Learning Support and Trivial Prototypes for Interpretable Image Classification (Supplementary Material)

Chong Wang[1] Yuyuan Liu[1] Yuanhong Chen [1] Fengbei Liu[1] Yu Tian[2]
Davis McCarthy[3] Helen Frazer[4] Gustavo Carneiro[5]
[1] Australian Institute for Machine Learning, University of Adelaide
[2] Harvard University [3] St Vincent's Institute of Medical Research
[4] St Vincent's Hospital Melbourne [5] CVSSP, University of Surrey

1. Training Details

In our experiments, the proposed ST-ProtoPNet method is trained with Adam [6] optimiser. The training process starts from a warm-up stage for 5 epochs, where the pre-trained CNN backbone is frozen and we only optimise the parameters of add-on layers and prototype layers, using a learning rate of 3×10^{-3}. Then, we jointly train the CNN backbone, add-on layers, and prototype layers, where the learning rate for the CNN backbone is chosen as 1×10^{-4} (CUB [14]), 1×10^{-4} (Cars [7]), 1×10^{-5} (Dogs [5]). In the third stage of fully-connected (FC) layer optimisation, we utilise a learning rate of 1×10^{-4} for 10 iterations. In cropped CUB and cropped Cars datasets, we decrease the learning rate by a factor of 0.2 per 2 epochs and use a weight decay (5×10^{-4} for VGG and ResNet, 1×10^{-3} for DenseNet) on the CNN backbone and add-on layers. In full CUB, we decrease the learning rate by a factor of 0.2 per 20 epochs, a weight decay of 1×10^{-4} is applied on all CNN backbones and add-on layers. In full Dogs, we decrease the learning rate by a factor of 0.2 per 10 epochs, a weight decay of 5×10^{-4} is applied on all CNN backbones and add-on layers. For cropped CUB and cropped Cars, the two 1×1 convolutional layers in the add-on layers are activated by a tanh function, as a form of normalisation.

Note that, during training we alternate the optimisation of each branch of our ST-ProtoPNet among mini-batches, where we observe that our method only introduces marginal extra computational burden and does not require much additional training time to converge since the CNN backbone is shared and optimised by both branches.

We implemented our ST-ProtoPNet method with Pytorch [12] framework and performed experiments on a machine with 4 NVIDIA GeForce RTX 3090 GPUs.

2. Additional Results on Cars and Dogs

In Tables 1 and 2, we demonstrate the classification results of deep-learning based methods that have different levels of interpretability on Stanford Cars and Stanford Dogs datasets, respectively.

As can be observed in Table 1, an ensemble of three ST-ProtoPNets (based on VGG19, ResNet34, and DenseNet121) reaches a high accuracy of 93.8% on Stanford Cars, outperforming the three ensembled TesNet method (93.1%) that uses the same CNN backbones. Notice that our three ensembled ST-ProtoPNet also exceeds the three ensembled ProtoPNet, ProtoTree, and ProtoPool methods by a margin of at least 2.4%. Additionally, the ensemble of five ST-ProtoPNets, based on VGG19, ResNet34, ResNet152, DenseNet121, and DenseNet161, achieves the best classification accuracy of 94.1%.

In Table 2, we can see that a single ResNet152 based ST-ProtoPNet model has a high accuracy of 87.2% on full Stanford Dogs. An ensemble of three ST-ProtoPNets, based on ResNet50, ResNet152, and DenseNet161, performs better than other three ensembled methods utilising the same CNN backbones, e.g., ProtoPNet and Deformable ProtoPNet. The ensemble of five ST-ProtoPNets, using VGG19, ResNet50, ResNet152, DenseNet121, and DenseNet161, obtains the state-of-the-art classification accuracy of 88.7%, surpassing other competing methods that are also based on an ensemble of five models, e.g., ProtoPNet and Deformable ProtoPNet.

3. Ablation on Coefficient λ_3

The coefficient λ_3 in our method controls the strength of the proposed closeness loss ℓ_{cls} and discrimination loss ℓ_{dsc} for the learning of support and trivial prototypes, respectively. We perform an ablation experiment to study the sensitivity of our ST-ProtoPNet method to the hyperparameter λ_3. Table 3 shows the effect of λ_3 on the classification accuracy of our ST-ProtoPNet. As evident, if λ_3 is too small, other loss terms (e.g., cross-entropy, clustering, and separation) will dominate the optimisation process, which leads to performance degradation. However, if λ_3 is too large,

Interpretability level	Method	Accuracy
None	B-CNN [9]	91.3
Object-level attention	CSG [8]	91.6
Part-level attention	MA-CNN [16]	92.8
	RA-CNN [3]	92.5
	TASN [17]	**93.8**
Part-level attention + Prototypes	Region [4]	90.9
	ProtoPNet* [1]	91.4
	ProtoTree* [11]	90.5
	TesNet* [15]	93.1
	ProtoPool* [13]	91.1
	ST-ProtoPNet* (ours)	93.8
	ProtoTree** [11]	91.5
	ProtoPool** [13]	91.6
	ST-ProtoPNet** (ours)	**94.1**

Table 1. Classification accuracy and interpretability level of different methods on Stanford Cars. *: Ensembled of three models. **: Ensembled of five models.

Interpretability level	Method	Accuracy
Part-level attention	FCAN [10]	84.2
	RA-CNN [3]	87.3
Part-level attention + Prototypes	ProtoPNet [1]	79.7
	Deformable ProtoPNet [2]	86.5
	ST-ProtoPNet (ours)	87.2
	ProtoPNet* [1]	83.6
	Deformable ProtoPNet* [2]	87.1
	ST-ProtoPNet* (ours)	88.6
	ProtoPNet** [1]	84.1
	Deformable ProtoPNet** [2]	87.3
	ST-ProtoPNet** (ours)	**88.7**

Table 2. Classification accuracy and interpretability level of different methods on Stanford Dogs. *: Ensembled of three models. **: Ensembled of five models.

λ_3	0	0.01	0.1	0.5	1.0	2.0	5.0	10.0
Accuracy	81.6	81.8	82.7	83.4	83.5	83.3	83.0	81.8

Table 3. Ablation to study the effect of hyperparameter λ_3 on the accuracy of our ResNet34-based ST-ProtoPNet on cropped CUB.

other losses (particularly the cross-entropy loss) will be distracted, resulting in lower discrimination ability. Therefore, we choose $\lambda_3 = 1.0$ in all our experiments.

4. Ablation on Independent Add-on Layers

In our ST-ProtoPNet method, the support and trivial ProtoPNets are branched after the CNN backbone and they have independent add-on layers. One may also consider using shared add-on layers, where both the support and trivial prototypes will employ the same feature maps. We thus design an ablation study on full CUB to compare the classification results between using independent and shared add-on layers. Table 4 gives the experimental results with VGG19, ResNet50, and Dense121 as CNN backbones. We can observe that using independent add-on layers exhibits higher classification accuracy across different CNN backbones. One possible explanation is the support and trivial

Method	VGG19	ResNet50	Dense121
ST-ProtoPNet (shared add-on layers)	79.2	87.5	81.2
ST-ProtoPNet (independent add-on layers)	80.2	88.0	81.8

Table 4. Ablation study on using independent or shared add-on layers for our ST-ProtoPNet method.

Method	Accuracy before pruning	Accuracy after pruning	Accuracy after pruning and optimising last layer	Number of pruned prototype
ProtoPNet [1]	78.2	74.4	78.0	666
Trivial ProtoPNet	78.6	74.4	78.3	701
Support ProtoPNet	79.7	74.7	79.4	409

Table 5. Effect of prototype pruning on different ProtoPNet methods trained on cropped CUB.

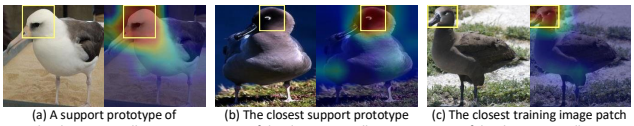

Figure 1. (a) A support prototype from the class Laysan Albatross. (b) Closest support prototype not of the class Laysan Albatross. (c) Closest training image patch not of the class Laysan Albatross.

prototypes have distinctive characteristics that require different feature representations for classification.

5. Effect of Prototype Pruning

Prototype pruning [1] can automatically remove non-discriminative background prototypes. Less pruned prototypes indicate more prototypes focus on the discriminative object regions, we thus favour a smaller number of pruned prototypes. We conduct experiments with pruning for the support and trivial prototypes. Following [1], during pruning we employ 6 nearest latent patches for each prototype and set the pruning threshold $\tau = 3$. In order to achieve fair comparison with the vanilla ProtoPNet [1], we do not use the orthonormality loss ℓ_{ort}, and add only our proposed closeness loss ℓ_{cls} or discrimination loss on ℓ_{dsc} to the vanilla ProtoPNet [1] to establish the Support ProtoPNet or Trivial ProtoPNet, respectively. Table 5 presents the experimental results on cropped CUB by using VGG19 as CNN backbone. As can be seen, the number of pruned support prototypes are substantially reduced, compared with that of trivial prototypes in the vanilla ProtoPNet [1] and Trivial ProtoPNet, verifying that our proposed support prototypes are less likely to be localised on background regions.

6. A Visual Study for Support Prototypes

We provide a visual study to qualitatively validate the resemblance of support prototypes of different classes. To be specific, for a support prototype of class c, we show its closest support prototype as well as closest training image patch that are not of class c. Results in Fig. 1 illustrate for

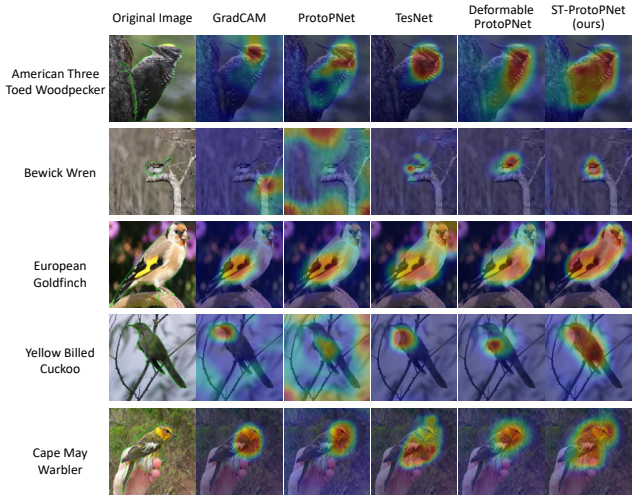

Figure 2. Activation map for images from the full CUB test set. The bird in the original images is indicated by a green boundary outline extracted from the annotated segmentation mask. For prototype-based methods, the activation map is obtained by averaging all activation maps of the class.

a support prototype from the class Laysan Albatross, both its closest support prototype and training image patch have similar appearance (bird's head) and come from the class Sooty Albatross, indicating that a support prototype of a class does closely resemble that of a similar but wrong class and help tell apart difficult classes.

7. Visualisation of Object Localisation on CUB

In Fig. 2, we present some examples of the activation map generated by our ST-ProtoPNet and other competing methods for images from the full CUB test set. It can be observed that, compared with the vanilla ProtoPNet [1], our method mainly relies on the object (i.e., bird) region instead of the background pixels. In addition, we also notice that our method has more activation coverage on the target object than TesNet [15] and Deformable ProtoPNet [2], which demonstrates that our proposed method utilises both the support and trivial prototypes to effectively capture much richer representations of the target object.

8. More Examples of Prototype Visualisation

In Fig. 3 and 4, we provide more visual examples of typical prototypes learned by our top-performing models on Stanford Cars and Stanford Dogs datasets. Specifically, we display prototypes of Dense161-based and ResNet152-based ST-ProtoPNet for Stanford Cars and Stanford Dogs, respectively. Following previous studies [1, 2], all these prototypes are visualised by projecting onto their nearest training patch in the latent feature space.

8.1. Stanford Cars

In Fig. 3, we demonstrate the visual prototypes of four classes (Acura RL Sedan 2012, Acura TL Sedan 2012, Acura TL Type-S 2008, and Acura TLS Sedan 2012) learn from cropped Stanford Cars dataset. We can note from Fig. 3(a) that the support prototypes mainly capture visually similar features of different classes, such as the light and front bumper. By contrast, the trivial prototypes in Fig. 3(b) are inclined to focus on not only lights but also other car parts, such as wheel, door, and window. For the cropped Cars dataset, the target objects of cars usually occupy a large area in the whole image, so we rarely observe background regions in trivial prototypes.

8.2. Stanford Dogs

In Fig. 4, we demonstrate some typical prototypes of five classes (Rhodesian Ridgeback, Toy Terrier, Komondor, Rottweiter, and Dhole) from full Stanford Dogs dataset. It can be observed from Fig. 4(a) that the support prototypes often contain similar visual patterns of different classes, e.g., dog's eye, mouth, and head. By contrast, trivial prototypes in Fig. 4(b) are usually from dog's neck, leg, and belly (lower surface) regions. We also notice that two trivial prototypes capture background regions (rightmost ones of the Rottweiter and Dhole classes).

9. More Examples of Interpretable Reasoning

We provide more examples of the interpretable reasoning process of our ST-ProtoPNet method on Stanford Cars (Dense161-based) and Stanford Dogs (ResNet152-based) datasets, as illustrated in Fig. 5 and Fig. 6. For each example, we display two support and trivial prototypes of the predicted class as well as the corresponding source training images where these prototypes come. For a testing image, the activation maps and highest similarity scores with these prototypes are also provided. For simplicity, we only demonstrate the prototypes, activation maps, and similarity scores contributing to the predicted class for each example.

9.1. Stanford Cars

Fig. 5 displays the interpretable reasoning of our ST-ProtoPNet method in classifying a testing car (Acura TL Type-S 2008) image. In each ProtoPNet branch, our method compares the testing image with training prototypes and computes the corresponding similarity scores, which is then weighted and summed to produce the classification logits. The final classification logits is the sum of logits from both branches. In particular, the support prototypes are mostly active on the fog light and headlight of the car. Meanwhile, the trivial prototypes have high activations on the car's window and wheel. In this case, the support ProtoPNet branch generates a comparably larger similarity score

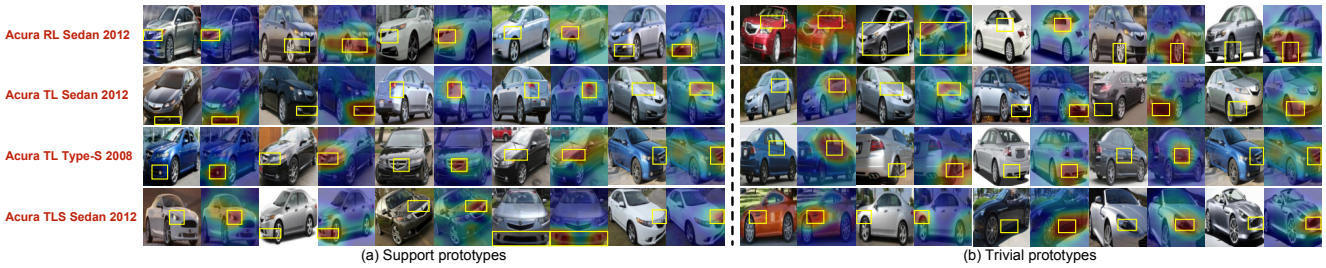

Figure 3. Visual comparison between the support (a) and trivial (b) prototypes from cropped Stanford Cars, where each row exhibits prototypes of the same class. In each pair, the first column shows the original image with a prototype indicated in a yellow bounding box, the second column demonstrates the prototype's corresponding activation map.

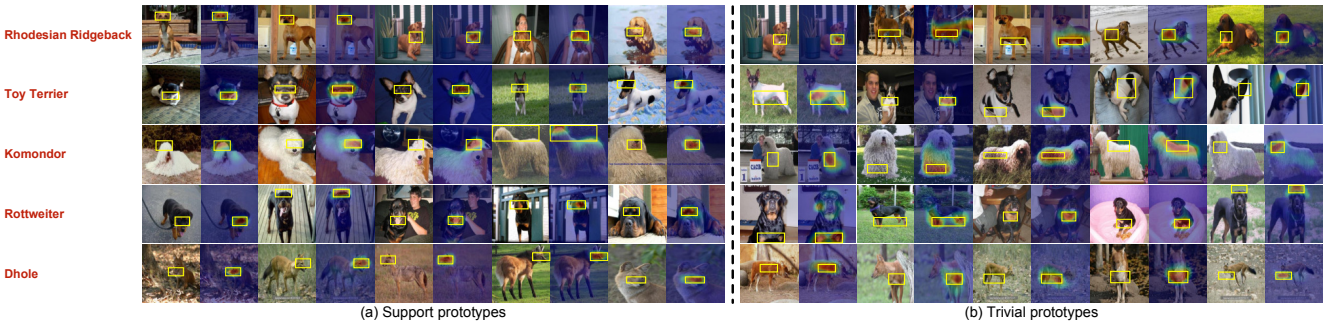

Figure 4. Visual comparison between the support (a) and trivial (b) prototypes from full Stanford Dogs, where each row exhibits prototypes of the same class. In each pair, the first column shows the original image with a prototype indicated in a yellow bounding box, the second column demonstrates the prototype's corresponding activation map.

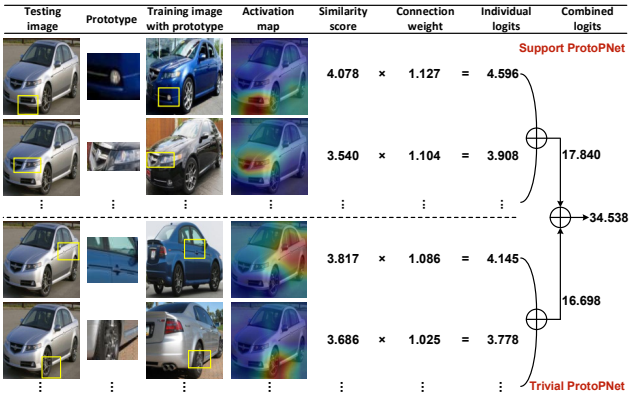

Figure 5. An example of the interpretable reasoning of our ST-ProtoPNet for classifying a testing car (Acura TL Type-S 2008) image.

(17.840) in comparison with the trivial ProtoPNet branch (16.698), meaning that the testing car image is more similar to the support prototypes than the trivial ones. We can see that the support and trivial prototypes make complementary predictions contributing to the final classification decision.

9.2. Stanford Dogs

Fig. 6 reveals the interpretable reasoning of our ST-ProtoPNet method in classifying a testing dog (Rhodesian Ridgeback) image. To be specific, when classifying the Rhodesian Ridgeback dog, its head is usually activated by the support prototypes while its neck and belly are often activated by the trivial prototypes. We should notice that for the full Stanford Dogs dataset, we utilise cosine similarity [2], instead of projection metric [15] used in cropped Stanford Cars, to measure the similarity between the testing image's feature map and prototypes, so the similarity scores are within the range of [0, 1]. In this example, the support and trivial ProtoPNet branches obtain an accumulated similarity score of 35.902 and 35.583, respectively. By using both support and trivial prototypes, our ST-ProtoPNet can capture richer representations of the target object (dog in this example) from different perspectives to achieve complementary interpretations.

References

[1] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3

[2] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF*

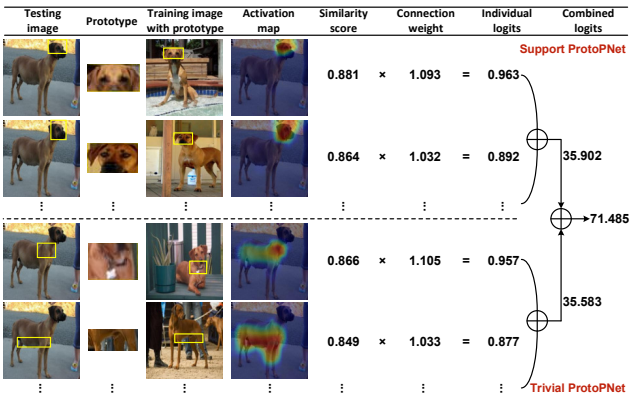

Figure 6. An example of the interpretable reasoning of our ST-ProtoPNet for classifying a testing dog (Rhodesian Ridgeback) image.

Conference on Computer Vision and Pattern Recognition, pages 10265–10275, 2022. 2, 3, 4

[3] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4438–4446, 2017. 2

[4] Zixuan Huang and Yin Li. Interpretable and accurate fine-grained recognition via region grouping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8662–8672, 2020. 2

[5] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-grained Visual Categorization (FGVC)*, volume 2. Citeseer, 2011. 1

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 1

[8] Haoyu Liang, Zhihao Ouyang, Yuyuan Zeng, Hang Su, Zihao He, Shu-Tao Xia, Jun Zhu, and Bo Zhang. Training interpretable convolutional neural networks by differentiating class-specific filters. In *European Conference on Computer Vision*, pages 622–638. Springer, 2020. 2

[9] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015. 2

[10] Xiao Liu, Tian Xia, Jiang Wang, Yi Yang, Feng Zhou, and Yuanqing Lin. Fully convolutional attention networks for fine-grained recognition. *arXiv preprint arXiv:1603.06765*, 2016. 2

[11] Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14933–14943, 2021. 2

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[13] Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, pages 351–368. Springer, 2022. 2

[14] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 1

[15] Jiaqi Wang, Huafeng Liu, Xinyue Wang, and Liping Jing. Interpretable image recognition by constructing transparent embedding space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 895–904, 2021. 2, 3, 4

[16] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5209–5217, 2017. 2

[17] Heliang Zheng, Jianlong Fu, Zheng-Jun Zha, and Jiebo Luo. Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5012–5021, 2019. 2