Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes

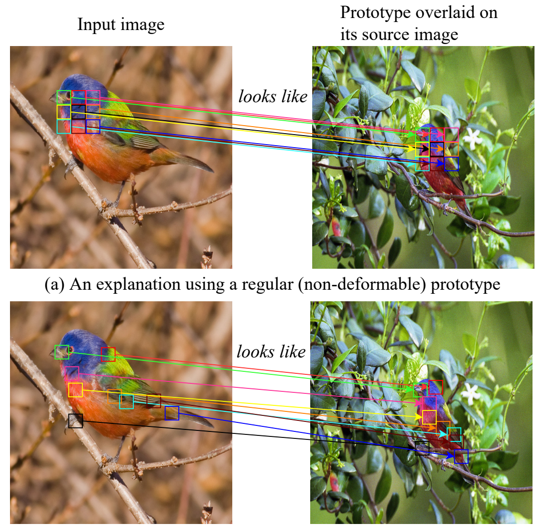
Jon Donnelly
University of Maine
jonathan.donnelly@maine.edu

Alina Jade Barnett
Duke University
alina.barnett@duke.edu

Chaofan Chen
University of Maine
chaofan.chen@maine.edu

Abstract

We present a deformable prototypical part network (Deformable ProtoPNet), an interpretable image classifier that integrates the power of deep learning and the interpretability of case-based reasoning. This model classifies input images by comparing them with prototypes learned during training, yielding explanations in the form of "this looks like that." However, while previous methods use spatially rigid prototypes, we address this shortcoming by proposing spatially flexible prototypes. Each prototype is made up of several prototypical parts that adaptively change their relative spatial positions depending on the input image. Consequently, a Deformable ProtoPNet can explicitly capture pose variations and context, improving both model accuracy and the richness of explanations provided. Compared to other case-based interpretable models using prototypes, our approach achieves state-of-the-art accuracy and gives an explanation with greater context. The code is available at https://github.com/jdonnelly36/Deformable-ProtoPNet.



(a) An explanation using a regular (non-deformable) prototype

(b) An explanation using a deformable prototype

Figure 1. How an input image of a painted bunting is compared with (a) a regular (non-deformable) prototype and (b) a deformable prototype of the painted bunting class (overlaid on its source image).

1. Introduction

Machine learning has been adopted in many domains, including high-stakes applications such as healthcare [2, 29], finance [46], and criminal justice [3]. In these critical domains, interpretability is essential in determining whether we can trust predictions made by machine learning models. In computer vision, there is a growing stream of research that aims to produce accurate yet interpretable image classifiers by integrating the power of deep learning and the interpretability of case-based reasoning [2, 4, 28, 45]. These models learn a set of *prototypes* from training images, and make predictions by comparing parts of the input image with prototypes learned during training. This enables explanations of the form "this is an image of a painted bunting, because *this* part of the image looks like *that* prototypical part of a painted bunting," as in Figure 1(a). However, existing prototype-based models for computer vision use spatially rigid prototypes, which cannot explicitly account for geometric transformations or pose variations of objects.

Inspired by recent work on modeling geometric transformations in convolutional neural networks [5, 16, 17, 53], we propose a *deformable prototypical part network* (*Deformable ProtoPNet*), a case-based interpretable neural network that provides spatially flexible *deformable prototypes*. In a Deformable ProtoPNet, each prototype is made up of several prototypical parts that *adaptively change their relative spatial positions* depending on the input image. This enables each prototype to detect object features with a higher tolerance to spatial transformations, as the parts within a prototype are allowed to move. Figure 1(b) illustrates the idea of a deformable prototype; when an input image is compared with a deformable prototype, the prototypical parts within the deformable prototype adaptively change their relative spatial positions to detect similar parts of the input image. Consequently, a Deformable ProtoPNet can explicitly capture pose variations, and improve both model accuracy and the richness of explanations provided.

The main contributions of our paper are as follows: (1) We developed the first prototypical case-based interpretable neural network that provides spatially flexible deformable prototypes. (2) We improved the accuracy of case-based interpretable neural networks by introducing angular margins to the training algorithm. (3) We showed that Deformable ProtoPNet can achieve state-of-the-art accuracy on the CUB-200-2011 bird recognition dataset [47] and the Stanford Dogs [18] dataset.

2. Related Work

There are two general approaches to interpreting deep neural networks: (1) explaining trained neural networks *posthoc*; and (2) building inherently interpretable neural networks that can explain themselves. *Posthoc* explanation techniques (e.g., model approximations using interpretable surrogates [26, 33], activation maximizations [9, 30, 48], saliency visualizations [1, 35, 36, 38–40, 49]) do not make a neural network *inherently interpretable*, as *posthoc* explanations are not used by the original network during prediction, and may not be faithful to what the original network computes [34].

A Deformable ProtoPNet uses case-based reasoning with prototypes to build an *inherently interpretable network*. This idea was explored in [20] and carried further in [4], where a prototypical part network (ProtoPNet) was introduced. ProtoPNet uses the similarity scores between an input image and the learned prototypes to generate explanations for its predictions in the form of "this looks like that," as in Figure 1(a). The ProtoPNet model has been extended multiple times [27, 28, 45]. We build our Deformable ProtoPNet upon the ProtoPNet and TesNet models [45]. TesNet uses a cosine similarity metric to compute similarities between image patches and prototypes in a latent space, and introduces loss terms to encourage the prototype vectors within a class to be orthogonal to each other and to separate the latent spaces of different classes.

All previous prototype-based image classifiers use spatially rigid prototypes. In contrast, Deformable ProtoPNet is the first network to use spatially flexible deformable prototypes, where each prototype consists of several prototypical parts that adaptively change their relative spatial positions depending on the input image (Figure 1(b)). In this way, our Deformable ProtoPNet can capture pose variations and offer a richer explanation for its predictions than previous image classifiers that use case-based reasoning.

Deformations and Geometric Transformations. Our work relates closely to previous work modeling object deformations and geometric transformations. One of the early attempts at modeling deformations in computer vision models was provided by Deformable Part Models (DPMs) [10], which model deformations as deviations of object parts from their (heuristically chosen) "anchor" positions. The original DPMs use histogram-of-oriented-gradients (HOG) features [6] to represent objects and their parts, and are trained using latent support vector machines (SVMs). The idea of modeling spatial deformations, encapsulated in DPMs, has been extended into convolutional neural networks (CNNs). The inference algorithm of a DPM was shown to be equivalent to a CNN with distance transform pooling [12], and distance transform pooling was extended into a deformation layer in [31] for pedestrian detection and deformation pooling (def-pooling) layers in a DeepID-Net [32] for generic object detection. More recent developments include spatial transformer networks [16], and networks with active convolutions [17] and deformable convolutions [5, 53]. Spatial transformer networks [16] predict global parametric transformations (e.g., affine transformations) to be applied to an input image or a convolutional feature map, with the goal of normalizing the "pose" of the target object in the image. Active convolutions [17] learn and apply the same deformation to a convolutional filter, when scanning the filter across all spatial locations of the input feature map. Deformable convolutions [5, 53], on the other hand, learn to predict deformations that will be applied to convolutional filters at each spatial location of the input feature map. This means that the deformations are different across spatial locations and input images.

Our Deformable ProtoPNet builds upon the Deformable Convolutional Network [5, 53], by using a similar mechanism to generate offsets for deforming prototypes. However, our Deformable ProtoPNet is different from the Deformable Convolutional Network (and the previous work) in two major ways: (1) our Deformable ProtoPNet offers deformable prototypes whose individual parts can be visualized and understood by human beings; (2) by constraining the image features and the representations of prototypes and prototypical parts to be fixed-length vectors, our Deformable ProtoPNet learns an embedding space that has a geometric interpretation, where image features are clustered around similar prototypes on a hypersphere.

Deep Metric Learning Using Margins. Our work also relates to previous work that performs deep metric learning using cosine [44] or angular margins [8, 23, 24, 43]. These techniques use unit vectors to represent classes in the fully-connected last layer of a neural network, allowing us to reinterpret the logit of a class for a given input as the cosine of the angle between the class vector and the latent representation of the input. A margin can then be introduced to increase the angle between the latent representation of a training example and the vector of its target class during training, decreasing the target class logit during training, forcing the network to "try harder" to further reduce the angle in order to lower the cross entropy loss. At the end of training with margins, the latent representations of the training examples from the same class will be clustered in angular space around the vector of that class, and they will be separated by some angular margin from the latent representations of the training examples from a different class. In training our Deformable ProtoPNet, we apply angular margins to inflate the prototype activations of the incorrect-class prototypes for each training example during training.

3. Deformable Prototypes

3.1. Overview of a Deformable Prototype

We will first discuss the general formulation of a non-deformable prototype, as defined in previous work (e.g., [4]). Let $\mathbf{p}^{(c,l)}$ denote the l-th prototype of class c, represented as a tensor of the shape $\rho_1 \times \rho_2 \times d$ with $\rho = \rho_1\rho_2$ spatial positions, and let $\mathbf{p}^{(c,l)}_{m,n}$ denote the d-dimensional vector at the spatial location (m, n) of the prototype tensor $\mathbf{p}^{(c,l)}$, with $m \in \{-\lfloor\rho_1/2\rfloor, ..., \lfloor\rho_1/2\rfloor\}$ and $n \in \{-\lfloor\rho_2/2\rfloor, ..., \lfloor\rho_2/2\rfloor\}$. (A 3×3 prototype has $\rho_1 = \rho_2 = 3$ and $m, n \in \{-1, 0, 1\}$.) Let $\mathbf{z}$ denote a tensor of image features with shape $\eta_1 \times \eta_2 \times d$, produced by passing an input image through some feature extractor (e.g., a CNN), and let $\mathbf{z}_{a,b}$ denote the d-dimensional vector at the spatial location (a, b) of the image-feature tensor $\mathbf{z}$. In the previous work [4], the prototype's height and the width satisfy $\rho_1 \le \eta_1$ and $\rho_2 \le \eta_2$, and its depth is the same as that of $\mathbf{z}$. We can interpret each prototype as representing a patch in the input image, and we can compare a prototype with each $\rho_1 \times \rho_2$ patch of an image-feature tensor using an L^2-based similarity function. Mathematically, for each spatial position (a, b) in an image-feature tensor $\mathbf{z}$, a regular non-deformable prototype computes its similarity with a $\rho_1 \times \rho_2$ patch of $\mathbf{z}$ centered at (a, b) as:

$$g(\mathbf{z})^{(c,l)}_{a,b} = \text{sim}\left(\sum_m \sum_n \|\mathbf{p}^{(c,l)}_{m,n} - \mathbf{z}_{a+m,b+n}\|_2^2\right), \quad (1)$$

where sim is a function that inverts an L^2-distance (in the latent space of image features) into a similarity measure. In a ProtoPNet [4] and a ProtoTree [28], an L^2-based similarity was used to compare a prototype and an image patch in the latent space, presumably because it is intuitive to think about similarity as "closeness" in a Euclidean space.

In a ProtoPNet [4], a prototype (prototypical part) is a spatially contiguous patch, regardless of the number of its spatial positions ρ. For example, Figure 1(a)(right) illustrates a 3×3 non-deformable prototype that can be used in a ProtoPNet. In a Deformable ProtoPNet, we define a prototypical part within a (deformable) prototype to be a 1×1 patch (of shape $1 \times 1 \times d$) within a prototype tensor (of shape $\rho_1 \times \rho_2 \times d$) (see Figure 2). In particular, we use $\hat{\mathbf{p}}^{(c,l)}$ to denote the l-th deformable prototype of class c, again represented as a tensor of the shape $\rho_1 \times \rho_2 \times d$ with $\rho = \rho_1\rho_2$ spatial positions, and we use $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ to denote the (m, n)-th prototypical part within the deformable prototype $\hat{\mathbf{p}}^{(c,l)}$. Figure 1(b)(right) illustrates a deformable prototype of 9 spatial positions (represented as a $3 \times 3 \times d$ tensor), where each spatial position is viewed as an individual prototypical part that can move around, and represents a semantic concept that is *spatially decoupled* from other prototypical parts. For notational consistency, we use $\hat{\mathbf{z}}$ to denote a tensor of image features that will be compared with a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$, and we use $\hat{\mathbf{z}}_{a,b}$ to denote the d-dimensional vector at the spatial location (a, b) of the image-feature tensor $\hat{\mathbf{z}}$.

In a Deformable ProtoPNet, we require all prototypical parts $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ (a d-dimensional vector) of all deformable prototypes $\hat{\mathbf{p}}^{(c,l)}$ to have the same L^2 length:

$$\|\hat{\mathbf{p}}^{(c,l)}_{m,n}\|_2 = r = 1/\sqrt{\rho}, \quad (2)$$

so that when we represent a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ as a stacked vector of its constituent prototypical parts $\hat{\mathbf{p}}^{(c,l)}_{m,n}$, all deformable prototypes have the same L^2 length, which is equal to $\|\hat{\mathbf{p}}^{(c,l)}\|_2 = \sqrt{\rho r^2} = 1$ (i.e., all deformable prototypes are unit vectors). We also require every spatial location (a, b) of every image-feature tensor $\hat{\mathbf{z}}$ to have the same L^2 length:

$$\|\hat{\mathbf{z}}_{a,b}\|_2 = r = 1/\sqrt{\rho}. \quad (3)$$

With equations (2) and (3), we can rewrite the squared L^2 distance between $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ and $\hat{\mathbf{z}}_{a+m,b+n}$ in equation (1) as: $\|\hat{\mathbf{p}}^{(c,l)}_{m,n} - \hat{\mathbf{z}}_{a+m,b+n}\|_2^2 = \sum_m \sum_n (2r^2 - 2\hat{\mathbf{p}}^{(c,l)}_{m,n} \cdot \hat{\mathbf{z}}_{a+m,b+n})$. With similarity function $\text{sim}(\kappa) = -\kappa/2 - 1$, the similarity (defined in equation (1)) between a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ of shape $\rho_1 \times \rho_2 \times d$ and a $\rho_1 \times \rho_2$ patch, centered at (a, b), of the image-feature tensor $\hat{\mathbf{z}}$ becomes:

$$g(\hat{\mathbf{z}})^{(c,l)}_{a,b} = \sum_m \sum_n \hat{\mathbf{p}}^{(c,l)}_{m,n} \cdot \hat{\mathbf{z}}_{a+m,b+n} \quad (4)$$

before we allow the prototype to deform. Note that equation (4) is equivalent to a convolution between $\hat{\mathbf{p}}^{(c,l)}$ and $\hat{\mathbf{z}}$, but with the added constraints given by equations (2) and (3).

To allow a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ to deform, we introduce offsets to enable each prototypical part $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ to move around when the prototype is applied at a spatial location (a, b) on the image-feature tensor $\hat{\mathbf{z}}$. Mathematically, equation (4) becomes:

$$g(\hat{\mathbf{z}})^{(c,l)}_{a,b} = \sum_m \sum_n \hat{\mathbf{p}}^{(c,l)}_{m,n} \cdot \hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}, \quad (5)$$

where $\Delta_1 = \Delta_1(\hat{\mathbf{z}}, a, b, m, n)$ and $\Delta_2 = \Delta_2(\hat{\mathbf{z}}, a, b, m, n)$ are functions depending on $\hat{\mathbf{z}}$, a, b, m, and n (further explained in Section 3.2). These offsets allow us to evaluate the similarity between a prototypical part $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ and the image feature $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ at a deformed position $(a+m+\Delta_1, b+n+\Delta_2)$ rather than the regular grid position $(a+m, b+n)$. Since Δ_1 and Δ_2 are typically fractional, we use feature interpolation to define image features at fractional positions (discussed in Section 3.2). We further require interpolated image features to have the same L^2 length of r as those image features at regular grid positions, namely:

$$\|\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}\|_2 = r = 1/\sqrt{\rho}. \tag{6}$$

Note that equation (5) is equivalent to a deformable convolution [5, 53] between $\hat{\mathbf{p}}^{(c,l)}$ and $\hat{\mathbf{z}}$, but with the added constraints given by equations (2) and (6).

It is worth noting that similarity defined in equation (5) has a simple geometric interpretation. Let $\theta(\mathbf{v}, \mathbf{w})$ denote the angle between two vectors, and let

$$g(\hat{\mathbf{z}})^{(c,l)}_{a,b,m,n} = \hat{\mathbf{p}}^{(c,l)}_{m,n} \cdot \hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2} \tag{7}$$

denote the contribution of the prototypical part $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ to the similarity score of the prototype. Note that equation (7) is *exactly* equal to $\cos(\theta(\hat{\mathbf{p}}^{(c,l)}_{m,n}, \hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}))$, which is the cosine similarity between $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ and $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$. Since both $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ and $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ have the same L^2 length r (equations (2) and (6)), all prototypical parts and all (interpolated) image features live on a d-dimensional hypersphere of radius r. This means that an interpolated image feature vector $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ is considered similar (has a large cosine similarity) to a prototypical part $\hat{\mathbf{p}}^{(c,l)}_{m,n}$ *only* when the angle between them is small on the hypersphere.

A similar geometric interpretation also holds between an entire deformable prototype and image features at deformed positions. Let $\hat{\mathbf{z}}^{\Delta}_{a,b}$ denote the interpolated image features $\hat{\mathbf{z}}_{a-\lfloor\rho_1/2\rfloor+\Delta_1,b-\lfloor\rho_2/2\rfloor+\Delta_2}$, ..., $\hat{\mathbf{z}}_{a+\lfloor\rho_1/2\rfloor+\Delta_1,b+\lfloor\rho_2/2\rfloor+\Delta_2}$ at ρ deformed positions, stacked into a column vector. Note that $\hat{\mathbf{z}}^{\Delta}_{a,b}$ has L^2 length $\|\hat{\mathbf{z}}^{\Delta}_{a,b}\|_2 = 1$. We can then rewrite equation (5) as:

$$g(\hat{\mathbf{z}})^{(c,l)}_{a,b} = \hat{\mathbf{p}}^{(c,l)} \cdot \hat{\mathbf{z}}^{\Delta}_{a,b} = \cos(\theta(\hat{\mathbf{p}}^{(c,l)}, \hat{\mathbf{z}}^{\Delta}_{a,b})),$$

which is exactly the cosine similarity between $\hat{\mathbf{p}}^{(c,l)}$ and $\hat{\mathbf{z}}^{\Delta}_{a,b}$. Since both $\hat{\mathbf{p}}^{(c,l)}$ and $\hat{\mathbf{z}}^{\Delta}_{a,b}$ are unit vectors, all deformable prototypes and all collections of interpolated image features at ρ deformed positions live on a ρd-dimensional hypersphere of radius 1. This means that a collection of interpolated image features $\hat{\mathbf{z}}^{\Delta}_{a,b}$ is considered similar to an *entire* deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ *only* when the angle between them is small on the hypersphere.

With the similarity between a deformable prototype and a collection of image features at deformed locations defined in equation (5), we now define the similarity score between a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ and an *entire* image-feature tensor $\hat{\mathbf{z}}$ to be its maximum similarity to any set of positions:

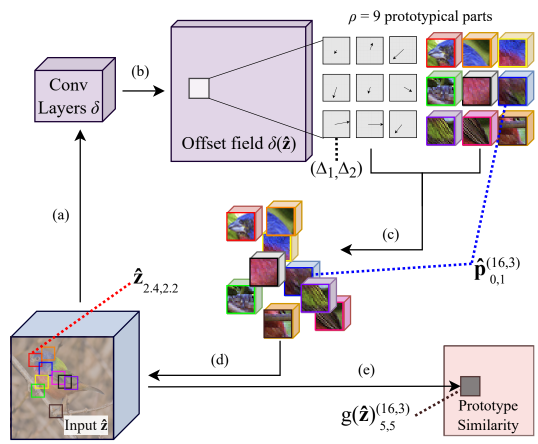


Figure 2. How a deformable prototype is applied to the latent representation of an input image of a painted bunting. (a) The latent input $\hat{\mathbf{z}}$ is fed into the offset prediction function δ to produce (b) a field of offsets. These offsets are used to (c) alter the spatial position of each prototypical part, which are (d) compared to the input to (e) compute prototype similarity according to equation (5).

$$g(\hat{\mathbf{z}})^{(c,l)} = \max_{a,b} g(\hat{\mathbf{z}})^{(c,l)}_{a,b} \tag{8}$$

In our experiments, we trained Deformable ProtoPNets using both 3×3 and 2×2 deformable prototypes. A 2×2 deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ can be implemented as a tensor of the shape $2 \times 2 \times d$ ($\rho_1 = \rho_2 = 2$) with dilation 2 and with $\rho = \rho_1\rho_2 = 4$ prototypical parts at $(m, n) \in \{(-1,-1), (-1,1), (1,-1), (1,1)\}$.

3.2. Offset Generation and Feature Interpolation

As in Figure 2, the offsets used for deformable prototypes are computed using an offset prediction function δ that maps fixed length input features $\hat{\mathbf{z}}$ to an offset field with the same spatial size as $\hat{\mathbf{z}}$. At each spatial center location, this field contains 2ρ components, corresponding to a (Δ_1, Δ_2) pair of offsets for each of the ρ prototypical parts.

The offsets (Δ_1, Δ_2) produced by δ may be integer or fractional. Prior work [5, 16, 17, 53] uses bilinear interpolation to compute the value of these fractional locations. In contrast, we do not use bilinear interpolation because it is not feasible for a Deformable ProtoPNet, as the similarity function specified in equation (5) relies on the assumption that the image feature vector $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ is of L^2 length r; without this assumption, similarities will no longer be dependent only on the angle between a prototype and image features. Bilinear interpolation breaks this assumption, because when interpolating between two vectors that have the same L^2 norm, bilinear interpolation does not preserve the L^2 norm for the interpolated vector. This can be informally explained geometrically: bilinear interpolation chooses a point on the hyperplane that intersects the four interpolated points, meaning that it will never fall on the hypersphere for a fractional location. We use an L^2 norm-preserving interpolation function, introduced in Theorem 3.1, to solve this problem. A proof of Theorem 3.1 can be found in the supplement.

Theorem 3.1. Let $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4 \in \mathbb{R}^n$ be vectors such that $\|\hat{\mathbf{z}}_i\| = r$ for all $i \in 1, 2, 3, 4$ for some constant r, and let $\hat{\mathbf{z}}^2$ denote the element-wise square of a vector. For some constants $\alpha \in [0, 1]$ and $\beta \in [0, 1]$, the bilinear interpolation operation $\mathbf{z}_{\text{interp}} = (1-\alpha)(1-\beta)\hat{\mathbf{z}}_1 + (1-\alpha)\beta\hat{\mathbf{z}}_2 + \alpha(1-\beta)\hat{\mathbf{z}}_3 + \alpha\beta\hat{\mathbf{z}}_4$ does not guarantee that $\|\mathbf{z}_{\text{interp}}\|_2 = r$. However, the L^2 norm-preserving interpolation operation $\mathbf{z}_{\text{interp}} = \sqrt{(1-\alpha)(1-\beta)\hat{\mathbf{z}}_1^2 + (1-\alpha)\beta\hat{\mathbf{z}}_2^2 + \alpha(1-\beta)\hat{\mathbf{z}}_3^2 + \alpha\beta\hat{\mathbf{z}}_4^2}$ guarantees that $\|\mathbf{z}_{\text{interp}}\|_2 = r$.

Finally, the theoretical framework of a deformable prototype requires that every spatial location $\hat{\mathbf{z}}_{a,b}$ and $\hat{\mathbf{p}}_{m,n}^{(c,l)}$ of $\hat{\mathbf{z}}$ and $\hat{\mathbf{p}}^{(c,l)}$ have L^2 length r. In our implementation, we guarantee this by always normalizing and scaling both the image features extracted by a CNN at every spatial location (a, b) of a convolutional output $\mathbf{z}$, as well as every prototypical part of a deformable prototype, to length r, before they are used in computation. Specifically, we compute $\hat{\mathbf{z}}_{a,b} = r\mathbf{z}_{a,b}/\|\mathbf{z}_{a,b}\|_2$ for every spatial location (a, b) of the convolutional output $\mathbf{z}$ and $\hat{\mathbf{p}}_{m,n}^{(c,l)} = r\mathbf{p}_{m,n}^{(c,l)}/\|\mathbf{p}_{m,n}^{(c,l)}\|_2$ for every (m, n)-th part of a deformable prototype. However, this normalization is undefined when $\|\mathbf{p}_{m,n}^{(c,l)}\|_2 = 0$ or $\|\mathbf{z}_{a,b}\|_2 = 0$. This is a problem because zero padding and the ReLU activation function can both create a feature vector $\mathbf{z}$ with L^2 norm 0. We address this problem by appending a uniform channel of a small value $\epsilon = 10^{-5}$ to $\mathbf{p}^{(c,l)}$ and $\mathbf{z}$ prior to normalization. In particular, an all-0 feature vector $\mathbf{z}_{a,b}$ produced by a CNN will become $\begin{bmatrix}0 & \dots & 0 & \epsilon\end{bmatrix}$, which has an L^2 norm of ϵ.

4. Deformable ProtoPNet

Figure 3 gives an overview of the architecture of a Deformable ProtoPNet. A Deformable ProtoPNet consists of a CNN backbone f that maps an image $\mathbf{x}$ to latent image features $\mathbf{z}$, which are normalized to length r at each spatial location into $\hat{\mathbf{z}}$, followed by a deformable prototype layer g that contains deformable prototypes as defined in Section 3.2, and a fully connected last layer h, which combines the similarity scores produced by deformable prototypes into a class score for each class.

4.1. Training

Similar to [4], the training of a Deformable ProtoPNet proceeds in three stages.

Stage 1: Stochastic gradient descent (SGD) of layers before last layer. We perform stochastic gradient descent over the features of f and g while keeping h fixed. By doing so we aim to learn a useful feature space where the image features $\hat{\mathbf{z}}_{a,b}^{\Delta}$ of inputs of class c are clustered around prototypes $\hat{\mathbf{p}}^{(c,l)}$ of the same class, but separated from those of other classes on a hypersphere. To achieve this, we use the cluster and separation losses as in [4] and adapted for the angular space in [45]. The cluster and separation losses are defined as:

$$\ell_{\text{clst}} = -\frac{1}{N}\sum_{i=1}^{N} \max_{\hat{\mathbf{p}}^{(c,l)}:c=y^{(i)}} g(\hat{\mathbf{z}}^{(i)})^{(c,l)} \tag{9}$$

and

$$\ell_{\text{sep}} = \frac{1}{N}\sum_{i=1}^{N} \max_{\hat{\mathbf{p}}^{(c,l)}:c\neq y^{(i)}} g(\hat{\mathbf{z}}^{(i)})^{(c,l)} \tag{10}$$

respectively, where N is the total number of inputs, $\hat{\mathbf{z}}^{(i)}$ is the image feature tensor normalized and scaled at each spatial location for input i, $y^{(i)}$ is the label of $\mathbf{x}^{(i)}$, and all other values are as defined previously.

We were inspired by recent work in margin-based softmax losses [8, 23, 24, 43, 44] to further encourage this clustering structure by modifying traditional cross entropy loss. Specifically, we use a new form of cross entropy: *subtractive margin cross entropy*. This is defined as:

$$\text{CE}^{(-)} = \sum_{i=1}^{N} -\log \frac{\exp\left(\sum_{c,l} w_h^{((c,l),y^{(i)})} g^{(-)}(i)^{(c,l)}\right)}{\sum_{c'} \exp\left(\sum_{c,l} w_h^{((c,l),c')} g^{(-)}(i)^{(c,l)}\right)}, \tag{11}$$

where $w_h^{((c,l),c')}$ denotes the last layer connection between the similarity of prototype $\hat{\mathbf{p}}^{(c,l)}$ and class c',

$$g^{(-)}(i)^{(c,l)} = \begin{cases} g(\hat{\mathbf{z}}^{(i)})^{(c,l)} & \text{if } c = y^{(i)} \\ \max\limits_{a,b} \cos(\lfloor \theta(\hat{\mathbf{p}}^{(c,l)}, \hat{\mathbf{z}}_{a,b}^{\Delta,(i)}) - \phi \rfloor_+) & \text{else} \end{cases} \tag{12}$$

for a fixed margin $\phi = 0.1$, and $\lfloor\ \rfloor_+$ denotes the ReLU function. Subtractive margin cross entropy encourages a well separated feature space by artificially decreasing the angle between a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ of class c and the collection of deformed image features $\hat{\mathbf{z}}_{a,b}^{\Delta,(i)}$ from the i-th training image with $y^{(i)} \neq c$, thereby inflating the cosine similarity between the two and increasing the class score of the incorrect class c. In order to reduce the value of this loss, the network has to try harder to counter the introduced margin ϕ by further increasing the angle between a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ and an image feature $\hat{\mathbf{z}}_{a,b}^{\Delta,(i)}$ of an incorrect class, resulting in a stronger separation between classes on the latent hypersphere.

While the subtractive margin encourages separation between classes, it does not encourage diversity between prototypes within a class and between prototypical parts within

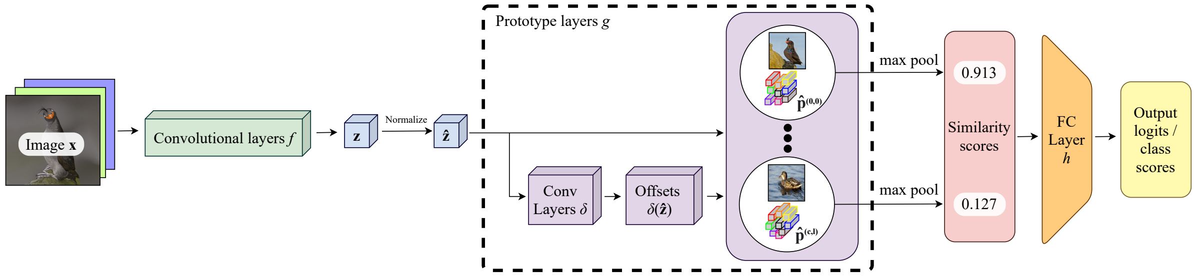


Figure 3. The architecture for Deformable ProtoPNet.

a prototype. In particular, we have observed that deformations without further regularization often result in duplications of prototypical parts within a prototype. Inspired by [45], we discourage this behavior by introducing orthogonality loss between prototypical parts. This is formulated as:

$$\ell_{\text{ortho}} = \sum_c \|\mathbf{P}^{(c)}\mathbf{P}^{(c)\top} - r^2\mathbf{I}^{(\rho L)}\|_F^2, \tag{13}$$

where L is the number of deformable prototypes in class c, ρL is the total number of prototypical parts from all prototypes of class c, $\mathbf{P}^{(c)} \in \mathbb{R}^{\rho L \times d}$ is a matrix with every prototypical part of every prototype from class c arranged as a row in the matrix, and $\mathbf{I}^{(\rho L)}$ is the $\rho L \times \rho L$ identity matrix. The matrix multiplication $\mathbf{P}^{(c)}\mathbf{P}^{(c)\top}$ in equation (13) contains an inner product between every pair of prototypical parts in class c; by encouraging this to be close to the scaled identity matrix $r^2\mathbf{I}^{(\rho L)}$, we encourage the prototypical parts to be orthogonal to one another and thereby increase the diversity of semantic concepts represented by prototypical parts. This loss differs from [45] because it encourages orthogonality at both the prototype and the prototypical part level. Whereas the orthogonality loss in [45] encourages orthogonality between each pair of prototypes within a class, equation (13) encourages orthogonality between *all prototypical parts* within a class. A visualization of the space created by these terms can be seen in Figure 4.

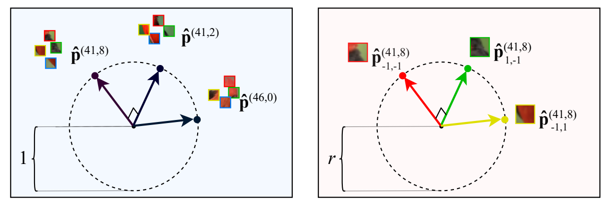
With these loss terms defined, our overall loss term during the first stage of training is:

$$\ell = \text{CE}^{(-)} + \lambda_1\ell_{\text{sep}} + \lambda_2\ell_{\text{clst}} + \lambda_3\ell_{\text{ortho}} \tag{14}$$

where $\lambda_1 = 0.01, \lambda_2 = 0.1$ and $\lambda_3 = 0.1$ are hyperparameters chosen empirically. As in [4], the last layer connection between each deformable prototype and its class is set to 1; all other connections are set to -0.5.

Stage 2: Projection of prototypes. We project each deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ onto the most similar collection of interpolated image features $\hat{\mathbf{z}}_{a,b}^{(\Delta)}$ from some training image $\mathbf{x}$. Mathematically, this is formulated as:

$$\mathbf{p}^{(c,l)} \leftarrow \underset{\hat{\mathbf{z}}_{a,b}^{(\Delta)}}{\text{argmax}}\ \cos\left(\theta(\hat{\mathbf{p}}^{(c,l)} \cdot \hat{\mathbf{z}}_{a,b}^{(\Delta)})\right). \tag{15}$$



(a) Within the same class, prototypes are encouraged to be orthogonal to one another.

(b) Within each prototype, prototypical parts are encouraged be orthogonal to one another.

Figure 4. A representation of the latent space learned by Deformable PrototPNet.

In this projection scheme, we allow projection onto fractional locations and we project all prototypical parts within each prototype onto the *same* training image, which promotes cohesion among parts of a single prototype.

Stage 3: Optimization of the last layer. In this stage, we fix all other model parameters and optimize over the last layer connections h. Let $w_h^{((c,l),c')}$ be defined as previously described. For this stage, we use the loss function:

$$\ell_{\text{last}} = \text{CE} + \lambda_1 \sum_{c,l} \sum_{c' \neq c} |w_h^{((c,l),c')}|, \tag{16}$$

where $\lambda_1 = 10^{-3}$ and CE is standard cross entropy loss. The second term on the right-hand side of equation (16) discourages negative reasoning processes as explained in [4].

4.2. Prototype Visualizations

With prototype projection, we can associate each deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ with a training image $\mathbf{x}$. Before we describe how we map a prototypical part to an image patch, we first define a downsampling factor γ as the ratio of spatial downsampling between the original image and the image-feature tensor. For images of spatial size 224×224 with latent representations of spatial size of 14×14, we have $\gamma = \frac{224}{14} = 16$.

In order to produce a visualization of a deformable prototype on an input image $\mathbf{x}$, we pass the image $\mathbf{x}$ through the network. This enables us to obtain the center location

Model	VGG16	VGG19	Res34	Res50	Res152	Dense121	Dense161
Baseline	70.9	71.3	76.0	78.7	79.2	78.2	80.0
ProtoPNet [4]	70.3*	72.6*	72.4*	81.1*	74.3*	74.0*	75.4*
Def. ProtoPNet (3×3,nd)	67.9	71.1	76.7	85.9	78.2	76.5	79.6
Def. ProtoPNet (3×3)	73.8	75.4	76.7	86.1	78.8	76.4	79.7
Def. ProtoPNet (2×2,nd)	**76.0**	**76.1**	**76.8**	**86.4**	79.2	78.9	80.8
Def. ProtoPNet (2×2)	75.7	76.0	**76.8**	**86.4**	**79.6**	**79.0**	**81.2**

Table 1. Accuracy of Deformable ProtoPNets with 3×3 and 2×2 deformable prototypes, compared to that of the baseline models, ProtoPNets, and Deformable ProtoPNets without deformations (denoted (nd)) across different base architectures. *We retrained ProtoPNets on full images for direct comparison, and report the accuracy numbers on full images here, so the numbers differ from those reported in [4].

(a', b') that produced the best similarity for prototype $\hat{\mathbf{p}}^{(c,l)}$:

$$(a', b') = \underset{a,b}{\operatorname{argmax}}\; g(\hat{\mathbf{z}}_{a,b}^{(\Delta)})^{(c,l)}.$$

We can then retrieve the (Δ_1, Δ_2) offset pair for each prototypical part $\hat{\mathbf{p}}_{m,n}^{(c,l)}$ from the location (a', b') of the offset field $\delta(\hat{\mathbf{z}})$. These values tell us that the prototypical part $\hat{\mathbf{p}}_{m,n}^{(c,l)}$ is compared to the image features at spatial location $(a'+m+\Delta_1, b'+n+\Delta_2)$. To find the corresponding patch in the original image, we create a square bounding box in the original image centered at $(\gamma(a'+m+\Delta_1), \gamma(b'+n+\Delta_2))$ of height and width γ for each prototypical part $\hat{\mathbf{p}}_{m,n}^{(c,l)}$. Since all parts of a deformable prototype must be projected onto (interpolated) image features from the same image, this allows us to view all parts of a prototype on the same image.

4.3. Reasoning Process

Figure 5 shows the reasoning process of a Deformable ProtoPNet in classifying a test image $\mathbf{x}$. In particular, for a given image $\mathbf{x}$ and for every class c, a Deformable ProtoPNet tries to find evidence for $\mathbf{x}$ belonging to class c, by comparing the latent features $\hat{\mathbf{z}}$ with every learned deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ of class c. In Figure 5, our Deformable ProtoPNet tries to find evidence for the test image being a vermilion flycatcher by comparing the image's latent features with each deformable prototype (whose constituent prototypical parts are visualized in the "Prototypical parts" column) of that class. As shown in the figure, the prototypical parts within a deformable prototype, which can be visualized as patches from some training image, can adaptively change their relative spatial positions as the deformable prototype is scanned across the input image to compute a prototype similarity score at each center location according to equation (5). The maximum score across all spatial locations is taken according to equation (8), producing a single "similarity score" for the prototype, which is multiplied by a class connection score from the fully connected layer h to produce a prototype contribution score. These are summed across all prototypes, yielding a final score for the class.

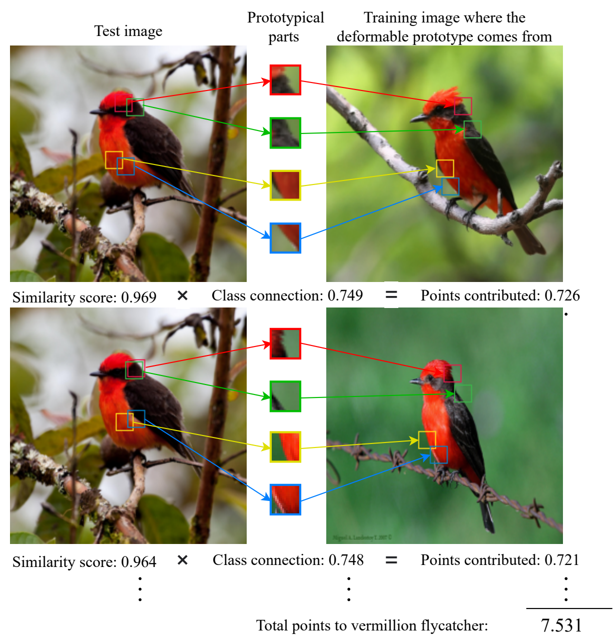


Figure 5. The reasoning process of a Deformable ProtoPNet with 2×2 deformable prototypes.

5. Experiments and Numerical Results

We conducted a case study of our Deformable ProtoPNet on the full (uncropped) CUB-200-2011 bird species classification dataset [47]. We trained Deformable ProtoPNets with 6 3×3 deformable prototypes per class, and Deformable ProtoPNets with 10 2×2 deformable prototypes per class, unless otherwise specified. We ran experiments using VGG [37], ResNet [13], and DenseNet [14] as CNN backbones f. The ResNet-50 backbone was pretrained on iNaturalist [41], and all other backbones were pretrained using ImageNet [7]. See the supplement for more details regarding our experimental setup.

We find that Deformable ProtoPNet can achieve competitive accuracy across multiple backbone architectures. As shown in Table 1, our Deformable ProtoPNet achieves higher accuracy than ProtoPNet [4] and the non-

Margin	Ortho Loss	Deformations	Accuracy
0.1	0	No	86.2
0.1	0	Yes	**86.4**
0.1	0.1	No	**86.4**
0.1	0.1	Yes	**86.4**
0	0	Yes	86.1
0.1	0	Yes	**86.4**
0	0.1	Yes	85.2
0.1	0.1	Yes	**86.4**

Table 2. Ablation studies using 2×2 prototypes.

Interpretability	Model: accuracy on CUB-200-2011
None	B-CNN [22]: 85.1 (bb), 84.1 (f)
Object-level attention	CAM [52]: 70.5 (bb), 63.0 (f)
	CSG [21] 82.6 (bb), 78.5 (f)
Part-level attention	PA-CNN [19]: 82.8 (bb)
	MG-CNN [42]: 83.0 (bb), 81.7 (f)
	MA-CNN [50]: 86.5 (f)
	RA-CNN [11]: 85.3 (f)
	TASN [51]: 87.0 (f)
Part-level attention + prototypes	Region [15]: 81.5 (bb), 80.2 (f)
	ProtoPNet* [4]: 84.8 (bb), 81.1 (f)
	ProtoTree [28]: 82.2 (f)
	ProtoTree [28]: 87.2 (f)**
	TesNet* [45]: 86.2 (bb), 83.5 (f)
	Def. ProtoPNet [nd,3p,6pc]: 85.9 (f)
	Def. ProtoPNet [nd,2p,10pc]: 86.4 (f)
Part-level attention. + prototypes + deformations	Def. ProtoPNet [3p,1pc]: 81.5 (f)
	Def. ProtoPNet [3p,3pc]: 83.7 (f)
	Def. ProtoPNet [3p,6pc]: 86.1 (f)
	Def. ProtoPNet [2p,10pc]: 86.4 (f)
	Def. ProtoPNet [2p,10pc]: 87.8 (f)**
Interpretability	**Model: accuracy on Stanford Dogs**
Part-level attention	FCAN [25]: 84.2
	RA-CNN [11]: 87.3
Part-level attention + prototypes	ProtoPNet [4]: 77.3
	Def. ProtoPNet[nd,3p,10pc]: 86.5
Part-level attention + prototypes + deformations	**Def. ProtoPNet[3p,10pc]: 86.5**

Table 3. Accuracy and interpretability of Deformable ProtoPNet compared to other models on CUB-200-2011 (top) and Stanford Dogs (bottom). Methods using bounding boxes are marked (bb) and methods using full, uncropped images are marked (f). For the Deformable ProtoPNets, we denote k prototypes per class as $k$pc, Deformable ProtoPNets with 2×2 protoypes as 2p, Deformable ProtoPNets with 3×3 protoypes as 3p, and Deformable ProtoPNets without deformations as nd. *Using 3 ensembled models. **Using 5 ensembled models.

interpretable baseline model in all cases. For all backbone architectures except VGG-16 and VGG-19 [37], a Deformable ProtoPNet with deformations and 2×2 deformable prototypes has the best performance across the models with the same backbone. We ran additional experiments on Stanford Dogs [18] and found that our Deformable ProtoPNet also performs well across multiple backbone architectures on that dataset. See the supplement for details.

We find that using deformations, orthogonality loss, and subtractive margin generally improves (or maintains) accuracy. As shown in Table 1, introducing deformations improves (or maintains) accuracy for most backbone architectures. We performed additional ablation studies on ResNet-50-based Deformable ProtoPNets with and without deformations – these models were trained using various settings of the subtractive margin and orthogonality loss. As shown in Table 2 (top), introducing deformations improves (or maintains) accuracy under the same settings of margin and orthogonality loss. As shown in Table 2 (bottom), introducing subtractive margin generally improves accuracy under the same settings of orthogonality loss and deformations. As shown in both Table 2 (top) and Table 2 (bottom), introducing orthogonality loss maintains accuracy in most cases.

We find that Deformable ProtoPNet can achieve state-of-the-art accuracy. As Table 3 (top) shows, a single Deformable ProtoPNet can achieve high accuracy (86.1% with 6 3×3 prototypes per class, 86.4% with 10 2×2 prototypes per class) on full test images from CUB-200-2011 [47], outperforming a single ProtoTree [28] (82.2%) and 3 ensembled TesNets [45] (83.5%). Additionally, 5 ensembled Deformable ProtoPNets using 2×2 prototypes outperform all competing models, achieving state-of-the-art accuracy (87.8%). Table 3 (bottom) shows that Deformable ProtoPNet also performs well on Stanford Dogs [18], achieving accuracy (86.5%) competitive with the state-of-the-art.

6. Conclusion

We presented Deformable ProtoPNet, a case-based interpretable neural network with deformable prototypes. The competitive performance and transparency of this model will enable wider use of interpretable models for computer vision. One limitation of Deformable ProtoPNet is that offsets are shared across all deformable prototypes at each spatial location. Another limitation is that we observed semantic mismatches between some prototypical parts and image parts that are considered "similar" by Deformable ProtoPNet. We plan to address these limitations in future work.

Acknowledgment. This work was supported in part by the GPU cluster, funded by NSF MRI # 1919478, at the Advanced Computing Group at the University of Maine.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PloS one*, 10(7):e0130140, 2015. 2

[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*, 3:1061–1070, Dec. 2021. 1

[3] Richard Berk, Drougas Berk, and Drougas. *Machine Learning Risk Assessments in Criminal Justice Settings*. Springer, 2019. 1

[4] Chaofan Chen, Oscar Li, Daniel Tao, Alina Jade Barnett, Cynthia Rudin, and Jonathan K Su. This Looks Like That: Deep Learning for Interpretable Image Recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1, 2, 3, 5, 6, 7, 8

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017. 1, 2, 4

[6] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. Ieee, 2005. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009. 7

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2019. 2, 5

[9] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing Higher-Layer Features of a Deep Network. Technical Report 1341, the University of Montreal, June 2009. Also presented at the Workshop on Learning Feature Hierarchies at the 26th International Conference on Machine Learning (ICML 2009), Montreal, Canada. 2

[10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009. 2

[11] 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. 8

[12] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable Part Models are Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 437–446, 2015. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. 7

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, Honolulu, HI, July 2017. IEEE. 7

[15] 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. 8

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial Transformer Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 1, 2, 4

[17] Yunho Jeon and Junmo Kim. Active Convolution: Learning the Shape of Convolution for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4201–4209, 2017. 1, 2, 4

[18] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel Dataset for Fine-Grained Image Categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 2, 8

[19] 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. 8

[20] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 2

[21] 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. 8

[22] 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. 8

[23] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 212–220, 2017. 2, 5

[24] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-Margin Softmax Loss for Convolutional Neural Networks. In Maria Florina Balcan and Kilian Q. Weinberger,

editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 507–516, New York, New York, USA, 20–22 Jun 2016. PMLR. 2, 5

[25] Xiao Liu, Tian Xia, Jiang Wang, and Yuanqing Lin. Fully Convolutional Attention Localization Networks: Efficient Attention Localization for Fine-Grained Recognition. *CoRR*, abs/1603.06765, 2016. 8

[26] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[27] Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. This Looks Like That, Because... Explaining Prototypes for Interpretable Image Recognition. In *3rd ECML PKDD International Workshop and Tutorial on eXplainable Knowledge Discovery in Data Mining (XKDD)*, 2021. 2

[28] 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 (CVPR)*, pages 14933–14943, 2021. 1, 2, 3, 8

[29] Anand Nayyar, Lata Gadhavi, and Noor Zaman. Machine learning in healthcare: review, opportunities and challenges. *Machine Learning and the Internet of Medical Things in Healthcare*, pages 23–45, 2021. 1

[30] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 3387–3395, 2016. 2

[31] Wanli Ouyang and Xiaogang Wang. Joint Deep Learning for Pedestrian Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2056–2063, 2013. 2

[32] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qiu, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy, et al. DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2015. 2

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”Why Should I Trust You?” Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016. 2

[34] 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, May 2019. 2

[35] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 06–11 Aug 2017. 2

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop at the 2nd International Conference on Learning Representations (ICLR Workshop)*, 2014. 2

[37] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. 7, 8

[38] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 2

[39] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv preprint arXiv:1412.6806*, 2014. 2

[40] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017. 2

[41] 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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2018. 7

[42] 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. 8

[43] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive Margin Softmax for Face Verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018. 2, 5

[44] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. CosFace: Large Margin Cosine Loss for Deep Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5265–5274, 2018. 2, 5

[45] 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 (ICCV)*, pages 895–904, October 2021. 1, 2, 5, 6, 8

[46] Thierry Warin and Aleksandar Stojkov. Machine Learning in Finance: A Metadata-Based Systematic Review of the Literature. *Journal of Risk and Financial Management*, 14(7):302, 2021. 1

[47] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 2, 7, 8

[48] Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding Neural Networks through Deep Visualization. In *Deep Learning Workshop at the 32nd International Conference on Machine Learning (ICML)*, 2015. 2

[49] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 818–833, 2014. 2

[50] 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. 8

[51] 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. 8

[52] 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. 8

[53] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable Convnets V2: More Deformable, Better Results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9308–9316, 2019. 1, 2, 4