# Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes (Supplement)

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

## 1. Proof of Theorem 3.1

**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 $\|\mathbf{z}_{\text{interp}}\|_2 = r$ for all $\alpha \in [0,1]$ and $\beta \in [0,1]$. (The square root is taken element-wise.)

*Proof.* To show that bilinear interpolation of vectors with the same $L^2$ norm does *not* in general preserve the $L^2$ norm, consider the following example with:

$$\hat{\mathbf{z}}_1 = \begin{bmatrix} r \\ 0 \\ 0 \\ 0 \end{bmatrix}, \hat{\mathbf{z}}_2 = \begin{bmatrix} 0 \\ r \\ 0 \\ 0 \end{bmatrix}, \hat{\mathbf{z}}_3 = \begin{bmatrix} 0 \\ 0 \\ r \\ 0 \end{bmatrix}, \hat{\mathbf{z}}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ r \end{bmatrix}.$$

For $\alpha = \beta = 1/2$, using bilinear interpolation, we have:

$$\begin{aligned}
\|\mathbf{z}_{\text{interp}}\|_2^2 &= \|(1-\alpha)(1-\beta)\hat{\mathbf{z}}_1 + (1-\alpha)\beta\hat{\mathbf{z}}_2 \\
&\quad + \alpha(1-\beta)\hat{\mathbf{z}}_3 + \alpha\beta\hat{\mathbf{z}}_4\|_2^2 \\
&= \left\| \frac{1}{4}\begin{bmatrix} r \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 0 \\ r \\ 0 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 0 \\ 0 \\ r \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 0 \\ 0 \\ 0 \\ r \end{bmatrix} \right\|_2^2 \\
&= \left\| \begin{bmatrix} \frac{1}{4}r \\ \frac{1}{4}r \\ \frac{1}{4}r \\ \frac{1}{4}r \end{bmatrix} \right\|_2^2 \\
&= \left(\frac{1}{4}r\right)^2 + \left(\frac{1}{4}r\right)^2 + \left(\frac{1}{4}r\right)^2 + \left(\frac{1}{4}r\right)^2 \\
&= \frac{1}{4}r^2,
\end{aligned}$$

which means

$$\|\mathbf{z}_{\text{interp}}\|_2 = \sqrt{\frac{1}{4}r^2} = \frac{1}{2}r \neq r.$$

This counter-example shows that bilinear interpolation generally does not preserve the $L^2$ norm of the interpolated vector.

Now, we show that the $L^2$ norm preserving interpolation operation $\mathbf{z}_{\text{interp}} = \sqrt{(1-\alpha)(1-\beta)\hat{\mathbf{z}}_1^2 + (1-\alpha)\beta\hat{\mathbf{z}}_2^2 + \alpha(1-\beta)\hat{\mathbf{z}}_3^2 + \alpha\beta\hat{\mathbf{z}}_4^2}$, when applied to vectors $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4 \in \mathbb{R}^n$ with the same $L^2$ norm $r$, yields an interpolated vector of the same $L^2$ norm $r$. Let

$$\hat{\mathbf{z}}_i = \begin{bmatrix} \hat{z}_{i,1} \\ \hat{z}_{i,2} \\ \vdots \\ \hat{z}_{i,n} \end{bmatrix} \text{ for all } i \in \{1,2,3,4\},$$

where $\hat{z}_{i,j}$ denotes the $j$-th component of the vector $\hat{\mathbf{z}}_i$. The element-wise square of the vector $\hat{\mathbf{z}}_i$ is given by:

$$\hat{\mathbf{z}}_i^2 = \begin{bmatrix} \hat{z}_{i,1}^2 \\ \hat{z}_{i,2}^2 \\ \vdots \\ \hat{z}_{i,n}^2 \end{bmatrix} \text{ for all } i \in \{1,2,3,4\}.$$

Using the $L^2$ norm preserving interpolation on $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4 \in \mathbb{R}^n$ with $\|\hat{\mathbf{z}}_i\| = r$ for all $i \in \{1,2,3,4\}$, *for all* $\alpha \in [0,1]$ and $\beta \in [0,1]$, and letting $\tilde{\alpha} = 1 - \alpha$ and

$\tilde{\beta} = 1 - \beta$, we have:

$$
\begin{aligned}
&\|\mathbf{z}_{\text{interp}}\|_2^2 \\
&= \left\| \sqrt{\tilde{\alpha}\tilde{\beta}\hat{\mathbf{z}}_1^2 + \tilde{\alpha}\beta\hat{\mathbf{z}}_2^2 + \alpha\tilde{\beta}\hat{\mathbf{z}}_3^2 + \alpha\beta\hat{\mathbf{z}}_4^2} \right\|_2^2 \\
&= \left\| \begin{bmatrix} \sqrt{\tilde{\alpha}\tilde{\beta}\hat{z}_{1,1}^2 + \tilde{\alpha}\beta\hat{z}_{2,1}^2 + \alpha\tilde{\beta}\hat{z}_{3,1}^2 + \alpha\beta\hat{z}_{4,1}^2} \\ \sqrt{\tilde{\alpha}\tilde{\beta}\hat{z}_{1,2}^2 + \tilde{\alpha}\beta\hat{z}_{2,2}^2 + \alpha\tilde{\beta}\hat{z}_{3,2}^2 + \alpha\beta\hat{z}_{4,2}^2} \\ \vdots \\ \sqrt{\tilde{\alpha}\tilde{\beta}\hat{z}_{1,n}^2 + \tilde{\alpha}\beta\hat{z}_{2,n}^2 + \alpha\tilde{\beta}\hat{z}_{3,n}^2 + \alpha\beta\hat{z}_{4,n}^2} \end{bmatrix} \right\|_2^2 \\
&= \quad \tilde{\alpha}\tilde{\beta}\hat{z}_{1,1}^2 + \tilde{\alpha}\beta\hat{z}_{2,1}^2 + \alpha\tilde{\beta}\hat{z}_{3,1}^2 + \alpha\beta\hat{z}_{4,1}^2 \\
&\quad + \tilde{\alpha}\tilde{\beta}\hat{z}_{1,2}^2 + \tilde{\alpha}\beta\hat{z}_{2,2}^2 + \alpha\tilde{\beta}\hat{z}_{3,2}^2 + \alpha\beta\hat{z}_{4,2}^2 \\
&\quad ... \\
&\quad + \tilde{\alpha}\tilde{\beta}\hat{z}_{1,n}^2 + \tilde{\alpha}\beta\hat{z}_{2,n}^2 + \alpha\tilde{\beta}\hat{z}_{3,n}^2 + \alpha\beta\hat{z}_{4,n}^2 \\
&= \tilde{\alpha}\tilde{\beta}\|\hat{\mathbf{z}}_1\|_2^2 + \tilde{\alpha}\beta\|\hat{\mathbf{z}}_2\|_2^2 + \alpha\tilde{\beta}\|\hat{\mathbf{z}}_3\|_2^2 + \alpha\beta\|\hat{\mathbf{z}}_4\|_2^2 \\
&= (1-\alpha)(1-\beta)r^2 + (1-\alpha)\beta r^2 + \alpha(1-\beta)r^2 + \alpha\beta r^2 \\
&= (1-\alpha)r^2 + \alpha r^2 \\
&= r^2,
\end{aligned}
$$

which means

$$\|\mathbf{z}_{\text{interp}}\|_2 = \sqrt{r^2} = r,$$

thus completing the proof. □

In a Deformable ProtoPNet, we apply the $L^2$ norm preserving interpolation operation to compute the interpolated image features $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ for given values of $a$, $b$, $m$, and $n$, as follows:

$$
\begin{aligned}
&\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2} \\
&= \sqrt{(1-\alpha)(1-\beta)\hat{\mathbf{z}}_1^2 + (1-\alpha)\beta\hat{\mathbf{z}}_2^2 + \alpha(1-\beta)\hat{\mathbf{z}}_3^2 + \alpha\beta\hat{\mathbf{z}}_4^2}
\end{aligned}
$$

with

$$\hat{\mathbf{z}}_1 = \hat{\mathbf{z}}_{\lfloor a+m+\Delta_1 \rfloor, \lfloor b+n+\Delta_2 \rfloor}, \tag{S1}$$

$$\hat{\mathbf{z}}_2 = \hat{\mathbf{z}}_{\lfloor a+m+\Delta_1 \rfloor, \lceil b+n+\Delta_2 \rceil}, \tag{S2}$$

$$\hat{\mathbf{z}}_3 = \hat{\mathbf{z}}_{\lceil a+m+\Delta_1 \rceil, \lfloor b+n+\Delta_2 \rfloor}, \tag{S3}$$

$$\hat{\mathbf{z}}_4 = \hat{\mathbf{z}}_{\lceil a+m+\Delta_1 \rceil, \lceil b+n+\Delta_2 \rceil}, \tag{S4}$$

and

$$\alpha = (a+m+\Delta_1) - \lfloor a+m+\Delta_1 \rfloor = \Delta_1 - \lfloor \Delta_1 \rfloor, \tag{S5}$$

$$\beta = (b+n+\Delta_2) - \lfloor b+n+\Delta_2 \rfloor = \Delta_2 - \lfloor \Delta_2 \rfloor. \tag{S6}$$

## 2. Backpropagation through a Deformable Prototype

Recall that a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$, when applied at the spatial position $(a, b)$ on the image-feature tensor $\hat{\mathbf{z}}$, computes its similarity with the interpolated image features $\hat{\mathbf{z}}_{a,b}^{\Delta}$ according to the following equation:

$$g(\hat{\mathbf{z}})_{a,b}^{(c,l)} = \sum_m \sum_n \hat{\mathbf{p}}_{m,n}^{(c,l)} \cdot \hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}, \tag{S7}$$

where we have

$$\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2} = \sqrt{\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)} \tag{S8}$$

and we have defined

$$
\begin{aligned}
&\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2) \\
&= (1-\alpha)(1-\beta)\hat{\mathbf{z}}_1^2 + (1-\alpha)\beta\hat{\mathbf{z}}_2^2 + \alpha(1-\beta)\hat{\mathbf{z}}_3^2 + \alpha\beta\hat{\mathbf{z}}_4^2,
\end{aligned}
$$

where $\hat{\mathbf{z}}_1$, $\hat{\mathbf{z}}_2$, $\hat{\mathbf{z}}_3$, $\hat{\mathbf{z}}_4$, $\alpha$, and $\beta$ are given by equations (S1), (S2), (S3), (S4), (S5), and (S6), respectively. Note that $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$ can be rewritten as:

$$
\begin{aligned}
&\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2) \\
&= \sum_i \sum_j \hat{\mathbf{z}}_{i,j}^2 \max(0, 1 - |(a+m+\Delta_1) - i|) \\
&\qquad \cdot \max(0, 1 - |(b+n+\Delta_2) - j|).
\end{aligned} \tag{S9}
$$

To show that we can back-propagate gradients through a deformable prototype, it is sufficient to show that we can compute the gradients of the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ (when a deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ is applied at the spatial position $(a, b)$ on the image-feature tensor $\hat{\mathbf{z}}$), with respect to every $(m, n)$-th prototypical part $\hat{\mathbf{p}}_{m,n}^{(c,l)}$ of the deformable prototype $\hat{\mathbf{p}}^{(c,l)}$ and with respect to every (discrete) spatial position $\hat{\mathbf{z}}_{i,j}$ of the image-feature tensor $\hat{\mathbf{z}}$.

From equation (S7), it is easy to see that the gradient of the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ with respect to the $(m, n)$-th prototypical part $\hat{\mathbf{p}}_{m,n}^{(c,l)}$ is given by:

$$\frac{\partial g(\hat{\mathbf{z}})_{a,b}^{(c,l)}}{\partial \hat{\mathbf{p}}_{m,n}^{(c,l)}} = \hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}^{\top}.$$

Before we derive the gradient of the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ with respect to a (discrete) spatial position $\hat{\mathbf{z}}_{i,j}$ of the image-feature tensor $\hat{\mathbf{z}}$, note that the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ is computed by first computing $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$ using equation (S9), followed by computing $\hat{\mathbf{z}}_{a+m+\Delta_1,b+n+\Delta_2}$ by taking the square root of $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$ element-wise (equation (S8), and finally computing the similarity score using equation S7. In particular, in the first step when we compute $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$ using equation (S9), note that

$\Delta_1 = \Delta_1(\hat{\mathbf{z}}, a, b, m, n)$ and $\Delta_2 = \Delta_2(\hat{\mathbf{z}}, a, b, m, n)$ are functions depending on $\hat{\mathbf{z}}$, $a$, $b$, $m$, and $n$ – in particular, for all (discrete) spatial positions $(a, b)$ on the image-feature tensor $\hat{\mathbf{z}}$ and for all $(m, n)$-th prototypical parts, $\Delta_1$ and $\Delta_2$ are produced by applying convolutional layers to $\hat{\mathbf{z}}$. Hence, there are three ways in which $\hat{\mathbf{z}}$ can influence $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$: (1) directly through the $\hat{\mathbf{z}}_{i,j}^2$ term in equation (S9), (2) through $\Delta_1$, and (3) through $\Delta_2$. We have to take into account all three ways in which $\hat{\mathbf{z}}$ influences $\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)$, when applying the chain rule to compute the gradient of the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ with respect to a (discrete) spatial position $\hat{\mathbf{z}}_{i,j}$ of the image-feature tensor $\hat{\mathbf{z}}$. In particular, we have:

$$\begin{aligned}
&\frac{\partial g(\hat{\mathbf{z}})_{a,b}^{(c,l)}}{\partial \hat{\mathbf{z}}_{i,j}} \\
&= \sum_m \sum_n \frac{\partial g(\hat{\mathbf{z}})_{a,b}^{(c,l)}}{\partial \hat{\mathbf{z}}_{a+m+\Delta_1, b+n+\Delta_2}} \frac{\partial \hat{\mathbf{z}}_{a+m+\Delta_1, b+n+\Delta_2}}{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)} \\
&\quad \left( \frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \hat{\mathbf{z}}_{i,j}} \right. \\
&\qquad + \frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \Delta_1} \frac{\partial \Delta_1}{\partial \hat{\mathbf{z}}_{i,j}} \\
&\qquad \left. + \frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \Delta_2} \frac{\partial \Delta_2}{\partial \hat{\mathbf{z}}_{i,j}} \right).
\end{aligned} \tag{S10}$$

From equation (S7), for given values of $a$, $b$, $m$, and $n$, we have:

$$\frac{\partial g(\hat{\mathbf{z}})_{a,b}^{(c,l)}}{\partial \hat{\mathbf{z}}_{a+m+\Delta_1, b+n+\Delta_2}} = (\hat{\mathbf{p}}_{m,n}^{(c,l)})^\top. \tag{S11}$$

From equation (S8), for given values of $a$, $b$, $m$, $n$, we have:

$$\frac{\partial \hat{\mathbf{z}}_{a+m+\Delta_1, b+n+\Delta_2}}{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)} = \operatorname{diag}\left( \frac{1}{2\sqrt{\zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}} \right), \tag{S12}$$

where diag is a function that converts a $d$-dimensional vector into a $d \times d$ diagonal matrix, and the square root is taken element-wise.

From equation (S9), for given values of $a$, $b$, $m$, $n$, we have:

$$\begin{aligned}
&\frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \hat{\mathbf{z}}_{i,j}} \\
&= \operatorname{diag}\big( 2\hat{\mathbf{z}}_{i,j} \max(0, 1 - |(a + m + \Delta_1) - i|) \\
&\qquad \cdot \max(0, 1 - |(b + n + \Delta_2) - j|) \big),
\end{aligned} \tag{S13}$$

$$\begin{aligned}
&\frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \Delta_1} \\
&= \sum_i \sum_j \hat{\mathbf{z}}_{i,j}^2 \max(0, 1 - |(b + n + \Delta_2) - j|) \\
&\qquad \cdot \begin{cases} 0 & \text{if } |(a + m + \Delta_1) - i| \geq 1 \\ 1 & \text{if } -1 < (a + m + \Delta_1) - i < 0 \\ -1 & \text{if } 0 \leq (a + m + \Delta_1) - i < 1, \end{cases}
\end{aligned} \tag{S14}$$

and

$$\begin{aligned}
&\frac{\partial \zeta^{(a,b,m,n)}(\hat{\mathbf{z}}, \Delta_1, \Delta_2)}{\partial \Delta_2} \\
&= \sum_i \sum_j \hat{\mathbf{z}}_{i,j}^2 \max(0, 1 - |(a + m + \Delta_1) - i|) \\
&\qquad \cdot \begin{cases} 0 & \text{if } |(b + n + \Delta_2) - j| \geq 1 \\ 1 & \text{if } -1 < (b + n + \Delta_2) - j < 0 \\ -1 & \text{if } 0 \leq (b + n + \Delta_2) - j < 1. \end{cases}
\end{aligned} \tag{S15}$$

With equations (S11) – (S15), and noting that the gradients $\frac{\partial \Delta_1}{\partial \hat{\mathbf{z}}_{i,j}}$ and $\frac{\partial \Delta_2}{\partial \hat{\mathbf{z}}_{i,j}}$ are well-defined (because $\Delta_1$ and $\Delta_2$ are produced by applying convolutional layers to $\hat{\mathbf{z}}$, and convolutional layers are differentiable), we have shown that the gradient of the prototype similarity score $g(\hat{\mathbf{z}})_{a,b}^{(c,l)}$ with respect to a (discrete) spatial position $\hat{\mathbf{z}}_{i,j}$ of the image-feature tensor $\hat{\mathbf{z}}$, given in equation (S10), is well-defined. Hence, we can back-propagate through a deformable prototype.

## 3. More Examples of Reasoning Processes

We present more examples of reasoning processes produced by the top performing models on each dataset (for CUB-200-2011 [10], we show ResNet50-based Deformable ProtoPNet [4] with $3 \times 3$ prototypes and with $2 \times 2$ prototypes; for Stanford Dogs [6], we show Resnet152-based Deformable ProtoPNet with $3 \times 3$ prototypes) in Figure 1, Figure 2 and Figure 3. In each case, we show the two deformable prototypes of the predicted class that produced the highest similarity scores. The similarity with these prototypes provide evidence for the test image belonging to the predicted class. For simplicity, we show only the evidence contributing to the predicted class for each test image.

Figure 1 shows the reasoning process of the best performing Deformable ProtoPNet on CUB-200-2011 [10] for a test image of an eastern towhee (top), a test image of an Acadian flycatcher (middle), and a test image of a pied billed grebe (bottom). To find evidence for the bird in Figure 1 (top) being an eastern towhee, our Deformable ProtoPNet compares each deformable prototype of the eastern

towhee class to the test image by scanning the prototype across the test image (in the latent space of image features) – in particular, the prototypical parts within a deformable prototype can adaptively change their relative spatial positions, as the deformable prototype moves across the test image (in the latent space), looking for image parts that are semantically similar to its prototypical parts. In the end, the Deformable ProtoPNet will take the highest similarity across the image for each deformable prototype as the similarity score between that prototype and the image – for the test image of an eastern towhee in Figure 1 (top), the similarity score between the top prototype of the eastern towhee class and the test image is $0.960$. Since cosine similarity is used by a Deformable ProtoPNet, all similarity scores fall between $-1$ and $1$, so a similarity score of $0.960$ is very high. For each comparison with a deformable prototype, colored boxes on the test image in the figure show the spatial arrangement of the prototypical parts that is used to produce the similarity score. The similarity score produced by each deformable prototype is then multiplied by a class connection value to produce the points contributed by that prototype, and the points contributed by all prototypes within a class are added to produce a class score. The class with the highest class score is the predicted class. For the test image of an eastern towhee in Figure 1 (top), the class score of the eastern towhee class is $6.945$, which is the highest among all classes, making eastern towhee the predicted class.

Similarly, Figure 1 (middle) shows the reasoning process of the best performing Deformable ProtoPNet with $3 \times 3$ prototypes on CUB-200-2011 [10] for a test image of an Acadian flycatcher, and Figure 1 (bottom) shows the reasoning process for a test image of a pied billed grebe.

Figure 2 shows the reasoning process of the best performing Deformable ProtoPNet with $2 \times 2$ prototypes on CUB-200-2011 [10] for a test image of a black-footed albatross (top), a test image of a rusty blackbird (middle) and for a test image of a bronzed cowbird (bottom).

Figure 3 shows the reasoning process of the best performing Deformable ProtoPNet on Stanford Dogs [6] for a test image of a Norfolk terrier (top), a test image of a toy terrier (middle) and for a test image of a bull mastiff (bottom).

## 4. Local Analysis: Visualizations of Most Similar Prototypes to Given Images

In this section, we visualize the most similar prototypes to a given test image (we call this a *local analysis* of the test image), for a number of test images. Figure 4 shows the two most similar deformable prototypes (learned by the best performing Deformable ProtoPNet using $3 \times 3$ prototypes) for each of three test images from CUB-200-2011 [10]. Figure 5 shows the two most similar deformable prototypes (learned by the best performing Deformable ProtoPNet using $2 \times 2$ prototypes) for each of three test images from CUB-200-2011 [10]. Figure 6 shows the two most similar deformable prototypes (learned by the best performing Deformable ProtoPNet) for each of three test images from Stanford Dogs [6]. For each test image on the left, the top row shows the two most similar deformable prototypes, and the bottom row shows the spatial arrangement of the prototypical parts on the test image that produced the similarity score for the corresponding prototype. In general, the most similar prototypes for a given image come from the same class as that of the image, and there is some semantic correspondence between a prototypical part and the image patch it is compared to under the spatial arrangement of the prototypical parts where the deformable prototype achieves the highest similarity across the image.

## 5. Global Analysis: Visualizations of Most Similar Images to Given Prototypes

In this section, we visualize the most similar training and test images to a given deformable prototype (we call this a *global analysis* of the prototype), for a number of deformable prototypes. Figure 7 shows the two most similar training images and the two most similar test images for each of three deformable prototypes learned by the best performing Deformable ProtoPNet with $3 \times 3$ prototypes on CUB-200-2011 [10]. Figure 8 shows the two most similar training images and the two most similar test images for each of three deformable prototypes learned by the best performing Deformable ProtoPNet with $3 \times 3$ prototypes on CUB-200-2011 [10]. Figure 9 shows the two most similar training images and the two most similar test images for each of three deformable prototypes learned by the best performing Deformable ProtoPNet on Stanford Dogs [6]. For each deformable prototype on the left, we show two most similar images from the training set (middle) and two most similar images from the test set (right). In general, the most similar training and test images for a given deformable prototype come from the same class as that of the prototype, and for each of the most similar images, there is some semantic correspondence between most prototypical parts and the image patches they are compared to under the spatial arrangement of the prototypical parts where the deformable prototype achieves the highest similarity across the image.

## 6. Numerical Results on Stanford Dogs

We conducted another case study of our Deformable ProtoPNet on Stanford Dogs [6]. We trained each Deformable ProtoPNet with 10 $3 \times 3$ deformable prototypes per class, where each prototype was composed of 9 prototypical parts. We ran experiments using VGG-19 [8], ResNet-152 [4], and DenseNet-161 [5] as CNN backbones. All backbones were pretrained using ImageNet [2].

| Method | VGG-19 | ResNet-152 | Densenet161 |
|---|---|---|---|
| Baseline | 77.3 | 85.2 | **84.1** |
| ProtoPNet [1] | 73.6 | 76.2 | 77.3 |
| Def. ProtoPNet (nd) | 74.8 | **86.5** | 83.7 |
| Def. ProtoPNet | **77.9** | **86.5** | 83.7 |

Table 1. Accuracy of Deformable ProtoPNet compared to the baseline model, ProtoPNet [1], and Deformable ProtoPNet without deformations (denoted (nd)) across different base architectures on the Stanford Dogs dataset [6].

| Interpretability | Model: accuracy |
|---|---|
| Part-level attention | FCAN [7]: 84.2<br>**RA-CNN** [3]: **87.3** |
| Part-level attn. + learned prototypes | ProtoPNet [1]: 77.3<br>**Def. ProtoPNet(nd)**: **86.5** |
| Part-level attn. + learned prototypes + deformations | **Def. ProtoPNet**: **86.5** |

Table 2. Accuracy and interpretability of Deformable ProtoPNet compared to other interpretable models on Stanford Dogs [6]. We use (nd) to indicate a Deformable ProtoPNet without using deformations.

**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 the baseline uninterpretable architecture in two out of three cases, including the highest performing model based on ResNet-152 [4]. In all cases, we achieve substantially higher accuracy than ProtoPNet [1].

**We find that using deformations improves or maintains accuracy**. As shown in Table 1, a Deformable ProtoPNet with deformations achieves a level of accuracy higher than or equal to the corresponding model without deformations across all three CNN backbones – in particular, a Deformable ProtoPNet with deformations achieves a test accuracy more than $3\%$ higher than the one without deformations when VGG-19 is used as a backbone.

**We find that a single Deformable ProtoPNet achieves accuracy on par with the state-of-the-art.** As Table 2 shows, a Deformable ProtoPNet can achieve accuracy ($86.5\%$) competitive with the state-of-the-art.

## 7. Experimental Setup

### 7.1. Hyperparameters

We ran experiments using VGG [8], ResNet [4], and DenseNet [5] as CNN backbones $f$. The ResNet-50 backbone was pretrained on iNaturalist [9], and all other backbones were pretrained using ImageNet [2]. We used $14 \times 14$ as the spatial dimension (height and width) of the latent image-feature tensor. In particular, since the CNN backbones produce feature maps of spatial dimension $7 \times 7$, we obtain $14 \times 14$ feature maps by removing the final max pooling from the backbone architecture, or by upsampling from $7 \times 7$ feature maps via bilinear interpolation. The convolutional feature maps are augmented with a uniform channel of value $\epsilon = 10^{-5}$, and then normalized and rescaled at each spatial position as described in the main paper.

When prototypes were allowed to deform, we used two convolutional layers to predict offsets for prototype deformations – the first convolutional layer has $128$ output channels, and the second convolutional layer has either $18$ output channels for $3 \times 3$ prototypes (to produce $2$ offsets for each of the $9$ prototypical parts at each spatial position) or $8$ output channels for $2 \times 2$ prototypes (to produce $2$ offsets for each of the $4$ prototypical parts at each spatial position).

For our experiments on CUB-200-2011 [10], we trained each Deformable ProtoPNet with 6 deformable prototypes per class when each prototype was composed of 9 prototypical parts, and 10 deformable prototypes per class when each prototype was composed of 4 prototypical parts (except where otherwise specified). For our experiments on Stanford Dogs [6], we trained each Deformable ProtoPNet with 10 deformable prototypes per class where each prototype was composed of 9 prototypical parts.

We trained each Deformable ProtoPNet for 30 epochs. In particular, we started our training with a "warm-up" stage, in which we loaded and froze the pre-trained weights and biases and we froze the offset prediction branch, and focused on training the deformable prototype layer for 5 epochs (7 epochs for VGG- and DenseNet-based Deformable ProtoPNets on CUB-200-2011 [10]), at a learning rate of $3 \times 10^{-3}$. We then performed a second "warm-up" stage, in which we froze the offset prediction branch and focused on training the deformable prototype layer as well as the weights and biases of the CNN backbone for 5 more epochs, at a learning rate of $1 \times 10^{-4}$ for the CNN backbone parameters and $3 \times 10^{-3}$ for the deformable prototypes. Finally, we jointly trained all model parameters for the remaining training epochs, at a starting learning rate of $1 \times 10^{-4}$ for the CNN backbone parameters, $3 \times 10^{-3}$ for the deformable prototypes, and $5 \times 10^{-4}$ for the offset prediction branch. We reduced the learning rate by a factor of 0.1 every 5 epochs. We performed prototype projection and last layer optimization at epoch 20 and epoch 30.

### 7.2. Hardware and Software

Experiments were conducted on two types of servers. The first type of servers comes with Intel Xeon E5-2620 v3 (2.4GHz) CPUs and two to four NVIDIA A100 SXM4 40GB GPUs, and the experiments were run on the first type of servers with PyTorch version 1.8.1 and CUDA version 11.1. The second type of servers comes with TensorEX TS2-673917-DPN Intel Xeon Gold 6226 Processor (2.7Ghz) CPUs with two NVIDIA Tesla 2080 RTX Ti GPUs, and the experiments were run on the second type of servers with PyTorch version 1.10.0 and CUDA version 10.2.

The deformable prototypes were implemented in CUDA C++, while other components of Deformable ProtoPNet were implemented in Python 3 using PyTorch. Code is available at https://github.com/jdonnelly36/Deformable-ProtoPNet.

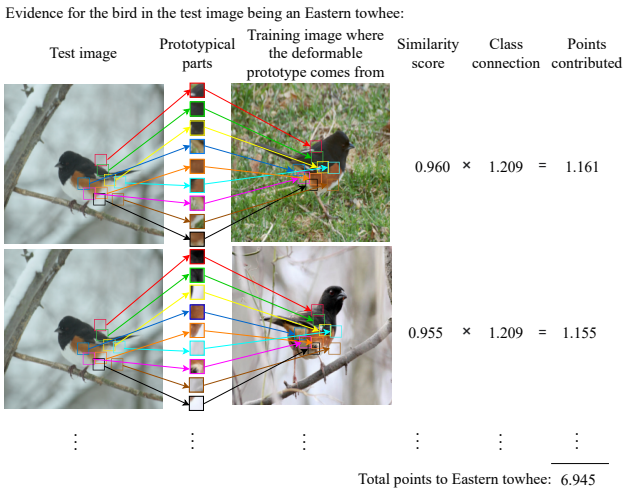


Figure 1. Example reasoning processes of a Deformable ProtoPNet with $3 \times 3$ prototypes when classifying a test image of an Eastern towhee (top), an Acadian flycatcher (middle), and a pied billed grebe (bottom). In each case, we show the two deformable prototypes of the predicted class that produced the highest similarity scores.

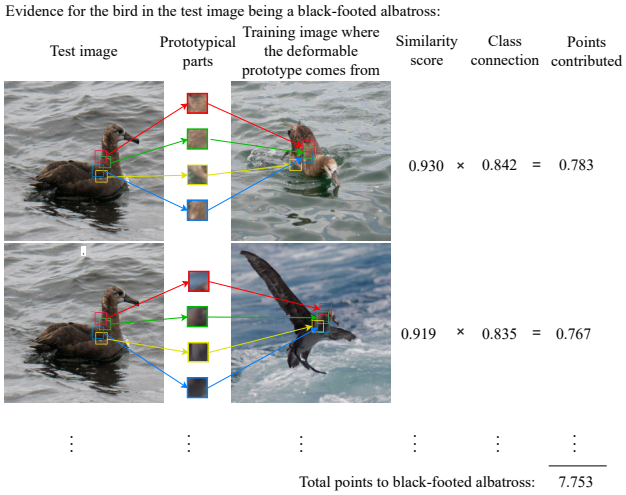


Figure 2. Example reasoning processes of a Deformable ProtoPNet with $2 \times 2$ prototypes when classifying a test image of a black-footed albatross (top), a rusty blackbird (middle), and a bronzed cowbird (bottom). In each case, we show the two deformable prototypes of the predicted class that produced the highest similarity scores.

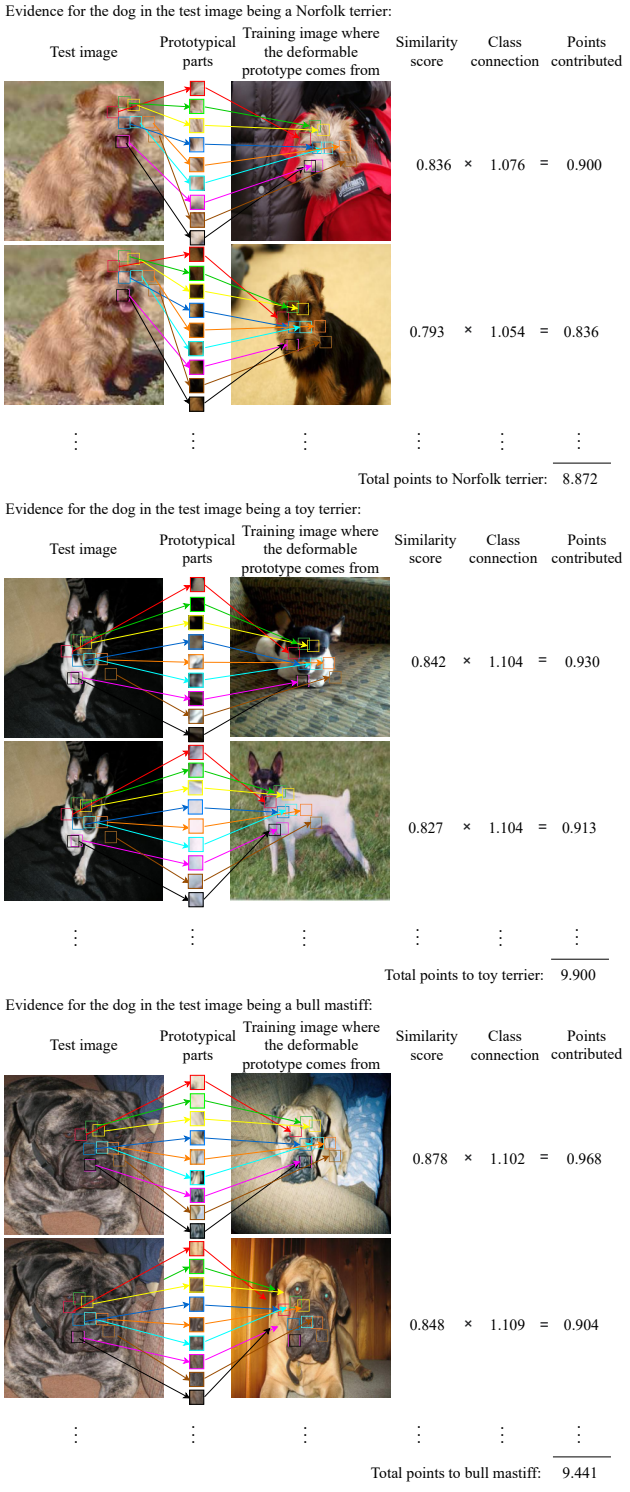


Figure 3. Example reasoning process of a Deformable ProtoPNet with $3 \times 3$ prototypes on a test image of a Norfolk terrier (top), a toy terrier (middle), and a bull mastiff (bottom). In each case, we show the two deformable prototypes of the predicted class that produced the highest similarity scores.

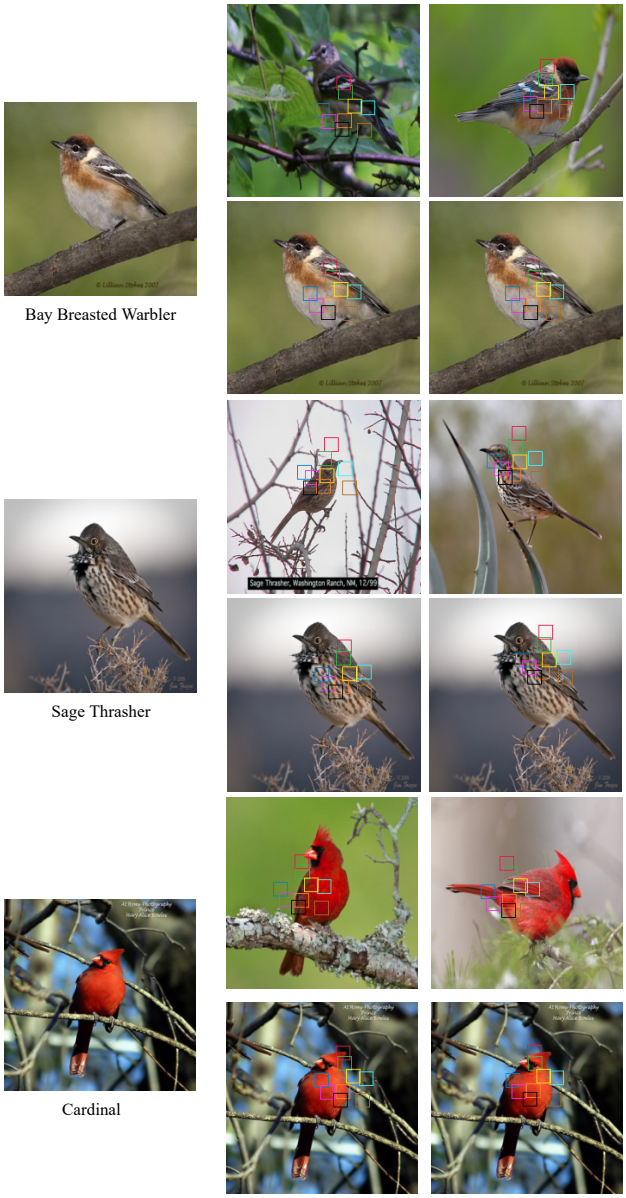


Figure 4. Local analyses of three test images from CUB-200-2011 [10] for a Deformable ProtoPNet with $3 \times 3$ prototypes. For each test image on the left, the top row shows the two most similar deformable prototypes, and the bottom row shows the spatial arrangement of the prototypical parts on the test image that produced the similarity score for the corresponding prototype.

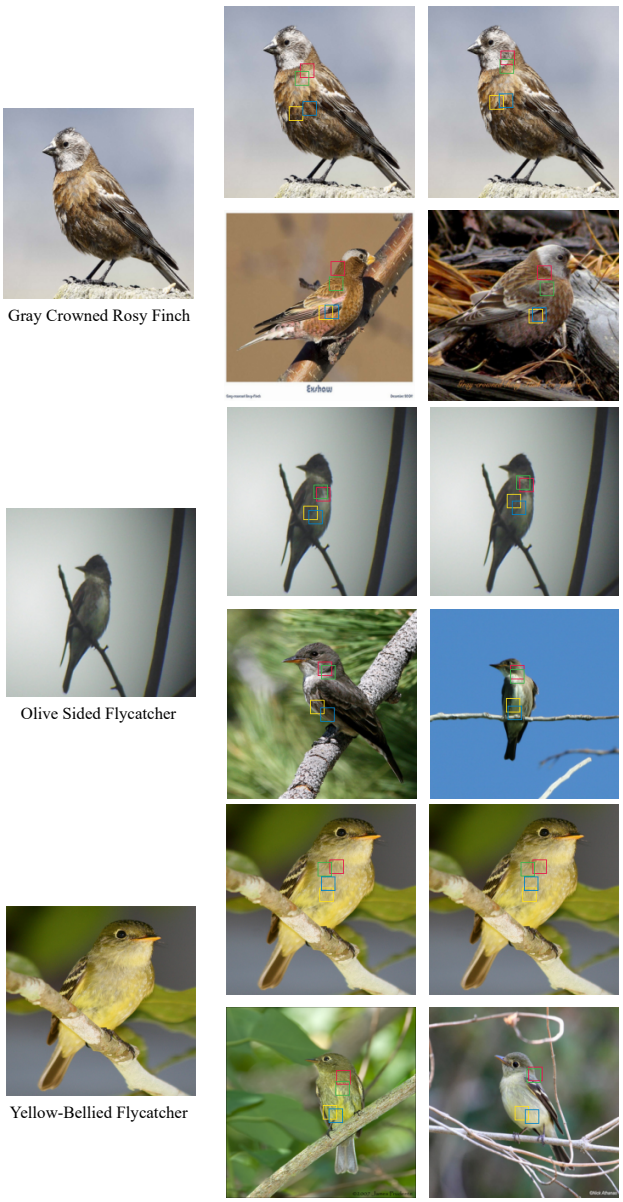


Figure 5. Local analyses of three test images from CUB-200-2011 [10] for a Deformable ProtoPNet with $2 \times 2$ prototypes. For each test image on the left, the top row shows the two most similar deformable prototypes, and the bottom row shows the spatial arrangement of the prototypical parts on the test image that produced the similarity score for the corresponding prototype.


Figure 6. Local analyses of three test images from Stanford Dogs [6] for a Deformable ProtoPNet with $3 \times 3$ prototypes. For each test image on the left, the top row shows the two most similar deformable prototypes, and the bottom row shows the spatial arrangement of the prototypical parts on the test image that produced the similarity score for the corresponding prototype.

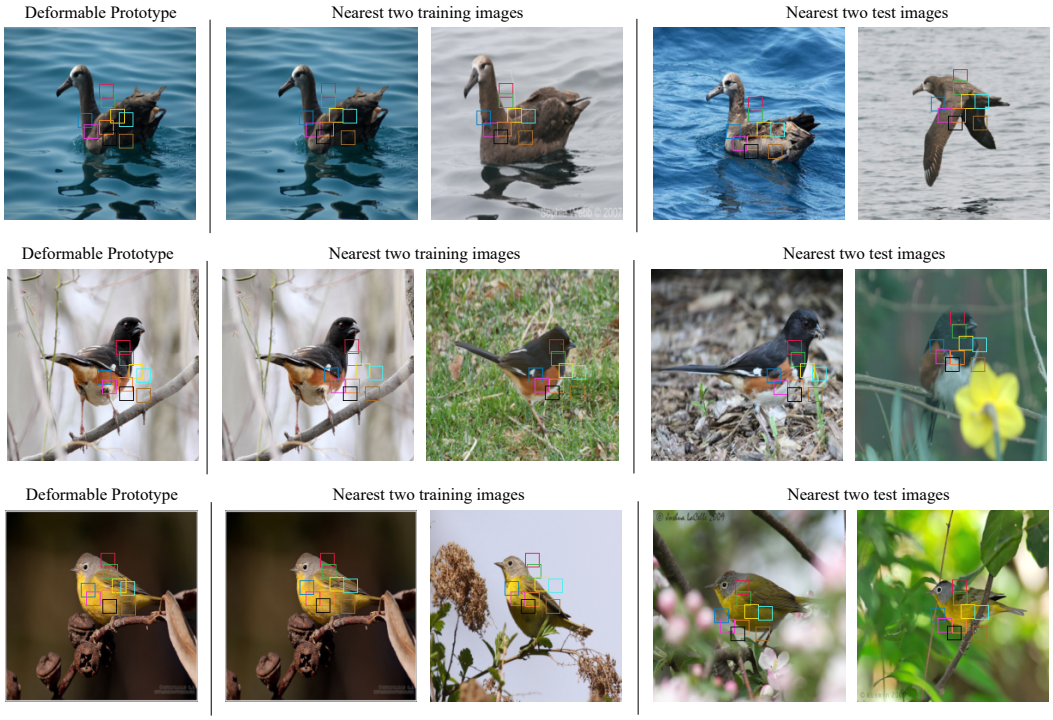


Figure 7. Global analyses of three deformable prototypes from CUB-200-2011 [10] for a Deformable ProtoPNet with $3 \times 3$ prototypes. For each deformable prototype on the left, we show two most similar images from the training set (middle) and two most similar images from the test set (right).

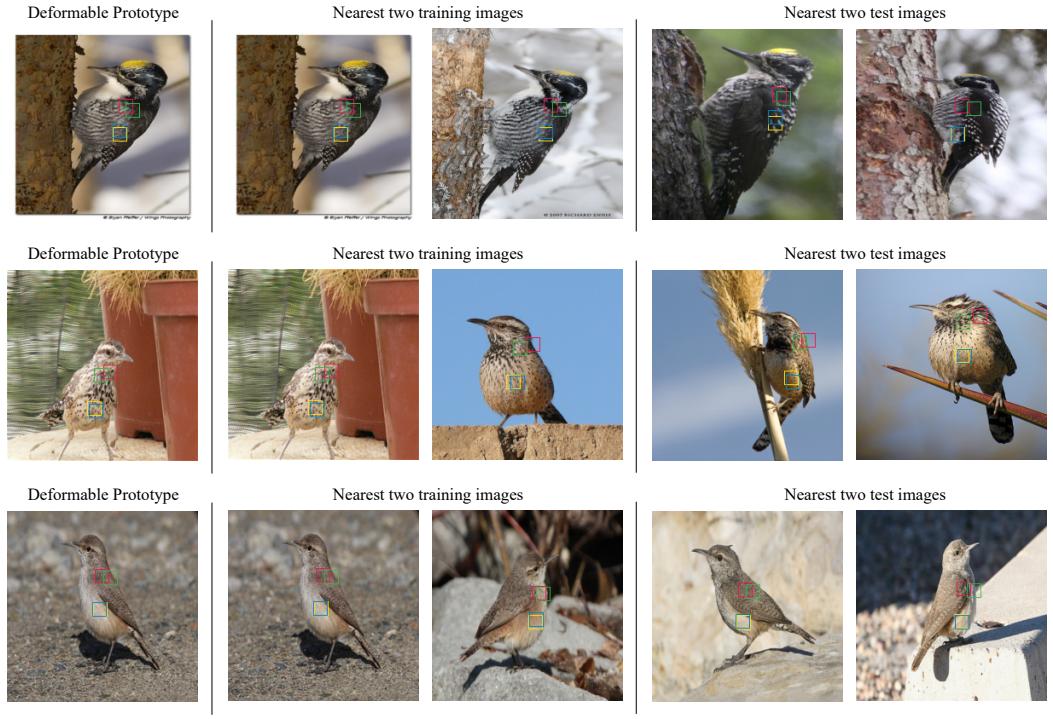


Figure 8. Global analyses of three deformable prototypes from CUB-200-2011 [10] for a Deformable ProtoPNet with $2 \times 2$ prototypes. For each deformable prototype on the left, we show two most similar images from the training set (middle) and two most similar images from the test set (right).

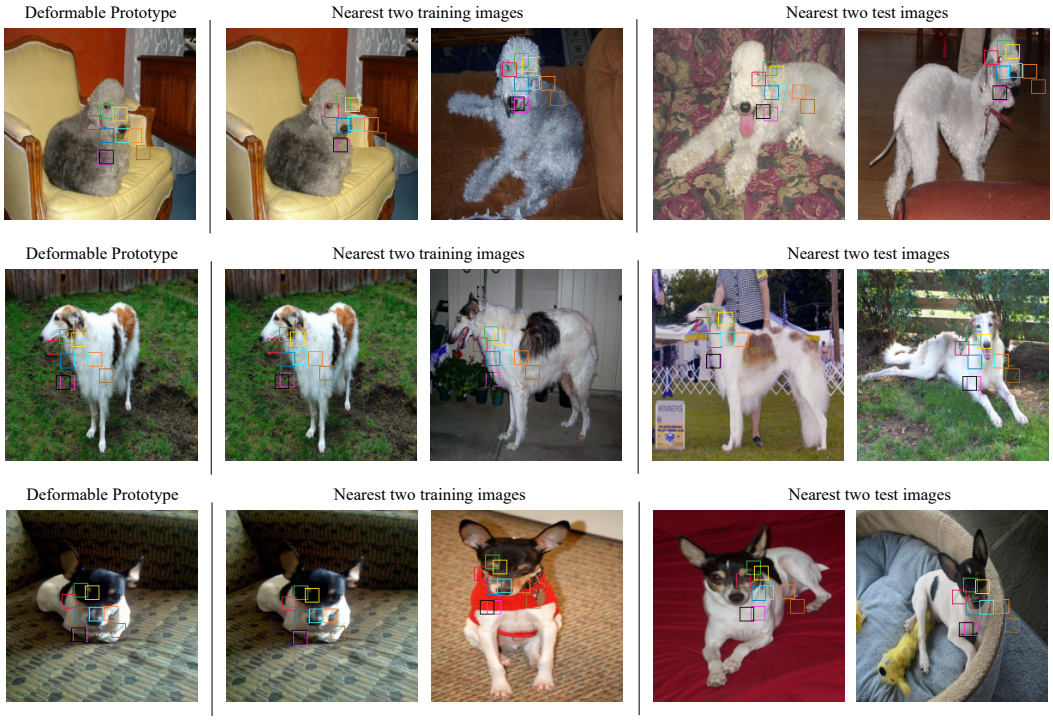


Figure 9. Global analysis for three deformable prototypes from Stanford Dogs [6] for a Deformable ProtoPNet with $3 \times 3$ prototypes. For each deformable prototype on the left, we show two most similar images from the training set (middle) and two most similar images from the test set (right).

## References


[1] 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. 5

[2] 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. 4, 5

[3] Jianlong Fu, Heliang Zheng, and Tao Mei. Look Closer to See Better: Recurrent Attention Convolutional Neural Network for Fine-Grained Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4438–4446, 2017. 5

[4] 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. 3, 4, 5

[5] 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. 4, 5

[6] 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. 3, 4, 5, 9, 11

[7] 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. 5

[8] 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. 4, 5

[9] 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. 5

[10] 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. 3, 4, 5, 8, 9, 10