Learning Support and Trivial Prototypes for Interpretable Image Classification

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

Abstract

Prototypical part network (ProtoPNet) methods have been designed to achieve interpretable classification by associating predictions with a set of training prototypes, which we refer to as trivial prototypes because they are trained to lie far from the classification boundary in the feature space. Note that it is possible to make an analogy between ProtoPNet and support vector machine (SVM) given that the classification from both methods relies on computing similarity with a set of training points (i.e., trivial prototypes in ProtoPNet, and support vectors in SVM). However, while trivial prototypes are located far from the classification boundary, support vectors are located close to this boundary, and we argue that this discrepancy with the well-established SVM theory can result in ProtoPNet models with inferior classification accuracy. In this paper, we aim to improve the classification of ProtoPNet with a new method to learn support prototypes that lie near the classification boundary in the feature space, as suggested by the SVM theory. In addition, we target the improvement of classification results with a new model, named ST-ProtoPNet, which exploits our support prototypes and the trivial prototypes to provide more effective classification. Experimental results on CUB-200-2011, Stanford Cars, and Stanford Dogs datasets demonstrate that ST-ProtoPNet achieves state-of-the-art classification accuracy and interpretability results. We also show that the proposed support prototypes tend to be better localised in the object of interest rather than in the background region.

1. Introduction

Deep convolutional neural networks (CNN) [28, 30, 16] have had remarkable achievements in various visual tasks, e.g., image recognition [16] and object detection [39]. Despite the excellent feature extraction and discrimination ability, CNNs are generally treated as black-box models due to their complex architectures, high-dimensional fea-
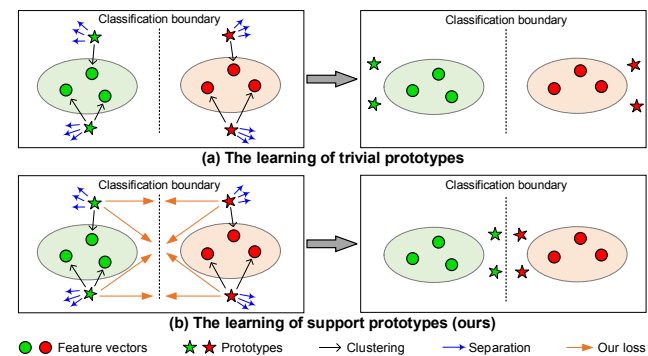

Figure 1. The difference between the learning of trivial and support prototypes. (a) Trivial prototypes: the separation loss pushes the prototypes of different classes as far as possible from the classification boundary. (b) Support prototypes: our new closeness loss enforces the prototypes of different classes to be as close as possible to the classification boundary.

ture spaces, and the enormous number of learnable parameters. Such lack of interpretability hinders their successful application in fields that require understandable and transparent decisions [40], e.g., disease diagnosis [47, 13], financial risk assessment [33], and autonomous driving [23].

Recently, increasing attention has been dedicated to the development of interpretable deep-learning models [25, 1, 4, 3]. A particularly interesting strategy is the prototype-based gray-box models, e.g., prototypical part network (ProtoPNet) [4, 11]. These methods are inherently interpretable since they can explain the model's decisions by showing image classification activation maps associated with a set of class-specific image prototypes. These prototypes are automatically learned from training samples, with classification score being computed by comparing testing image parts to the learned training prototypes.

ProtoPNet [4] is trained to learn a classifier from a set of class-specific prototypes by minimising the cross-entropy classification loss and two additional regularisation losses, namely: 1) a clustering loss that pulls together training image patches to at least one prototype of its own class; and 2) a separation loss that pushes apart training image

(a) Trivial Prototypes   (b) Support Prototypes   (c) Support vectors from SVM
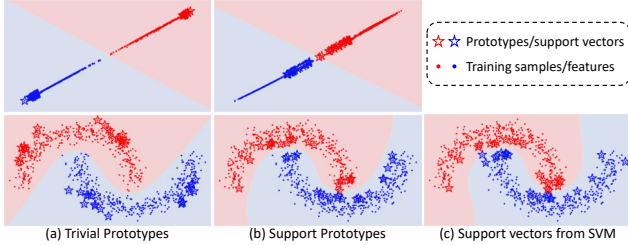
Figure 2. Two-moon results. (a) Trivial prototypes and training samples in the feature (top) and data (bottom) spaces from the original ProtoPNet [4]. (b) Support prototypes and training samples in the feature (top) and data (bottom) spaces from our method. (c) Support vectors and training samples from a Radial Basis Function (RBF) kernel based SVM [7]. In (a) and (b), each prototype is projected onto the nearest training sample in the feature space.

patches from all prototypes of other classes. The combination of these two losses pushes the prototypes as far as possible from the classification boundary, but still within the class distribution, so we call them trivial prototypes, as shown in Fig. 1(a). We also display the trivial prototypes, learned with a feed-forward neural network[1], for the two-moon problem in Fig. 2(a). Notice that these trivial prototypes are located far from the classification boundary. In fine-grained visual classification, the trivial prototypes can mistakenly focus on background regions instead of on the object of interest [42, 41], particularly for those classes with subtle foreground (object) differences but large background variations, as shown in Fig. 3. Different from ProtoPNet's trivial prototypes, the support vector machine (SVM) [7] classifier relies on a set of support vectors that are close to the classification boundary, as in Fig. 2(c). These support vectors are often treated as hard samples. Motivated by SVM, we propose the derivation of support (i.e., hard-to-learn) prototypes for ProtoPNet methods.

In this paper, we propose an alternative learning strategy for ProtoPNet, which forces the learned prototypes to resemble SVM's support vectors and to be located as close as possible to the classification boundary. The strategy is formulated by a new closeness loss that minimises the distance between prototypes of different classes. As shown in Fig. 1(b), our new loss enforces the prototypes to move closer to the classification boundary, as also demonstrated by Fig. 2(b) revealing that the support prototypes produced by the introduction of our new closeness loss are indeed more similar to the support vectors of SVM in Fig. 2(c). Furthermore, to improve the classification accuracy, we propose a new ST-ProtoPNet method that integrates both the support and trivial prototypes. The ST-ProtoPNet leverages the two distinct and complementary sets of prototypes to capture both hard (i.e., close to the boundary) and easy (i.e., far from the boundary) visual features for classification.
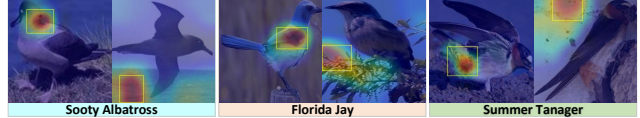

Figure 3. Example prototypes sampled from a VGG19-based ProtoPNet [4]. In each class, the left prototype focuses on object features while the right one captures background.

The major contributions of this work are:

1. We provide the first study that makes an analogy between the prototype learning from ProtoPNet and support vector learning from SVM, where we propose support (i.e., hard-to-learn) prototypes that can improve classification accuracy and interpretability.

2. We present a new ST-ProtoPNet method to exploit both support and trivial prototypes for interpretable image classification, where the two sets of prototypes can provide complementary information to improve classification accuracy.

3. We conduct extensive experiments on three benchmarks, showing that our ST-ProtoPNet outperforms current state-of-the-art (SOTA) methods in terms of classification accuracy and interpretability.

In our experiments, we also demonstrate that the trivial and support prototypes have different characteristics, where the trivial prototypes tend to focus on both local parts of the visual object of interest and the background, while the support prototypes mainly focus on object parts belonging to the visual class of interest.

2. Related Work

2.1. Classification Interpretability

The interpretation of classification results produced by deep neural networks can be achieved by a variety of post-hoc explanation techniques, e.g., explanatory surrogates [34, 56, 44], counterfactual examples [15, 46, 19], and saliency visualisation [45, 55, 59, 43]. Alternatively, prototype-based interpretable techniques can access the model's inner computations. ProtoPNet [4] is the original work that uses class-specific prototypes for interpretable image classification. Similar to ProtoPNet, TesNet [53] constructs class-specific transparent basis concepts on Grassmann manifold for the interpretable classification. Derived from ProtoPNet, Deformable ProtoPNet [11] employs spatially-flexible and deformable prototypes to adaptively capture meaningful object features. In ProtoPShare [42], a data-dependent merge-pruning method is presented to share prototypes among classes, which can reduce the number of prototypes used for classification. In contrast, ProtoPool [41] introduces a fully differentiable prototype assignment strategy to reduce the number of prototypes. In

Proto2Proto [20], a knowledge distillation method is designed to transfer interpretability from a teacher ProtoPNet to a shallow student ProtoPNet. ProtoTree [35] integrates the prototype learning into a binary neural decision tree that can explain its predictions by tracing a decision path throughout the tree. ViT-NeT [24] further establishes the prototype neural tree structure on visual transformers [12].

Because of the ability to self-explain classification results, prototype-based interpretability (e.g., ProtoPNet) has been widely utilised not only in the computer vision applications above, but also in medical imaging [2, 22, 51] and face recognition [48]. However, an open question faced by these methods is if the prototypes being learned are the ideal ones in terms of classification and interpretability.

2.2. SVM vs Prototype-based Classification

To better understand the role of prototypes, we consider the support vector machine (SVM) [7] classifier that finds support vectors to represent classes. More specifically, SVM learns the maximum-margin classifier defined by a classification boundary that maximises the distance to the closest training samples, which are the support vectors for the classes. The testing of SVM consists of computing a weighted similarity between a testing sample and the support vectors. It is interesting to note that the testing of prototype-based classifiers is also based on measuring the similarity between a testing image and a set of class-specific prototypes learned from the training process. Although the testing of SVM and prototype-based classifiers are similar, their training procedures are different. First, the training of a prototype-based classifier learns a fixed number of prototypes [4, 11], while the SVM classifier learns to weight a variable number of support vectors from the training set. Second, in prototype-based classifiers, the learned prototypes tend to be far from the classification boundary, which is contrary to the SVM training objective mentioned above.

The study of deep learning methods from an SVM theoretical perspective is a rich area of research [9, 38, 6], but there are many practical questions that need to be addressed, e.g., how to scale the kernel computation for large-scale datasets, how to shorten the training process [38], and how to integrate deep-learning features with the learning of the SVM classifier. In this paper, our focus is on adapting the learning of ProtoPNet's prototypes to make them similar to SVM's support vectors, by forcing prototypes to be as close as possible to the classification boundary.

2.3. Interpretable Ensemble Classification

Ensemble classification [10] is a classical machine learning approach that combines the results from multiple classifiers, with the goals of improving learning generalisation and classification calibration. The use of interpretable ensemble strategy has been explored in [4, 53, 35, 11, 41],

which is achieved by summing the classification logits of multiple prototype-based classifiers (e.g., ProtoPNets trained with different CNN backbones). In this work, we propose an interpretable ensemble classification by combining the predictions of two ProtoPNets with highly distinctive prototypes (i.e., support and trivial prototypes), which is different from previous studies where the type of prototypes produced by each classifier is very similar given that the same training objective is used for each classifier.

3. Preliminaries

We assume to have a training set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{|\mathcal{D}|}$, where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{H \times W \times R}$ is an image with R colour channels and $\mathbf{y} \in \mathcal{Y} \subset \{0, 1\}^C$ is a one-hot vector representation of the image class label. The interpretable ProtoPNet [4, 11] is trained to learn a set of prototypes $\mathcal{P} = \{\mathbf{p}_m\}_{m=1}^M$, where $\mathbf{p}_m \in \mathbb{R}^{\rho_1 \times \rho_2 \times D}$, with each of the C classes containing M/C prototypes. Without loss of generality, we assume $\rho_1 = \rho_2 = 1$, but the extension to general values is trivial. A typical ProtoPNet comprises four components: a CNN backbone, add-on layers, a prototype layer, and a fully connected (FC) layer. An input image $\mathbf{x}$ is fed to the CNN backbone $f_\theta : \mathcal{X} \to \mathcal{F}$ (parameterised by $\theta \in \Theta$, where $\mathcal{F} \subset \mathbb{R}^{\bar{H} \times \bar{W} \times \bar{D}}$) and then passed on to the add-on layers, denoted by $f_\omega : \mathcal{F} \to \mathcal{V}$ (parameterised by $\omega \in \Omega$), to produce a feature map $\mathbf{V} \in \mathcal{V} \subset \mathbb{R}^{\bar{H} \times \bar{W} \times D}$. The prototype layer computes the similarity between the feature map $\mathbf{V}$ and the M D-dimensional prototypes $\{\mathbf{p}_m\}_{m=1}^M$ to generate M similarity maps $\mathbf{S}_m^{(i,j)} = \text{sim}(\mathbf{V}(i, j, :), \mathbf{p}_m)$, where $i \in \{1, ..., \bar{H}\}$, $j \in \{1, ..., \bar{W}\}$, and $\text{sim}(\cdot, \cdot)$ represents a similarity measure, e.g., cosine similarity [11] and projection metric [53]. The prototype layer outputs M similarity scores from max-pooling $\mathcal{S} = \left\{ \max\limits_{i \in \{1,...,\bar{H}\}, j \in \{1,...,\bar{W}\}} \mathbf{S}_m^{(i,j)} \right\}_{m=1}^M$, which are fed to the FC layer $f_\phi : \mathcal{S} \to \Delta$, parameterised by $\phi \in \Phi$, to produce the classification prediction $\hat{\mathbf{y}} \in \Delta \subset [0, 1]^C$, where Δ denotes the probability space for C classes.

4. ST-ProtoPNet

An overview of our proposed ST-ProtoPNet method is illustrated in Fig. 4, which comprises a shared CNN backbone $f_\theta(\cdot)$, two interpretable ProtoPNet classification branches, namely: 1) the support ProtoPNet represented by add-on layers $f_{\omega^{(s)}}(\cdot)$, prototype layer with support prototypes $\mathcal{P}^{(s)}$, and FC layer $f_{\phi^{(s)}}(\cdot)$ which outputs the classification probability distribution $\hat{\mathbf{y}}^{(s)} \in \Delta$; and 2) the trivial ProtoPNet branch with its add-on layers $f_{\omega^{(t)}}(\cdot)$, trivial prototypes $\mathcal{P}^{(t)}$, and FC layer $f_{\phi^{(t)}}(\cdot)$ that generates probability predictions $\hat{\mathbf{y}}^{(t)} \in \Delta$. The final classification is obtained by combining the classification logits from both the support and trivial ProtoPNets. In our implementation, we construct the support and trivial ProtoPNet mainly based on the orig-
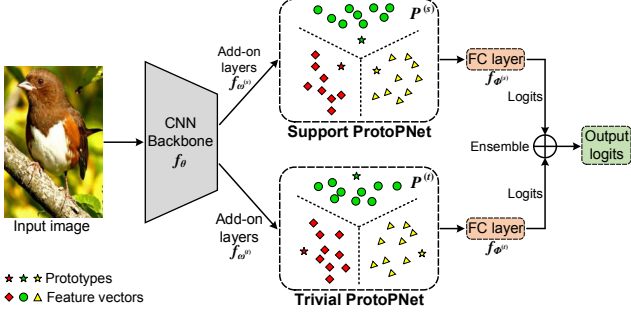
Figure 4. The architecture of our proposed ST-ProtoPNet method for the interpretable image classification.

inal ProtoPNet [4] and TesNet [53], as explained below.

4.1. Support ProtoPNet

The support ProtoPNet is designed to produce support prototypes that are as close as possible to the classification boundary, as shown in Fig. 1(b) and 2(b). The loss function to optimise the support ProtoPNet branch is defined as:

$$
\theta^*, \omega^{(s)*}, \mathcal{P}^{(s)*}, \phi^{(s)*} =
$$
$$
\arg \min_{\theta, \omega^{(s)}, \mathcal{P}^{(s)}, \phi^{(s)}} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \ell_{spt}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}, \phi^{(s)}).
$$
$$
(1)
$$

The loss for each training sample $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ in Eq. (1) above is represented by:

$$
\begin{aligned}
\ell_{spt}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}, \phi^{(s)}) &= \ell_{ce}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}, \phi^{(s)}) \\
&\quad - \lambda_1 \ell_{ct}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}) \\
&\quad + \lambda_2 \ell_{sp}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}) \\
&\quad - \lambda_3 \ell_{cls}(\mathcal{P}^{(s)}) \\
&\quad + \lambda_4 \ell_{ort}(\mathcal{P}^{(s)}),
\end{aligned}
$$
$$
(2)
$$

where λ_1, λ_2, λ_3, and λ_4 are hyper-parameters to balance each term, $\ell_{ce}(\cdot)$ denotes the cross-entropy classification loss, $\ell_{ct}(\cdot)$ and $\ell_{sp}(\cdot)$ represent the clustering and separation losses, respectively, which are introduced to regularise the ProtoPNet's training, as follows:

$$
\ell_{ct}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}) = \max_{\mathbf{p} \in \mathcal{P}_{\mathbf{y}}^{(s)}} \max_{\mathbf{v} \in \mathbf{V}^{(s)}} \text{sim}(\mathbf{v}, \mathbf{p}), \quad (3)
$$

$$
\ell_{sp}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(s)}, \mathcal{P}^{(s)}) = \max_{\mathbf{p} \notin \mathcal{P}_{\mathbf{y}}^{(s)}} \max_{\mathbf{v} \in \mathbf{V}^{(s)}} \text{sim}(\mathbf{v}, \mathbf{p}), \quad (4)
$$

where $\mathbf{V}^{(s)} = f_{\omega^{(s)}}(f_\theta(\mathbf{x}))$ is the feature map extracted from the input image $\mathbf{x}$, $\mathbf{v}$ represents one of the $\bar{H} \times \bar{W}$ feature vectors in $\mathbf{V}^{(s)}$ obtained by matrix vectorisation, $\mathbf{p}$ is a normalised prototype (i.e., unit vector) in $\mathcal{P}^{(s)}$, $\text{sim}(\cdot, \cdot)$

is one of the similarity functions defined in Sec. 3, and $\mathcal{P}_{\mathbf{y}}^{(s)}$ denotes the set of prototypes of class $\mathbf{y}$. The clustering loss in Eq. (3) and separation loss in Eq. (4) aim to learn a meaningful feature space in which the image features of a certain class are clustered around the prototypes of the class, and also well separated from those of other classes.

As mentioned in Sec. 1, the effect of the clustering and separation losses above tend to push the prototypes of different classes as far as possible from the classification boundary, resulting in trivial prototypes, as displayed in Fig. 1(a) and Fig. 2(a). In order to learn the proposed support prototypes, we introduce the following novel closeness loss ℓ_{cls} to explicitly enforce the prototypes of different classes to be close to each other, which is formulated as:

$$
\ell_{cls}(\mathcal{P}^{(s)}) = \sum_{c_1=1}^{C-1} \sum_{c_2=c_1+1}^{C} \min_{\mathbf{p}_m \in \mathcal{P}_{c_1}, \mathbf{p}_n \in \mathcal{P}_{c_2}} \mathbf{p}_m^\top \mathbf{p}_n. \quad (5)
$$

During training, this closeness loss ℓ_{cls} maximises the pair-wise prototype similarity, in the form of dot product $\mathbf{p}_m^\top \mathbf{p}_n$ between different classes in Eq. (5) above, with the goal of pulling the prototypes close to the classification boundary. On the one hand, as the prototypes move gradually towards the classification boundary, they are able to capture harder visual features from training samples. On the other hand, since the prototypes are located near the classification boundary, they can put pressure on the support ProtoPNet's feature learning (i.e., enforce it to learn highly discriminative feature representations for accurate classification), which is beneficial to extract more meaningful semantic information from training samples.

Ideally, each prototype of a class should focus on unique object parts of the training images (e.g., head, tail, and claw of birds), so that the prototypes can represent rich and diverse visual patterns. However, there is no particular constraints to guarantee such prototype diversity and the issue of prototype duplication [11] often occurs in the ProtoPNet family of models. To encourage the intra-class prototype diversity, we employ an orthonormality loss [53] so that prototypes within a class can represent dissimilar visual patterns of training samples, which is defined as:

$$
\ell_{ort}(\mathcal{P}^{(s)}) = \sum_{c=1}^{C} \|\mathbf{P}_c^\top \mathbf{P}_c - \mathbf{I}_{M/C}\|_F^2, \quad (6)
$$

where $\|\cdot\|_F^2$ represents Frobenius norm, $\mathbf{P}_c \in D \times \mathbb{R}^{(M/C)}$ stands for a matrix composed of the prototypes of class c (prototypes in each column of $\mathbf{P}_c$ are normalised), and $\mathbf{I}_{M/C}$ is an identity matrix of size $M/C \times M/C$.

4.2. Trivial ProtoPNet

As described in Sec. 4.1, the support ProtoPNet is developed to learn support (i.e., hard-to-learn) prototypes by

forcing them to be close to the classification boundary. Considering that training samples contain not only hard visual features but also important easy ones that the support prototypes cannot completely capture, we propose to also learn trivial prototypes to provide complementary classification information, and exploit both the support and trivial prototypes for improved interpretable classification.

The loss objective to optimise the trivial ProtoPNet branch is defined as follows:

$$\theta^*, \omega^{(t)*}, \mathcal{P}^{(t)*}, \phi^{(t)*} =$$
$$\arg \min_{\theta, \omega^{(t)}, \mathcal{P}^{(t)}, \phi^{(t)}} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \ell_{trv}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(t)}, \mathcal{P}^{(t)}, \phi^{(t)}).$$
(7)

The loss for each training image $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ in Eq. (7) above is represented by:

$$\begin{aligned}
\ell_{trv}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(t)}, \mathcal{P}^{(t)}, \phi^{(t)}) =\ & \ell_{ce}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(t)}, \mathcal{P}^{(t)}, \phi^{(t)}) \\
& - \lambda_1 \ell_{ct}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(t)}, \mathcal{P}^{(t)}) \\
& + \lambda_2 \ell_{sp}(\mathbf{x}, \mathbf{y}, \theta, \omega^{(t)}, \mathcal{P}^{(t)}) \\
& + \lambda_3 \ell_{dsc}(\mathcal{P}^{(t)}) \\
& + \lambda_4 \ell_{ort}(\mathcal{P}^{(t)}),
\end{aligned}$$
(8)

where λ_1, λ_2, λ_3, and λ_4 are hyper-parameters, $\ell_{ce}(\cdot)$ is the cross-entropy loss, the clustering loss ℓ_{ct}, separation loss ℓ_{sp}, and orthonormality loss ℓ_{ort} are the same as in the support ProtoPNet defined in Eq. (3), (4) and (6), respectively.

The trivial ProtoPNet targets the learning of easy prototypes that are far from the classification boundary and have a good discrimination ability. To help achieve this, we introduce a new discrimination loss ℓ_{dsc} to facilitate the inter-class separability between prototypes of different classes. This is formulated by minimising the pair-wise prototype similarities of different classes, as follows:

$$\ell_{dsc}(\mathcal{P}^{(t)}) = \sum_{c_1=1}^{C-1} \sum_{c_2=c_1+1}^{C} \max_{\mathbf{p}_m \in \mathcal{P}_{c_1}, \mathbf{p}_n \in \mathcal{P}_{c_2}} \mathbf{p}_m^\top \mathbf{p}_n. \quad (9)$$

4.3. Training and Testing

Training. Following the training strategies in [4, 11], the training procedure of our ST-ProtoPNet consists of 3 stages: 1) optimisation of the CNN backbone, add-on layers, and prototype layer, using a fixed FC layer initialised with +1.0 and -0.5 for correct and incorrect connection weights, respectively. A warm-up of 5 epochs is involved in this stage by updating only the parameters of add-on layers and prototype layer, with a frozen pre-trained CNN backbone. 2) prototype projection by updating each prototype with its nearest latent training image patch; and 3) optimisation of the FC layer, with an additional L_1 regularisation on the incorrect connection weights (initially fixed at

-0.5). In each stage, we alternate the optimisation of each branch of the ST-ProtoPNet between mini-batches. Notice that our method only brings marginal extra model parameters, computational complexity, and training time since the CNN backbone is shared and optimised by both branches.

Testing. To exploit the complementary results from both branches of ST-ProtoPNet, its final classification is obtained from the summed logits predicted by the two branches. It is worth noticing that this ensemble strategy introduces no loss of interpretablity but improved accuracy.

5. Experiments

We perform experiments on three fine-grained classification benchmark datasets: CUB-200-2011 [50], Stanford Cars [27], and Stanford Dogs [21]. To achieve fair comparison, we follow previous studies [4, 53] by applying offline data augmentations (e.g., random rotation, skew, shear, and left-right flip) on the cropped CUB and cropped Cars datasets (using the bounding boxes provided). We also validate our method on the full (i.e., uncropped) CUB and Dogs datasets, and employ the same online data augmentation methods (e.g., random affine transformation and left-right flip) as used in Deformable ProtoPNet [11]. All images are resized to 224×224 pixels as network input.

5.1. Experimental Settings

The proposed ST-ProtoPNet method is evaluated on the following CNN architectures: VGG-16, VGG-19, ResNet-34, ResNet-50, ResNet-152, DenseNet-121, and DenseNet-161. All CNN backbones are pre-trained on ImageNet [8], except for ResNet-50, which is pre-trained on iNaturalist [49] for the experiment on full CUB [11]. The add-on layers include two 1×1 convolutional layers. For simplicity, we utilise the same prototype dimension $D = 64$ for all CNN backbones on the three datasets. For cropped CUB and Cars datasets, following [53], we use 10 prototypes (5 support and 5 trivial) per class and the projection metric in the similarity function $\mathrm{sim}(\cdot, \cdot)$. In full CUB and Dogs datasets, to ensure comparison fairness with Deformable ProtoPNet [11] that uses 10 2×2 (full CUB) and 10 3×3 (full Dogs) deformable prototypes per class, we utilise the same total number of prototypes, i.e., 40 1×1 (20 support and 20 trivial) for full CUB and 90 1×1 (45 support and 45 trivial) for full Dogs. Also, we employ the cosine similarity in $\mathrm{sim}(\cdot, \cdot)$ and obtain 14×14 ($\bar{H} = \bar{W} = 14$) feature maps by upsampling the original 7×7 feature maps via a bi-linear interpolation step, as in [11]. Following previous prototype-based methods [4, 53, 11], we set $\lambda_1 = 0.8$, $\lambda_2 = 0.48$ and 0.08 for the support and trivial ProtoPNet branches respectively, $\lambda_4 = 0.001$. We choose $\lambda_3 = 1.0$ with an ablation provided in the supplementary material.

5.2. Interpretability Evaluation

Rather than performing user-based evaluations [60, 41], whose results are often subjective and difficult to reproduce [54], we leverage the annotated object masks to measure interpretability based on the following metrics:

Content Heatmap (CH) [37]: quantifies the percentage of an activation heatmap that lies within the annotated mask. Hence, we expect this metric to be high.

Outside-Inside Relevance Ratio (OIRR) [29]: calculates the ratio of mean activation outside the object to mean activation inside the object. A low OIRR indicates a method relies more on the object region and less on the context to support its decision, we thus anticipate OIRR to be low.

Intersection over Union (IoU) [5]: measures the mean IoU score, where a threshold of 0.5 is applied on the min-max normalised heatmap to select foreground objects.

Deletion AUC (DAUC) [36]: estimates a decrease in the probability of the predicted class as more and more important/activated pixels are removed. The area under the probability curve is defined as DAUC. Consequently, a sharp drop and a low DAUC mean better interpretation.

Average Inter-class Prototype Distance (AIPD), **Average Inter-class Feature Distance (AIFD)**: we had earlier stated that support prototypes of different classes should be close to each other and trivial prototypes should be far from each other. Thus, we compute the average inter-class cosine distance for prototypes and their nearest local feature representations, respectively. We expect AIPD < AIFD for support ProtoPNet and AIPD > AIFD for trivial ProtoPNet.

5.3. Classification Performance

Table 1 presents the classification accuracy (across 5 runs) of our proposed ST-ProtoPNet on cropped CUB and cropped Cars, where the Baseline is represented by non-interpretable black-box CNN models. As can be seen, our ST-ProtoPNet outperforms other competing methods across all backbones for the task of bird species classification. Also, our method achieves the best results for the car model identification task when using VGG and DenseNet architectures as the CNN backbone. In particular, our VGG19-based ST-ProtoPNet reaches an average accuracy of 83.2% and 91.7% on CUB and Cars, respectively, surpassing other methods with the most improvements across all backbones. Moreover, the support ProtoPNet generally performs better than methods utilising only trivial prototypes (e.g., ProtoP-Net, TesNet, and Trivial ProtoPNet), showing the importance of learning support prototypes for the interpretable classification. It is worth noting that our ST-ProtoPNet produces superior performance over the support ProtoPNet method, indicating that both support and trivial prototypes are useful and can provide complementary information for achieving accurate and interpretable classification.

Table 2 shows the classification results on full CUB and full Dogs. In both datasets, the classification accuracy of the original ProtoPNet method is generally worse than the non-interpretable counterpart (Baseline) for many CNN backbones. On the other hand, the accuracy by the trivial ProtoPNet and support ProtoPNet are substantially better than those by Baseline, ProtoPNet, and Deformable ProtoPNet. However, our ST-ProtoPNet achieves more significant performance gains and exhibits the best accuracy across most backbones, particularly when using a large number of prototypes (i.e., 40 1×1 prototypes per class for CUB and 90 1×1 prototypes per class for Dogs), demonstrating the effectiveness of utilising both the trivial and support prototypes for the interpretable image classification. Additionally, when using a smaller number of prototypes, i.e, 10 1×1 prototypes per class, our ST-ProtoPNet still has competitive classification accuracy across multiple backbones.

We further compare our ST-ProtoPNet with other deep-learning methods that can provide different levels of interpretability on CUB, with results shown in Table 3, where * and ** denote ensemble of models with different backbones. As evident, an ensemble of three ST-ProtoPNets can achieve high accuracy (87.9% on cropped images, 88.2% on full images), outperforming competing methods that are also based on an ensemble of three models (e.g., ProtoTree, TesNet, and ProtoPool). Moreover, the ensemble of five ST-ProtoPNets outperforms all other competing methods and obtains the best classification accuracy of 88.1% and 88.4% on cropped and full CUB images, respectively. More results on Cars and Dogs are given in the supplementary material.

5.4. Interpretability Comparison

We assess the model interpretabilty on full CUB using the annotated bird segmentation mask[2]. Quantitative results on the test set are given in Table 4, where all methods are based on the VGG19 backbone. We use GradCAM [43] for the non-interpretable baseline. For prototype-based methods, we average the activation map of all prototypes of a class to compute the metrics. We can see our proposed support ProtoPNet can effectively improve the interpretaiblity in all measures, showing the interpretations produced by our support prototypes are more likely to be object-dependent and focus less on context cues. Also, our ST-ProtoPNet method shows better interpretaiblity results than the support ProtoPNet in terms of CH, OIRR, and DAUC. Compared with the original ProtoPNet [4], our ST-ProtoPNet obtains significant interpretability improvements. We show some example activation maps in the supplementary material, and an experiment with prototype pruning is also provided.

Table 5 presents the computed AIPD and AIFD for the support and trivial ProtoPNet on cropped CUB, using VGG19 and ResNet34 as CNN backbones. As evident, the AIPD is indeed smaller than AIFD for the support ProtoP-

---

[2]http://www.vision.caltech.edu/datasets/

Method	CUB						Cars					
	VGG16	VGG19	ResNet34	ResNet152	Dense121	Dense161	VGG16	VGG19	ResNet34	ResNet152	Dense121	Dense161
Baseline	73.3 ± 0.2	74.7 ± 0.4	82.2 ± 0.3	80.8 ± 0.4	81.8 ± 0.1	82.1 ± 0.2	87.3 ± 0.4	88.5 ± 0.3	**92.6 ± 0.3**	**92.8 ± 0.4**	92.0 ± 0.3	92.5 ± 0.3
ProtoPNet [4]	77.2 ± 0.2	77.6 ± 0.2	78.6 ± 0.1	79.2 ± 0.3	79.0 ± 0.2	80.8 ± 0.3	88.3 ± 0.2	89.4 ± 0.2	88.8 ± 0.1	88.5 ± 0.3	87.7 ± 0.1	89.5 ± 0.2
TesNet [53]	81.3 ± 0.2	81.4 ± 0.1	82.8 ± 0.1	82.7 ± 0.2	84.8 ± 0.2	84.6 ± 0.3	90.3 ± 0.2	90.6 ± 0.2	90.9 ± 0.2	92.0 ± 0.2	91.9 ± 0.3	92.6 ± 0.3
Trivial ProtoPNet	80.8 ± 0.2	81.2 ± 0.2	82.5 ± 0.2	83.1 ± 0.3	83.9 ± 0.3	84.6 ± 0.3	90.1 ± 0.2	90.7 ± 0.2	91.1 ± 0.2	91.5 ± 0.2	91.4 ± 0.3	92.4 ± 0.3
Support ProtoPNet	81.7 ± 0.2	81.8 ± 0.3	83.0 ± 0.1	83.6 ± 0.2	84.7 ± 0.2	85.2 ± 0.3	90.9 ± 0.2	90.8 ± 0.2	91.0 ± 0.2	91.8 ± 0.2	91.7 ± 0.2	92.7 ± 0.3
ST-ProtoPNet (ours)	**82.9 ± 0.2**	**83.2 ± 0.2**	**83.5 ± 0.1**	**84.1 ± 0.2**	**85.4 ± 0.2**	**86.1 ± 0.2**	**91.1 ± 0.2**	**91.7 ± 0.2**	91.4 ± 0.1	92.0 ± 0.2	**92.3 ± 0.3**	92.7 ± 0.2

Table 1. Classification accuracy (%) on cropped CUB-200-2011 and Stanford Cars by competing methods using different CNN backbones.

Method	# Prototype	CUB							# Prototype	Dogs						
		VGG16	VGG19	ResNet34	ResNet50	ResNet152	Dense121	Dense161		VGG16	VGG19	ResNet34	ResNet50	ResNet152	Dense121	Dense161
Baseline	–	70.9	71.3	76.0	78.7	79.2	78.2	80.0	–	75.6	77.3	81.1	83.1	85.2	81.9	84.1
ProtoPNet [4]	1×1p, 10pc	70.3	72.6	72.4	81.1	74.3	74.0	75.4	1×1p, 10pc	70.7	73.6	73.4	76.4	76.2	72.0	77.3
ProtoPNet [4]	1×1p, 40pc	72.9	74.2	74.1	84.8	76.0	76.6	78.5	1×1p, 90pc	73.9	75.3	76.1	78.1	79.7	75.4	78.8
TesNet [53]	1×1p, 10pc	75.8	77.5	76.2	86.5	79.0	80.2	79.6	1×1p, 10pc	74.3	77.1	80.1	82.4	83.8	80.3	83.8
TesNet [53]	1×1p, 40pc	77.6	79.2	76.5	87.3	80.1	80.9	81.3	1×1p, 90pc	78.5	79.6	81.2	83.3	84.5	82.1	85.2
Deformable ProtoPNet [11]	2×2p, 10pc	75.7	76.0	76.8	86.4	79.6	79.0	81.2	3×3p, 10pc	75.8	77.9	80.6	82.2	86.5	80.7	83.7
Trivial ProtoPNet	1×1p, 40pc	80.0	79.5	77.5	87.2	80.8	81.1	82.1	1×1p, 90pc	78.6	80.4	82.6	85.0	87.0	82.3	85.9
Support ProtoPNet	1×1p, 40pc	80.4	80.0	**78.4**	87.5	80.2	81.5	82.4	1×1p, 90pc	79.0	80.6	83.0	85.1	**87.3**	82.6	86.2
ST-ProtoPNet (ours)	1×1p, 10pc	76.8	77.6	77.4	86.6	78.7	78.6	80.6	1×1p, 10pc	74.2	77.2	80.8	84.0	85.3	79.4	84.4
ST-ProtoPNet (ours)	1×1p, 40pc	**81.0**	**80.2**	78.2	**88.0**	**81.2**	**81.8**	**82.7**	1×1p, 90pc	**79.1**	**80.9**	**83.4**	**85.7**	87.2	**82.9**	**86.6**

Table 2. Classification accuracy (%) on full CUB-200-2011 and Stanford Dogs datasets by competing approaches using different CNN backbones, where $\rho_1 \times \rho_2$p denotes the spatial shape of prototypes and $k$pc represents k prototypes per class.

Interpretability level	Method	Accuracy (%)
None	B-CNN [32]	85.1 (b) 84.1 (f)
Object-level attention	CAM [59]	70.5 (b) 63.0 (f)
	CSG [31]	**82.6** (b) **78.5** (f)
Part-level attention	PA-CNN [26]	82.8 (b) –
	MG-CNN [52]	**83.0** (b) 81.7 (f)
	MA-CNN [57]	– 86.5 (f)
	RA-CNN [14]	– 85.3 (f)
	TASN [58]	– **87.0** (f)
Part-level attention + Prototypes	Region [17]	81.5 (b) 80.2 (f)
	ProtoPNet* [4]	84.8 (b) 81.1 (f)
	ProtoTree* [35]	– 86.6 (f)
	TesNet* [53]	86.2 (b) 83.5 (f)
	ProtoPool* [41]	87.5 (b) –
	ST-ProtoPNet* (ours)	87.9 (b) 88.2 (f)
	ProtoTree** [35]	– 87.2 (f)
	Deformable ProtoPNet** [11]	– 87.8 (f)
	ProtoPool** [41]	87.6 (b) –
	ST-ProtoPNet** (ours)	**88.1** (b) **88.4** (f)

Table 3. Classification accuracy and interpretability level of different methods on CUB-200-2011. "b" and "f" denote the model is trained and tested on cropped and full images, respectively. *: Ensemble of three models. **: Ensemble of five models.

Metric	GradCAM [43]	ProtoPNet [4]	TesNet [53]	DefProto [11]	TrvProto	SptProto	ST-Proto
CH (%, ↑)	52.46	48.66	59.38	52.09	63.05	63.87	**66.43**
IoU (%, ↑)	39.91	38.03	36.92	40.77	37.74	**42.04**	41.05
OIRR (%, ↓)	37.01	37.26	38.97	28.68	34.48	28.69	**28.09**
DAUC (%, ↓)	7.01	7.39	5.86	5.99	6.06	5.80	**5.74**

Table 4. Quantitative interpretability results on full CUB test set. DefProto = Deformable ProtoPNet, TrvProto = Trivial ProtoPNet, SptProto = Support ProtoPNet, ST-Proto = ST-ProtoPNet.

VGG19				ResNet34			
Support ProtoPNet		Trivial ProtoPNet		Support ProtoPNet		Trivial ProtoPNet	
AIPD	AIFD	AIPD	AIFD	AIPD	AIFD	AIPD	AIFD
0.7259	0.9264	0.9987	0.9232	0.6541	0.8573	1.000	0.9481

Table 5. Average inter-class prototype distance (AIPD) and average inter-class feature distance (AIFD) for the support and trivial ProtoPNet trained on cropped CUB.
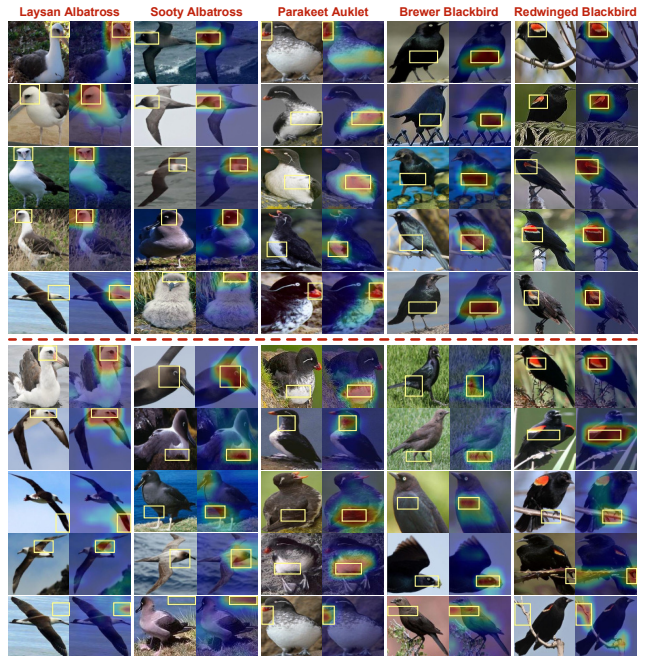

Figure 5. The support (top) and trivial (bottom) prototypes from cropped CUB. In each pair, the first column shows the original image with a prototype marked in a yellow bounding box, the second column is the prototype's corresponding activation map.

Net while AIPD is larger than AIFD for the trivial ProtoPNet. This result indicates that the support prototypes of different classes lie closer than their local feature representations and are more inclined to focus on visually similar (i.e., hard-to-learn) object parts of different classes.

5.5. Visualisation Analysis

To explore the differences between the support and trivial prototypes, we select 5 categories of birds with visu- ally similar features from cropped CUB to train the support and trivial ProtoPNet methods, with the learned prototypes shown in Fig. 5. We notice the support prototypes can cap-
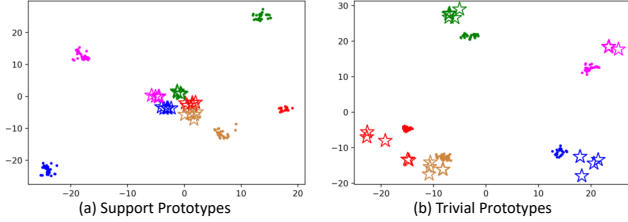
Figure 6. T-SNE results of support and trivial prototypes. The prototypes and their nearest latent training features are marked with stars and dots, respectively. We show results before the prototype projection stage to better visualise the relation between the prototypes and features. Each colour represents a different class.

ture subtle and fine visual features of different classes and they only focus on relevant bird parts, e.g., head and belly. This is reasonable since our algorithm is designed to produce prototypes that are as close as possible to each other, where the image prototypical parts should not only be discriminative but also share visually similar features among classes. By contrast, the trivial prototypes tend to focus not only on the relevant bird parts but also the background regions. For example, some trivial prototypes of the Laysan Albatross and Sooty Albatross classes capture the sea surface as they often appear with the sea background. We argue that this is because the trivial ProtoPNet may treat the background as an easy pattern to learn, focusing less on the object's visual parts of the class. In Fig. 6, we show the t-SNE result from the 5-category bird classification, where we note the support prototypes of different classes are located closer to each other, in comparison with the trivial prototypes.

Fig. 7 shows an example of the interpretable reasoning for our ST-ProtoPNet in classifying a testing bird image. As evident, each ProtoPNet branch calculates its own classification logits (weighted sum of similarity scores), which is then combined to generate the final prediction. Specifically, when classifying a Parakeet Auklet, the support prototypes are quite active on the bird's beak and belly. Meanwhile, the trivial prototypes have high activations on the bird's lower surface and neck. In this case, the support ProtoPNet obtains a relatively higher similarity score (22.925), compared with the trivial branch (20.313). Note that our ST-ProtoPNet exploits both the support and trivial prototypes to capture much richer representations of the object from different perspectives, which enables the realisation of complementary interpretations. More examples on Cars and Dogs are shown in the supplementary material.

5.6. Ablation Study

The closeness and discrimination losses. To validate the effectiveness of our proposed closeness loss in Eq. (5) and discrimination loss in Eq. (9), we first conduct ablation studies on full CUB and full Dogs by using ResNet50 and ResNet34 as the CNN backbone, respectively. Results are listed in Table 6. We can observe that both the close-
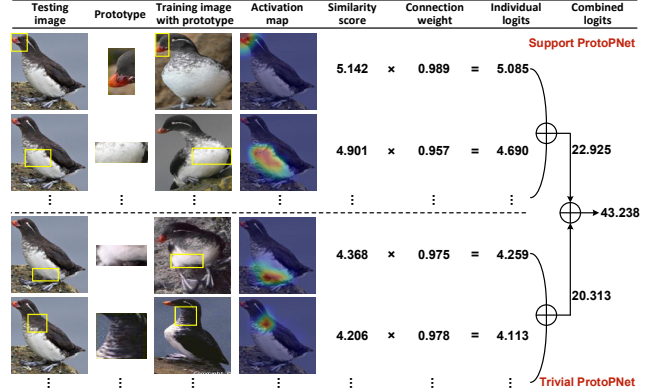

Figure 7. An example of the interpretable reasoning of our ST-ProtoPNet for classifying a testing Parakeet Auklet image.

Method	ℓ_{ct}	ℓ_{sp}	ℓ_{ort}	ℓ_{dsc}	ℓ_{cls}	Accuracy (%) CUB	Dogs
Baseline	✓	✓	✓			86.5	80.9
Trivial ProtoPNet	✓	✓	✓	✓		87.2	82.6
Support ProtoPNet	✓	✓	✓		✓	87.5	83.0
ST-ProtoPNet (ours)	✓	✓	✓	✓	✓	88.0	83.4

Table 6. Ablation analysis of the closeness loss ℓ_{cls} in Eq. (5) to learn support prototypes, and discrimination loss ℓ_{dsc} in Eq. (9) to learn trivial prototypes on full CUB-200-2011 and Stanford Dogs.

ness and discrimination losses can improve the accuracy, compared with the Baseline ProtoPNet method trained with only clustering, separation, and orthonormality losses. Note that the closeness loss introduces a larger performance improvement, which is attributed to the learning of support (i.e., hard-to-learn) prototypes.

Combining Support and Trivial Prototypes. We also investigate the importance of integrating the two complementary sets of support and trivial prototypes for improved classification. To achieve this, we first train a two-branch model where both branches learn the same type of prototypes and the final result is produced by the ensemble of them (Trivial Ensemble and Support Ensemble). Besides, for our ST-ProtoPNet, we also provide results of its individual branches (Trivial Branch and Support Branch). Table 7 shows the experimental results on cropped CUB. We can notice that combining the two different types of prototypes (ST-ProtoPNet) achieves superior performance over combining only the same type of prototypes (Trivial Ensemble and Support Ensemble), indicating that our performance improvements are from not only the ensemble strategy but also the two complementary sets of prototypes. Also, ST-ProtoPNet indeed exhibits higher accuracy than its individual branches, further verifying that the results from the two branches are complementary, and the combination of them is effective to improve the final classification accuracy.

6. Conclusion and Future Work

In this paper, we proposed the ST-ProtoPNet to exploit both support (i.e., hard-to-learn) and trivial (i.e., easy-to-learn) prototypes, where the two sets of prototypes can pro-

Method	VGG16	VGG19	ResNet34	ResNet152	Dense121	Dense161
Trivial Ensemble	81.4 ± 0.3	81.8 ± 0.2	82.7 ± 0.2	83.2 ± 0.3	84.4 ± 0.2	85.0 ± 0.3
Support Ensemble	82.1 ± 0.2	82.4 ± 0.3	83.0 ± 0.2	83.7 ± 0.3	84.8 ± 0.2	85.5 ± 0.2
Trivial Branch	81.0 ± 0.2	81.1 ± 0.3	82.4 ± 0.2	82.9 ± 0.3	84.1 ± 0.3	84.8 ± 0.3
Support Branch	81.5 ± 0.3	81.8 ± 0.3	82.8 ± 0.2	83.4 ± 0.3	84.6 ± 0.2	85.4 ± 0.2
ST-ProtoPNet (ours)	**82.9 ± 0.2**	**83.2 ± 0.2**	**83.5 ± 0.1**	**84.1 ± 0.2**	**85.4 ± 0.2**	**86.1 ± 0.2**

Table 7. Ablation study of the combination of support and trivial prototypes for improved classification on cropped CUB-200-2011.

vide complementary results for the interpretable image classification. Our ST-ProtoPNet is a general approach that can be easily applied to existing prototype-based interpretable models. One limitation for our method is that we empirically adopt the same number of support and trivial prototypes and the same total number of prototypes for each class. Considering the different learning difficulties and imbalanced training samples among classes in other real-world datasets, e.g., ImageNet [8], a better way to adaptively learn a flexible number of support and trivial prototypes is needed and deserves to be further investigated in our future work. Moreover, given that we mimic the behaviour of the support vectors of SVM classifier to obtain the support prototypes by forcing them to be as close as possible to the classification boundary, we plan to develop new methods to learn prototypes with gradient-based kernel techniques, e.g., neural tangent kernel [18] and path kernel [9].

References

[1] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yinhao Ren, Joseph Y Lo, and Cynthia Rudin. A case-based interpretable deep learning model for classification of mass lesions in digital mammography. *Nature Machine Intelligence*, 3(12):1061–1070, 2021.

[3] Moritz Böhle, Mario Fritz, and Bernt Schiele. B-cos networks: Alignment is all we need for interpretability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10329–10338, 2022.

[4] 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.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[6] Yilan Chen, Wei Huang, Lam Nguyen, and Tsui-Wei Weng. On the equivalence between neural network and support vector machine. *Advances in Neural Information Processing Systems*, 34:23478–23490, 2021.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[9] Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020.

[10] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

[11] Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10265–10275, 2022.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[13] Leyuan Fang, Chong Wang, Shutao Li, Hossein Rabbani, Xiangdong Chen, and Zhimin Liu. Attention to lesion: Lesion-aware convolutional neural network for retinal optical coherence tomography image classification. *IEEE transactions on medical imaging*, 38(8):1959–1970, 2019.

[14] 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.

[15] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] 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.

[18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Eoin M Kenny and Mark T Keane. On generating plausible counterfactual and semi-factual explanations for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11575–11585, 2021.

[20] Monish Keswani, Sriranjani Ramakrishnan, Nishant Reddy, and Vineeth N Balasubramanian. Proto2proto: Can you recognize the car, the way i do? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10233–10243, 2022.

[21] 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.

[22] Eunji Kim, Siwon Kim, Minji Seo, and Sungroh Yoon. Xprotonet: diagnosis in chest radiography with global and local explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15719–15728, 2021.

[23] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2942–2950, 2017.

[24] Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. Vit-net: Interpretable vision transformers with neural tree decoder. In *International Conference on Machine Learning*, pages 11162–11172. PMLR, 2022.

[25] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[26] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[27] 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.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[29] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Klaus-Robert Muller, and Wojciech Samek. Analyzing classifiers: Fisher vectors and deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2912–2920, 2016.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[31] 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.

[32] 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.

[33] Rong Liu, Feng Mai, Zhe Shan, and Ying Wu. Predicting shareholder litigation on insider trading from financial text: An interpretable deep learning approach. *Information & Management*, 57(8):103387, 2020.

[34] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[35] 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.

[36] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[37] Vipin Pillai and Hamed Pirsiavash. Explainable models with consistent interpretations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2431–2439, 2021.

[38] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[40] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[41] 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.

[42] Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. Protoshare: Prototypical parts sharing for similarity discovery in interpretable image classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1420–1430, 2021.

[43] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[44] Vivswan Shitole, Fuxin Li, Minsuk Kahng, Prasad Tadepalli, and Alan Fern. One explanation is not enough: structured attention graphs for image classification. *Advances in Neural Information Processing Systems*, 34:11352–11363, 2021.

[45] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[46] Damien Teney, Ehsan Abbasnedjad, and Anton van den Hengel. Learning what makes a difference from counterfactual examples and gradient supervision. In *European Conference on Computer Vision*, pages 580–599. Springer, 2020.

[47] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE*

Transactions on Neural Networks and Learning Systems, 32(11):4793–4813, 2020.

[48] Loc Trinh, Michael Tsang, Sirisha Rambhatla, and Yan Liu. Interpretable and trustworthy deepfake detection via dynamic prototypes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1973–1983, 2021.

[49] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2018.

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

[51] Chong Wang, Yuanhong Chen, Yuyuan Liu, Yu Tian, Fengbei Liu, Davis J McCarthy, Michael Elliott, Helen Frazer, and Gustavo Carneiro. Knowledge distillation to ensemble global and interpretable prototype-based mammogram classification models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 14–24. Springer, 2022.

[52] Dequan Wang, Zhiqiang Shen, Jie Shao, Wei Zhang, Xiangyang Xue, and Zheng Zhang. Multiple granularity descriptors for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2399–2406, 2015.

[53] 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.

[54] Pei Wang and Nuno Vasconcelos. A generalized explanation framework for visualization of deep learning model predictions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[55] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[56] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. Interpreting cnn knowledge via an explanatory graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[57] 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.

[58] 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.

[59] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[60] Andrea Zunino, Sarah Adel Bargal, Riccardo Volpi, Mehrnoosh Sameki, Jianming Zhang, Stan Sclaroff, Vittorio Murino, and Kate Saenko. Explainable deep classification models for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3233–3242, 2021.