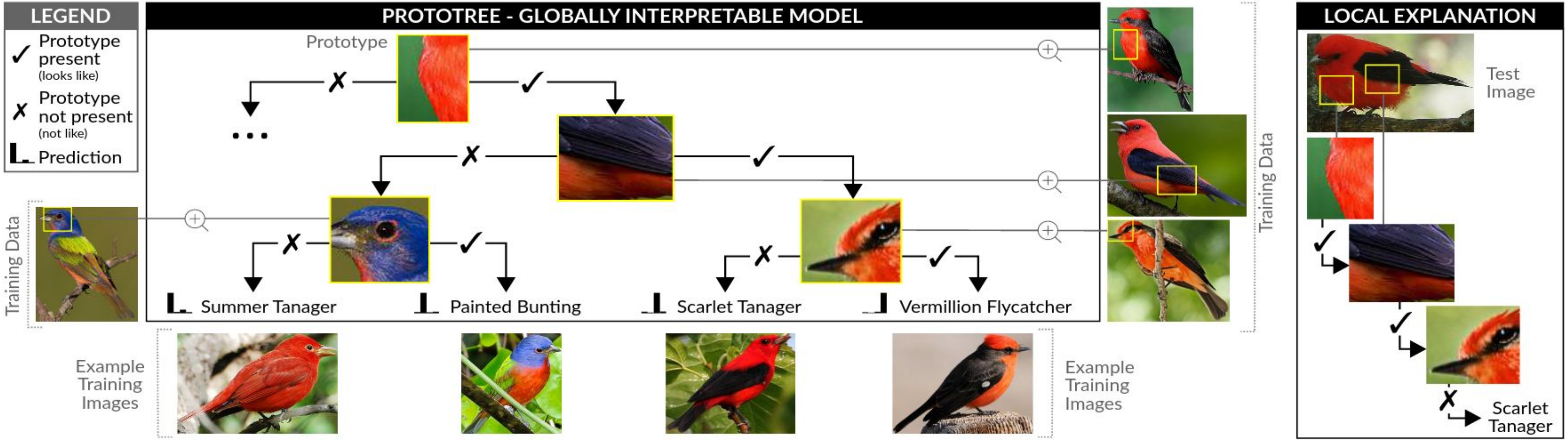


Figure 1. A visual description of the method

Related work

Using prototypes and/or decision trees was done before. Some ideas include:

- ➔ ProtoPNet: Have a bunch of prototypes (each having an assigned class), compute similarity with the image for each one, take a weighted sum
  - Issue: Lots of prototypes needed to get a good result
- ➔ DNDF: Have a “soft” decision tree (routing through a node is stochastic), with probability of taking left or right determined by a NN
  - Issue: The NNs at the decision tree nodes are still not interpretable

So, how does it work?

- ➔ First, we encode the images as feature maps with a CNN - this is the input to the decision tree
- ➔ Each node in the tree contains a prototype - a learnable “patch” in the latent space
- ➔ “Presence of prototype in the image” = minimum distance between the prototype and a patch of the image
- ➔ This value determines (in a soft fashion) whether we route to the left or to the right
- ➔ Leaf nodes contain class distributions

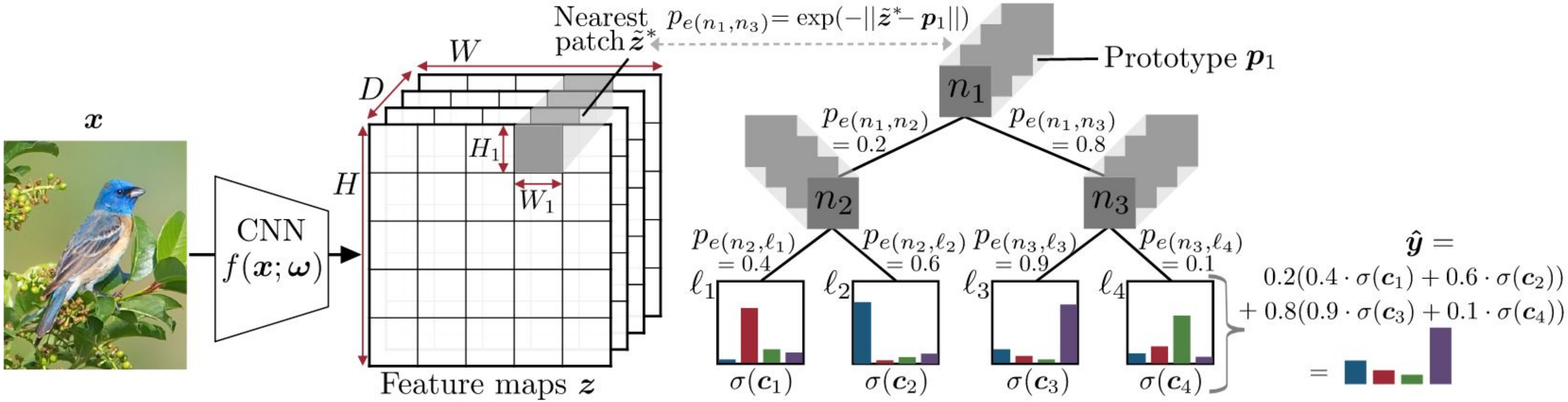


Figure 2. ProtoTree in more detail

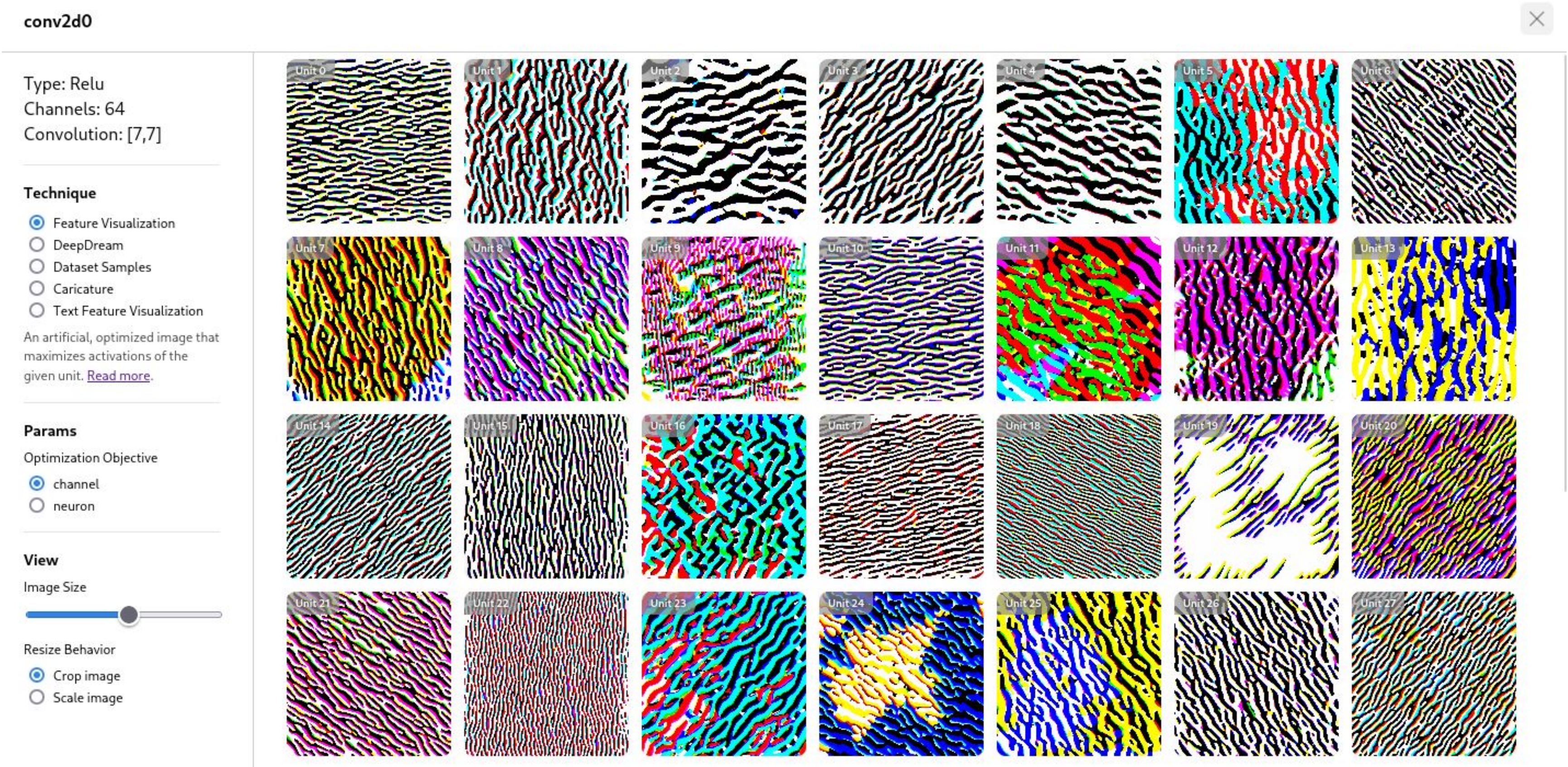


Figure 3. Patch similarity - somewhat similar to line/pattern detectors

Making it interpretable

- ➔ Prototypes are converted to actual patches by checking which patch in the training set matches the prototype best
- ➔ Want to convert “soft” decision trees to “hard” decision trees - select a path with highest probability
- ➔ Making the decision tree smaller via pruning leaves with small discriminative power

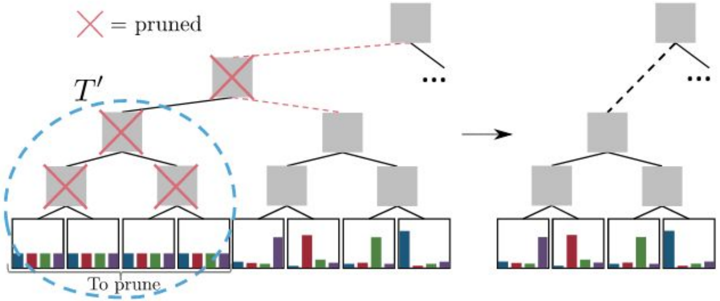
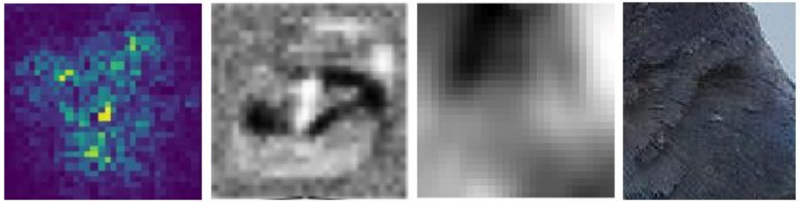


Figure 5: Pruning removes a subtree T', and its parent, in which all leaves have an (nearly) uniform distribution.



(a) DNDF [33] (b) SDT [15] (c) SDT [22] (d) Ours

Figure 3: Visualized root node from soft decision trees. Applied to resp. CIFAR10, MNIST, FashionMNIST and CUB. Republished with permission from the authors (a-c).

Datasets

→ The objective is *fine-grained* image classification
→ The two datasets tested are:
   - CUB: 200 types of birds
   - CARS: 196 models of cars

Results

- ➔ State of the art on CUB is **Metaformer** with 92.9% accuracy
- ➔ State of the art on CARS is **TResNet-L + ML-Decoder** with 96.41% accuracy

Data set	Method	Inter-pret.	Top-1 Accuracy	#Proto types
CUB (224×224)	Triplet Model [34]	-	**87.5**	n.a.
	TranSlider [58]	-	85.8	n.a.
	TASN [57]	o	87.0	n.a.
	ProtoPNet [9]	+	79.2	2000
	ProtoTree h=9 (ours)	++	82.2±0.7	**202**
	ProtoPNet ens. (3) [9]	+	84.8	6000
	ProtoTree ens. (3)	+	86.6	605
	ProtoTree ens. (5)	+	**87.2**	1008
CARS (224×224)	RAU [36]	-	**93.8**	n.a.
	Triplet Model [34]	-	93.6	n.a.
	TASN [57]	o	93.8	n.a.
	ProtoPNet [9]	+	86.1	1960
	ProtoTree h=11 (ours)	++	86.6±0.2	**195**
	ProtoPNet ens. (3) [9]	+	91.4	5880
	ProtoTree ens. (3)	+	90.3	586
	ProtoTree ens. (5)	+	**91.5**	977

Thoughts/Discussion

- ➔ The pixel-space prototypes are not used “directly”, but found after training - what if we can’t find it?
- ➔ In a similar vein, trying to explain how predictions are made to a stakeholder may be difficult (“similarity with a learnable patch in the latent space” doesn’t *sound* all that convincing)
- ➔ Still, it seems better than trying to explain Integrated Gradients or any other such methods
- ➔ Results on different datasets?