Abstract
Prototypical part neural networks (ProtoPartNNs), namely PROTOPNET and its derivatives, are an intrinsically interpretable approach to machine learning. Their prototype learning scheme enables intuitive explanations of the form, this (prototype) looks like that (testing image patch). But, does this actually look like that? In this work, we delve into why object part localization and associated heat maps in past work are misleading. Rather than localizing to object parts, existing ProtoPartNNs localize to the entire image, contrary to generated explanatory visualizations. We argue that detraction from these underlying issues is due to the alluring nature of visualizations and an over-reliance on intuition. To alleviate these issues, we devise new receptive field-based architectural constraints for meaningful localization and a principled pixel space mapping for ProtoPartNNs. To improve interpretability, we propose additional architectural improvements, including a simplified classification head. We also make additional corrections to PROTOPNET and its derivatives, such as the use of a validation set, rather than a test set, to evaluate generalization during training. Our approach, PIXPNET (Pixel-grounded Prototypical Part Network), is the only ProtoPartNN that truly learns and localizes to prototypical object parts. We demonstrate that PIXPNET achieves quantifiably improved interpretability without sacrificing accuracy.

IEEE Winter Conference on Applications of Computer Vision (WACV) 2024
Pixel-Grounded Prototypical Part Networks

Zachariah Carmichael\textsuperscript{1,2} Suhas Lohit\textsuperscript{2} Anoop Cherian\textsuperscript{2} Michael J. Jones\textsuperscript{2} Walter J. Scheirer\textsuperscript{1}

\textsuperscript{1}University of Notre Dame
\textsuperscript{2}Mitsubishi Electric Research Laboratories

\texttt{zcarmich@nd.edu, \{slohit, anoop.cherian, mjones\}@merl.com, walter.scheirer@nd.edu}

\section*{Abstract}

Prototypical part neural networks (ProtoPartNNs), namely PROTO\textsuperscript{NET} and its derivatives, are an intrinsically interpretable approach to machine learning. Their prototype learning scheme enables intuitive explanations of the form, this (prototype) looks like that (testing image patch). But, does this actually look like that? In this work, we delve into why object part localization and associated heat maps in past work are misleading. Rather than localizing to object parts, existing ProtoPartNNs localize to the entire image, contrary to generated explanatory visualizations. We argue that detraction from these underlying issues is due to the alluring nature of visualizations and an over-reliance on intuition. To alleviate these issues, we devise new receptive field-based architectural constraints for meaningful localization and a principled pixel space mapping for ProtoPartNNs. To improve interpretability, we propose additional architectural improvements, including a simplified classification head. We also make additional corrections to PROTO\textsuperscript{NET} and its derivatives, such as the use of a validation set, rather than a test set, to evaluate generalization during training. Our approach, PIXPN\textsuperscript{ET} (Pixel-grounded Prototypical part Network), is the only ProtoPartNN that truly learns and localizes to prototypical object parts. We demonstrate that PIXPN\textsuperscript{ET} achieves quantifiably improved interpretability without sacrificing accuracy.

\section{1. Introduction}

Prototypical part neural networks (ProtoPartNNs) are an attempt to remedy the inscrutability and fundamental lack of trustworthiness characteristic of canonical deep neural networks [11]. By learning prototypes of object parts, ProtoPartNNs make human-interpretable predictions with justifications of the form: \textit{this} (training image patch) looks like \textit{that} (testing image patch). Since black-box AI systems often obfuscate their deficiencies [26, 44, 66], ProtoPartNNs represent a shift in the direction of transparency. With unprecedented interest in AI from decision-makers in high-stakes industries – e.g., medicine, finance, and law [44, 47, 54, 67] – the demand for explainable AI systems is greater than ever. Further motivation for transparency is driven by real-world consequences of deployed black boxes [6, 46, 52] and mounting regulatory ordinance [19, 20, 43, 69].

ProtoPartNNs approach explainability from an intrinsically interpretable lens and offer many benefits over post hoc explanation. Whereas post hoc explainers estimate an explanation, ProtoPartNN explanations are part of the actual prediction process – explanations along the lines of “\textit{this} looks like \textit{that}” follow naturally from the symbolic form of the model itself. This implicit explanation is characteristic of models widely considered to be human-comprehensible [53]. Moreover, ProtoPartNNs enable concept-level debugging, human-in-the-loop learning, and implicit localization [11, 48, 50]. Being independent of the explained model, post hoc explainers have been found to be unfaithful, inconsistent, and unreliable [8, 32, 41, 64] (see Section 2 for expanded discussion).

When misunderstood or used inappropriately, explain-
urable AI (XAI) methods can have unintended consequences [33, 41]. This harm arises from unverified hypotheses, whether it is that explanations represent phenomena faithful to the predictor or meaningful properties of the predictor. So, why do we see such hypotheses proliferating throughout both academia and industry [33, 42]? The problem is very human – there is often an over-reliance on intuition that may lead to illusory progress or deceptive conclusions. Whether it is dependence on alluring visualizations or behavioral extrapolation from cherry-picked examples, XAI methods are often left insufficiently scrutinized and subject to “researcher degrees of freedom” [42, 60].

Recent evidence indicates that ProtoPartNNs may suffer from these same issues: PROTOPNET and its variants exhibit irrelevant prototypes, a human-machine semantic similarity gap, and exorbitant explanation size [28, 37, 63]. Unfortunately, in our study, we confirm that this is the case – there are several facets of existing ProtoPartNN explanations that do not result from the implicit form of the model: image part localization, pixel space grounding, and heat map visualizations. Instead, these are founded on unverified assumptions and an over-reliance on intuition, often justified a posteriori by attractive visuals. We demonstrate that, colloquially, this does not actually look like that, and here may not actually correspond to there – see Figure 1 for illustration. These issues with ProtoPartNNs are not limited to just PROTOPNET, but to all of its derivatives.

This work aims to elevate the interpretability of ProtoPartNNs by rectifying these facets. In doing so, all aspects of ProtoPartNN explanations are embedded in the symbolic form of the model. Our contributions are as follows:

- We identify that existing ProtoPartNNs based on PROTOPNET do not localize faithfully nor actually localize to image parts, but rather the full image in most cases.

- We propose a novel pixel space mapping based on the receptive fields of an architecture (we guarantee that here corresponds to there).

- We propose architectural constraints that we efficiently discover through a transfer task to enable true image part localization (this looks like that).

- We devise a novel functional algorithm for the receptive field calculation of any architecture.

- On several image classification tasks, our approach, PIXPINET, achieves competitive accuracy with other ProtoPartNNs while maintaining a higher degree of interpretability, as substantiated by functionally grounded XAI metrics, and being the only ProtoPartNN that truly localizes to image parts.

2. Background

In this section, we give a brief background of explainable AI methods, the PROTOPNET formulation, and an overview of PROTOPNET extensions.

**Explainable AI Methods**

Explainable AI (XAI) solutions can be classified as post hoc, intrinsically interpretable, or a hybrid of the two [59]. Whereas intrinsically interpretable methods are both the explanator and predictor, post hoc methods act as an explanator for an independent predictor. Unfortunately, post hoc explainers are known to be inconsistent, unfaithful, and possibly even intractable [5, 8, 12, 22, 41]. Furthermore, they are deceivable [3, 14, 15, 64] and have been shown to not affect, or even reduce, end-user task performance [31, 32]. While this is the case, post hoc explanations have been shown to possibly increase user trust in AI systems [10], improve end-user performance for some explanation types and tasks [31], and explain black boxes in trustless auditing schemes [9]. However, for high-stakes domains, post hoc explanation is frequently argued to be especially inappropriate [54].

For these numerous reasons, our work concerns intrinsically interpretable machine learning solutions (see [59] for a methodological overview). In particular, we are interested in prototypical part neural networks (ProtoPartNNs) [11].

**PROTOPNET Architecture**

Here, we go over the PROTOPNET architecture [11], a type of ProtoPartNN. As much of the formalism overlaps with our approach, Figure 2 can be referred to for visualization of the architecture. Let \(D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}\) be the data set where each sample \(x_i \in \mathbb{R}^{3 \times H \times W}\) is an image with a height of \(H\) and a width of \(W\), and each label \(y_i \in \{1, \ldots, C\}\) represents one of \(C\) classes.

A PROTOPNET comprises a neural network backbone responsible for embedding an image. The first component of the backbone is the core \(f_{\text{core}}\), which could be a RESNET [25], VGG [61], or DENSENET [29] as in [11]. Proceeding, there are the add-on layers \(f_{\text{add}}\) that are responsible for changing the number of channels in the output of \(f_{\text{core}}\). In PROTOPNET, \(f_{\text{add}}\) comprises two \(1 \times 1\) convolutional layers with ReLU and sigmoid activation functions for the first and second layers, respectively. The full feature embedding function is denoted by \(f = f_{\text{add}} \circ f_{\text{core}}\). This function gives us our embedded patches \(f(x_i) = Z_i \in \mathbb{R}^{D \times H_z \times W_z}\) which have \(D\) channels, a height of \(H_z\), and a width of \(W_z\).

In PROTOPNET, we are interested in finding the most similar embedded patch \(z\) for each prototype. Each prototype can be understood as the embedding of some prototypical part of an object, such as the head of a blue jay as in Figure 2. Each embedded patch can be thought of in the same way – ultimately, a well-trained network will find that the most similar embedded patch and prototype will both be, e.g., the head of a blue jay (this prototype looks like that embedded patch). This is accomplished using the prototype layer, \(g\). We use the notation \(g_{p_i}\) to denote the unit that computes the most similar patch \(z \in \text{patches}(Z_i)\) to prototype \(p_i\). The function \(\text{patches}(Z_i)\) yields a set of \(D \times H_p \times W_p\) embedded patches in a sliding window manner (\(H_p = W_p = 1\) for all prototypes).
in PROTOPNet. First, the pairwise distances between patches \( \text{patches}(Z_i) \) and prototypes \( P = \{p_j\}_{j=1}^P \) are computed using a distance function \( \phi \) where \( p_j \in \mathbb{R}^{D \times H_p \times W_p} \), \( H_p \) is the prototype kernel height, \( W_p \) is the prototype kernel width, and \( P \) is the total number of prototypes. Each prototype is class-specific and we denote the set of prototypes belonging to class \( y_i \) as \( P_{y_i} \subseteq P \). Subsequently, a min-pooling operation is performed to obtain the closest embedded patch for each prototype – each prototype (this) is “assigned” a single embedded patch (that). Finally, the distances are converted into similarity scores using a similarity function \( v \). Putting this process altogether for unit \( g_p \), we have

\[
g_p(Z_i) = v\left( \min_{z \in \text{patches}(Z_i)} \phi(z, p_j) \right).
\]

We denote the vector of all similarity scores for a sample as \( s_i = g(Z_i) \in \mathbb{R}^P \).

The architecture ends with a readout layer \( h \) that produces the logits as \( \hat{y_i} = h(s_i) \). In PROTOPNet, \( h \) is a fully-connected layer with positive weights to same-class prototype units and negative weights to non-class prototype units. Each logit can be interpreted as the sum of similarity scores weighed by their importance to the class of the logit. Note that this readout layer is not reflected in Figure 2. The full PROTOPNet output for a sample is given by \( (h \circ g \circ f)(x_i) \).

**ProtoPartNN Desiderata and PROTOPNet Variants**

Many extensions of PROTOPNet have been proposed, some of which make alterations that fundamentally affect the interpretability of the architecture. To differentiate these extensions, we propose a set of desiderata for ProtoPartNNs:

1. **Prototypes must correspond directly to image patches.**
   This can be accomplished via prototype replacement, which grounds prototypes in human-interpretable pixel space (see Section 4 for details).
2. **Prototypes must localize to image parts.**
3. **Case-based reasoning must be describable by linear or simple tree models.**

Architectures that satisfy all three desiderata are considered to be 3-way ProtoPartNNs – satisfying fewer diminishes the interpretability of the algorithm.

The idea of sharing prototypes between classes has been explored in PROTOPSHARE [56] (prototype merge-pruning) and PROTOPPOOL [55] (differential prototype assignment). In PROTOTree [51], the classification head is replaced by a differentiable tree, also with shared prototypes. An alternative embedding space is explored in TESNet [72] based on Grassmann manifolds. A ProtoPartNN-specific knowledge distillation approach is proposed in PROTOP2PROTO [34] by enforcing that student prototypes and embeddings should be close to those of the teacher. DEFORMABLE PROTOPNet [17] extends the PROTOPNet architecture with deformable prototypes. ST-PROTO [71] learns support prototypes that lie near the classification boundary and trivial prototypes that are far from the classification boundary.

In an attempt to improve PROTOPNet visualizations, an extension of layer-wise relevance propagation [2]. Prototypical Relevance Propagation (PRP), is proposed to create more model-aware explanations [23]. PRP is quantitatively more effective in debugging erroneous prototypes and assigning pixel relevance than the original approach.

**ProtoPartNN-Like Methods**

The following papers are inspired by PROTOPNet but cannot be considered to be the same class of model. This is due to not fulfilling the proposed ProtoPartNN desidera #1 (prototypes must correspond directly to image patches) and/or #3 (case-based reasoning must be describable by linear or simple tree models).

ViT-Net [36] combines a vision transformer (ViT) with a neural tree decoder that learns prototypes. In another transformer-based approach, PROTOPFORKER [73] exploits the inherent architectural features (local and global branches) of ViTs. SEMI-PROTOPNet [65] fixes the readout weights as NP-PROTOPNet [62] does and is used for power distribution network analysis. In S DFA-SANET [30], a shallow-deep feature alignment (S DFA) module aligns the similarity structures between deep and
shallow layers. In addition, a score aggregation (SA) module aggregates similarity scores to avoid learning inter-class information. Unfortunately, each of these networks omits prototype replacement with the typical justification being that doing so improves task accuracy. In addition, ViT-NeT has additional layers after $g$ that break the mapping back to pixel space and complicate its case-based reasoning.

3. The Problem with Existing ProtoPartNNs

Despite the many extensions of ProtoPNet, there are still fundamental issues with image part localization, pixel space grounding, and heat map visualizations, which preclude any existing ProtoPartNN from satisfying all three desiderata – all ProtoPartNNs violate desideratum #2: prototypes must localize to image parts. The underlying issues with existing ProtoPartNNs arise from 1) their pixel space mapping being reliant on spatial correlation between embedded patches and the input space, which is dubious; 2) their pixel space mapping being receptive field-invariant, arbitrarily localizing to some area in the input. Rather, intrinsically interpretable models should produce explanations implicit in the symbolic form of the model itself [53, 59].

As a refresher, the original visualization process involves three steps. First, a single similarity map $S_{ij} = \pi_p (Z_i) \in \mathbb{R}^{H_z / H_p \times W_z / W_p}$ is selected for visualization where $\pi_p$ gives the similarity map for prototype $p_j$. Each element of $S_{ij}$ is given by $v (\varphi (z, p_j))$ where $z \in \text{patches}(Z_i)$. Subsequently, this map is upsampled from $H_z / H_p \times W_z / W_p$ to $H \times W$ using bicubic interpolation, producing a heat map $M_{ij} \in \mathbb{R}^{H \times W}$. To localize within the image, the smallest bounding box is drawn around the largest 5% of heat map elements – this box is of variable size. While no justification is provided for this approach in the original paper [11], we believe that the intuition is that the embedded patches $Z_i$ maintain spatial correlation with the input. Finally, $M_{ij}$ and the bounding box can be superimposed on the input image for visualization. From here on out, we will refer to this as the original pixel space mapping, which is visualized in Figure 3a. It should also be noted that while this pixel space mapping is crucial in establishing interpretability, it is left undiscovered in the vast majority of ProtoPNet extensions.

Immediately, we can see several issues with this approach.

Here Does Not Correspond to There The original pixel space mapping is based on naive upsampling, which is invariant to architectural details. The approach will always assume that all similarity scores can be mapped to pixel space with a single linear transformation – an embedded patch at position $\langle t_x, t_y \rangle$ is effectively localized to position $\langle t_x W_p / W_z, t_y H_p / H_z \rangle$ in pixel space. This assumption of spatial correlation from high to low layers is easy to invalidate. For instance, even a simple latent transpose eradicates this correlation. The similarity scores of embedded patches do not determine where the architecture “looked” in the image. Rather, the architecture determines where the similarity scores correspond to in the image. Figure 1 demonstrates this discrepancy. Very recently, evidence in [27, 57] strongly corroborates our arguments about poor localization. We correct this pixel space mapping according to the receptive fields of the underlying neural architecture. The original approach also only provides a way to localize a prototype rather than any embedded patch – our method enables us to do so. Our approach is described in detail in Section 4 and we validate its correctness over the original approach in Section 5.

This Does Not Correspond to Just That ProtoPNet and its derivatives all elect to localize to a small region of the input by drawing a bounding box around the largest 5% of values of heat map $M_{ij}$ as shown in Figure 3a. While this produces alluring visualizations, most of the architectures evaluated in all prior approaches have a mean receptive field of 100% at the embedding layer. A mean receptive field of 100% means that every element of the embedding layer output is a complex function of every pixel in the input space. Is it fair to say that only ~5% of the input contributed to some part of a decision? Attribution within the input space spanned by a receptive field is unverifiable from both the feature-selectivity and feature-additivity points of view [7, 42]. This issue is visualized in Figure 1 for an architecture with a mean receptive field under 100%. Moreover, while selecting the top 5% of $M_{ij}$ may localize in accordance with its (faulty) intuition, it can actually localize to wildly inaccurate parts of the image (e.g., if multiple top values in $S_{ij}$ are all close), breaking the intuition of the (unfaithful) pixel space mapping. We go on to discuss our solution to this problem in Section 4.

The Allure of Visualization The original pixel space mapping appears to satisfy human intuitions. However, it is not based on well-justified aspects of explainability. Beyond the assumption of spatial correlation and naive localization, bicubic interpolation artificially increases the resolution of maps (see Figure 3a), which leads non-experts to believe that per-pixel attributions are estimated. In our proposed approach, these explanation aspects follow naturally from the symbolic interpretation of the model itself.

4. Fixing ProtoPartNNs

As discussed in Section 3, the underlying issues with ProtoPartNNs arise from 1) the original pixel space mapping being reliant on spatial correlation between embedded patches and the input space, which is dubious; 2) the original pixel space mapping being receptive field-invariant,

\[ \text{The lowest mean receptive field of an evaluated architecture is from VGG19 (~70%) [11].} \]
Our proposed architecture, **PixPNet** (Pixel-grounded Prototypical part Network), is largely based on **ProtoPNet** but mitigates these issues through symbolic interpretation of its architecture – see Figure 2 for an overview. In this section, we first describe a new algorithm for the calculation of receptive fields, describe our proposed fixes for prototype visualization and localization, and proceed with additional ProtoPartNN corrections and improvements. With the proposed improvements, PixPNet is the only ProtoPartNN that truly localizes to image parts, satisfying all three desiderata.

Receptive Field Calculation Algorithm  Before delving into our proposed remedies, we describe our approach to computing receptive fields precisely for any architecture. Our proposed algorithm, **FunctionalRF**, takes a neural network as input and outputs the exact receptive field of every neuron in the neural network. Recall that a neuron is a function of a subset of pixels defined by its receptive field. **FunctionalRF** represents receptive fields as hypercubes (multidimensional tensor slices). For instance, the slices for a 2D convolution with a \(5 \times 5\) kernel, stride of 1, and \(c_m\) channels at output position 3, 3 would be \(\{[1, c_m], [1, 5], [1, 5]\} \) where \([a, b]\) denotes the slice between \(a\) and \(b\). We can compute the mean receptive field of a layer as the average number of pixels within the receptive field of each hypercube element of a layer output. The algorithm does not rely on approximate methods nor architectural alignment assumptions like other approaches \([1, 45]\). The full algorithmic details are provided in Appendix C.

Corrected Pixel Space Mapping Algorithm  From **Embedding Space to Pixel Space**  For each prototype \(\mathbf{p}_j\), we have some \(z \in Z_i\), that is most similar. We are interested in knowing where \(z\) localizes to in an image \(x_i\). With **FunctionalRF** applied to the backbone, we have the precise pixel space region that \(z\) is a function of – this exactly corresponds to that. This can also be done for any \(\mathbf{p}_j\) after prototype replacement. Additionally, this process can actually be used to visualize any \(z \in Z_i\), unlike the procedure specified in the original pixel space mapping \([11]\). See Figure 3b for intuition as to how this process works.

**Producing a Pixel Space Heat Map**  In order to compute a pixel space heat map, we propose an algorithm based on **FunctionalRF** rather than naively upsampling an embedding space similarity map \(S_{ij}\). Our approach uses the same idea as going from embedding space to pixel space. Each pixel space heat map \(M_{ij} \in \mathbb{R}^{H \times W}\) is initialized to all zeros \((0_{H \times W})\), and corresponds to a sample \(x_i\) and a prototype \(\mathbf{p}_j\). Let \(M^S_{ij}\) be the region of \(M_{ij}\) defined by the receptive field of similarity score \(S \in S_{ij}\). For each \(S\), the pixel space heat map is updated as \(M^S_{ij} \leftarrow \max(M^S_{ij}, S)\) where \(\max(\cdot)\) is an element-wise maximum that appropriately handles the case of overlapping receptive fields. We take maxima instead of averaging values due to Eq. (1). Again, see Figure 3b for a visualization of this procedure. Further algorithmic details are provided in Appendix D.

**Improved Localization & the “Goldilocks” Zone**  To reiterate, the region localized by a ProtoPartNN is controlled by the receptive field of the embedding layers of \(f\). A fundamental goal of ProtoPartNNs is to identify and learn prototypical object parts. We propose to achieve this by constraining the receptive field of \(f\) to a range that yields image parts that are both meaningful and interpretable to humans.

It is well known that the receptive field of a neural network correlates with performance \([1, 45]\) to an extent – too small or large a receptive field can harm performance due to bias-variance trade-offs \([39]\). We hypothesize that there is a “Goldilocks” zone where the desired receptive field localizes to intelligible image parts without diminishing task performance. To corroborate this, we evaluate various backbone architectures at intermediate layers on ImageNette \([21]\), a subset of ImageNet \([13]\). The evaluation aims to produce architectures suitable for the backbone of PixPNet according to the criteria outlined prior. We propose this approach as performance on subsets of ImageNet has been shown to be reflective of performance on the full dataset \([16]\), and ImageNet performance strongly correlates.
Figure 4. The Pareto front of architectures trained on ImageNet [21] and evaluated at various intermediate layers. This front details the accuracy-localization size trade-offs and informs backbone selection of PiXipNet as in Section 5.

with performance on other vision datasets [38]. We detail the full experiment setup in Appendix F. The Pareto front of mean receptive field and accuracy for the evaluated architectures is shown in Figure 4. This front informs our backbone selection as detailed in Section 5.

**Simplified Classification Head** While the original fully-connected classification head $h$ is human-interpretable, it has several weaknesses – its explanation size limits its comprehensibility [37, 63] and it requires an additional training stage, adding up to 100 additional epochs in PROTOPNet\(^2\). We quantify explanation size in terms of *positive reasoning* and *negative reasoning* about the prediction of a class. For positive reasoning, the number of elements in an explanation with the original fully-connected layer is $2P/C$: one similarity score per class-specific prototype and a positive weight coefficient. However, considering both positive and negative reasoning involves $2P$ total explanation elements.

To address these limitations, we propose to replace the linear layer with a class-wise summation. This operation simply produces the logit of each class as the sum of class-specific similarity scores as $y_{ic} = \sum_{j \in \mathcal{P}_c} s_{ij}$ where $y_{ic}$ is the logit for class $c$ and $s_{ij}$ is the similarity score for prototype $p_j$. The layer is visualized in Figure 2. Our new parameter-free readout layer removes the additional training stage and comprises only $P/C$ explanation elements for both positive and negative reasoning. Substituting our layer in the original PROTOPNet configuration for the CUB-200-2011 dataset [11, 70] reduces the number of explanation elements for a class prediction from 4,000 down to just 10.

**Other Improvements** We also make a few smaller contributions. In prototype replacement, we remove duplicate prototypes (by image or sample) to encourage diversity. If duplicates are found, the next most-similar embedded patch is used in replacement instead. We also reformu-

late the similarity function $v$ to have lower numerical error (see Appendix G for details) as $v(d) = \log(\frac{1}{d^2} + 1)$ where $\varepsilon$ is mitigates division by zero and the distance $d = \phi(z, p_j)$. While PROTOPNet uses $\phi(z, p_j) = \|z - p_j\|_2^2$ (cosine distance), which has a desirable normalizing factor. This distance is also used in [4, 17, 35, 72]. In implementation, the distances are computed using generalized convolution [11, 24, 49].

**Training** Our multi-stage training procedure is similar to that of PROTOPNet. The first stage optimizes the full network, except for the readout layer, by minimizing Eq. (2) via stochastic gradient descent

$$\frac{1}{N} \sum_{i=1}^{N} L_{xent}(\hat{y}_i, y_i) + \lambda_{\text{cls}} L_{\text{cls}}(P, Z_i) + \lambda_{\text{sep}} L_{\text{sep}}(P, Z_i)$$

(2)

where $L_{xent}$ is the categorical cross-entropy loss function, $\lambda_{\text{cls}}$ and $\lambda_{\text{sep}}$ are auxiliary loss weights, and the auxiliary loss functions, $L_{\text{cls}}$ and $L_{\text{sep}}$, are defined as

$$L_{\text{cls}}(P, Z_i) = \frac{1}{N} \sum_{i=1}^{N} \min_{p_j \in P_{y_i}} \phi(z, p_j)$$

(3)

$$L_{\text{sep}}(P, Z_i) = -\frac{1}{N} \sum_{i=1}^{N} \min_{p_j \in P_{y_i} \setminus P_{z_i}} \phi(z, p_j).$$

(4)

The goal of $L_{\text{cls}}$ is to ensure that at least one embedded patch of every training image is similar to at least one prototype belonging to the class of the image. In contrast, the goal of $L_{\text{sep}}$ is to ensure that the embedded patches of every training image are dissimilar from prototypes not belonging to the class of the image.

Subsequently, the prototypes are replaced, which is arguably the most important stage of training as it grounds prototypes in human-comprehensible pixel space. The process involves replacing each prototype $p_j$ with an embedded patch $z$ of a training sample of the same class – the most similar embedded patch replaces the prototype. In the literature, *prototype replacement* is also referred to as prototype “pushing” or “projection.” We stick with “replacement” for the sake of clarity. Formally, this update can be written as $p_j \leftarrow \arg \min_{z \in \mathcal{P}_{\text{patches}}(Z_i)} \phi(z, p_j)$, s.t. $p_j \in P_{y_i}$. Without this update, the human interpretation of prototypes is unclear as prototypes are not grounded in pixel space.

In PROTOPNet and its variants, a third stage optimizes the linear readout layer. However, we do not employ this stage as our readout layer is parameter-free. The multi-stage optimization process can be repeated until convergence.

5. Experiments & Discussion

To validate our proposed approach, PiXipNet, we evaluate both its accuracy and interpretability on CUB-200-2011 [70]. We also show evaluation results on Stanford
Cars [40] in Appendix B. We draw comparisons against other ProtoPartNNs with a variety of measures. We elect to not crop images in CUB-200-2011 by their bounding box annotations to demonstrate the localization capability of PiXPNET. Hyperparameters, software, hardware, and other reproducibility details are specified in Appendix E.

Lastly, upon inspection of the original code base\(^3\), we discovered that the test set accuracy is used to influence training of ProtoPNet. In fact, neither ProtoPNet nor its extensions for image classification that are mentioned in Section 2 employ a validation set in provided implementations. See Appendix H for further details. In our implementation, we employ a proper validation set and tune hyperparameters only according to accuracy on this split.

**Accuracy** The experimental results in Table 1 show that PiXPNET obtains competitive accuracy with other approaches regardless of whether images are cropped by bird bounding box annotations — while we trade off network depth for interpretability, we outperform ProtoPNet and several of its derivatives. This is quite favorable as PiXPNET is the only method that truly localizes to image parts.

**Interpretability** We evaluate the interpretability of our approach with several functionally grounded metrics [18]. See Figure 2b for an example of a PiXPNET explanation.

**Relevance Ordering Test (ROT)** The ROT is a quantitative measure of how well a pixel space mapping attributes individual pixels according to prototype similarity scores [23]. First, a pixel space heat map \( M_{ij} \) is produced for a single sample \( x_i \) and prototype \( p_j \). Starting from a completely random image, pixels are added back to the random image one at a time in descending order according to \( M_{ij} \). As each pixel is added back, the similarity score for \( p_j \) is evaluated. This procedure is averaged over each class-specific prototype over 50 random samples. The faster that the original similarity score is recovered, the better the pixel space mapping is. Assuming a faithful pixel space mapping, a network with a mean receptive field of, e.g., 25%, will recover the original similarity score after 25% of the pixels are added back in the worst-case scenario.

We also introduce two aggregate measures of the ROT. First is the area under the similarity curve (AUSC) which is normalized by the difference between the original similarity score and the baseline value (similarity score for a completely random image)\(^4\). Second is the percentage of pixels added back to recover the original similarity score: pixel percentage to recovery (%2R).

We compare our pixel space mapping to the original upsampling approach and PRP [23]. However, the PRP implementation only supports ResNet architectures\(^5\), so it is not included in all experiments. The results in Table 2 demonstrate that our pixel space mapping best identifies the most

\(^3\)https://github.com/cfchen-duke/ProtoPNet

\(^4\)AUSC>1 is possible as the maximum possible similarity is unknown.

\(^5\)The hard-coded and complex nature of the PRP code base precludes simple extension to other architectures.

<table>
<thead>
<tr>
<th>BBox</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>Model</th>
<th>Expl. Size +</th>
<th>Expl. Size ±</th>
<th>P</th>
<th>MRF</th>
<th>Acc. ±</th>
<th>( S_{con} )</th>
<th>( S_{sta} )</th>
<th>Code</th>
<th>Val. Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>PiXPNET (Ours)</td>
<td>ResNeXt@layer3</td>
<td>10</td>
<td>10</td>
<td>2000</td>
<td>100</td>
<td><strong>81.76</strong></td>
<td>0.2</td>
<td>56.4</td>
<td>64.7</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>PiXPNET (Ours)</td>
<td>VGG19@maxpool5</td>
<td>10</td>
<td>10</td>
<td>2000</td>
<td>70.4</td>
<td>80.10</td>
<td>0.1</td>
<td>47.6</td>
<td>64.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>PiXPNET (Ours)</td>
<td>VGG16@maxpool5</td>
<td>10</td>
<td>10</td>
<td>2000</td>
<td>52.5</td>
<td>79.75</td>
<td>0.2</td>
<td>69.5</td>
<td>51.6</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>PiXPNET (Ours)</td>
<td>VGG13@maxpool4</td>
<td>10</td>
<td>10</td>
<td>2000</td>
<td>9.69</td>
<td>75.32</td>
<td>0.2</td>
<td>66.9</td>
<td>45.0</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ST-ProtoPNet [71]</td>
<td>DenseNet161</td>
<td>20</td>
<td>4000</td>
<td>2000</td>
<td>100</td>
<td>80.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ST-ProtoPNet [71]</td>
<td>DenseNet161</td>
<td>20</td>
<td>4000</td>
<td>2000</td>
<td>100</td>
<td>86.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ProtoPNet [11]</td>
<td>DenseNet121</td>
<td>20</td>
<td>4000</td>
<td>2000</td>
<td>100</td>
<td>84.80</td>
<td>0.2</td>
<td>63.1</td>
<td>66.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ProtoPool [55]</td>
<td>ResNet152</td>
<td>20</td>
<td>404</td>
<td>202</td>
<td>100</td>
<td>81.50</td>
<td>0.1</td>
<td>35.7</td>
<td>58.4</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Proto2Proto [34]</td>
<td>ResNet34</td>
<td>20</td>
<td>4000</td>
<td>2000</td>
<td>100</td>
<td>79.89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ProtoShare [56]</td>
<td>DenseNet161</td>
<td>1200</td>
<td>1200</td>
<td>600</td>
<td>100</td>
<td>76.45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ProtoTree [30, 51]</td>
<td>DenseNet121</td>
<td>18</td>
<td>404</td>
<td>202</td>
<td>100</td>
<td>73.20</td>
<td>21.5</td>
<td>24.4</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ProtoPFormer [73]</td>
<td>DeiT-S</td>
<td>40</td>
<td>8000</td>
<td>4000</td>
<td>100</td>
<td><strong>84.85</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ViT-Net [36, 73]</td>
<td>CaIT-XXS-24</td>
<td>8</td>
<td>30</td>
<td>15</td>
<td>100</td>
<td>84.51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ViT-Net [36]</td>
<td>SwinT-B</td>
<td>10</td>
<td>62</td>
<td>31</td>
<td>100</td>
<td>91.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>S DFA- SA [30]</td>
<td>DenseNet161</td>
<td>20</td>
<td>20</td>
<td>2000</td>
<td>100</td>
<td>86.80</td>
<td></td>
<td>73.2</td>
<td>73.5</td>
</tr>
</tbody>
</table>

Table 1. ProtoPartNN results on CUB-200-2011 with ImageNet used for pre-training. Columns D1, D2, and D3 correspond to the three desiderata established in Section 2. Our approach, PiXPNET, is the only method that is a 3-way ProtoPartNN, satisfying all three desiderata. “BBox” indicates whether a method crops each image using a bounding box annotation. The best results of ProtoPartNNs with and without such annotations are bold and underlined, respectively. The table is split based on whether the method meets at least two desiderata. The \( S_{con} \) and \( S_{sta} \) scores for other methods are taken from [30] and the top reported accuracy score is taken for each method.
important pixels in an image. Naturally, the mean receptive field correlates with both ROT scores.

**Explanation Size**  Recall from Section 4 that the explanation size is the number of elements in an explanation, i.e., similarity scores and weight coefficients. This number differs when considering positive or negative reasoning. Due to the original classification head being fully-connected, most ProtoPartNNs have large explanation sizes when considering both positive and negative reasoning, as shown in Table 1. In contrast, our explanation size comprises just 10 elements when reasoning about a decision. Our proposed classification head helps to prevent overwhelming users with information, which has been shown to be the case with other ProtoPartNNs [37].

**Consistency**  The consistency metric [30] quantifies how consistently each prototype localizes to the same human-annotated ground truth part. It evaluates both semantic similarity quality and the pixel space mapping to a degree. For a sample $x_i$ with label $y_i$, the pixel space mapping is computed for each prototype $p_j \in P_{y_i}$. Let $o_{p_j}(x_i) \in \mathbb{R}^K$ be a binary vector indicating which of $K$ object parts are actually visible in $x_i$. A single object part is associated with $p_j$ by taking the maximum frequency of an object part present in the pixel space mapping region across all applicable images. A prototype is said to be consistent if this frequency is at least $\mu$, i.e.,

$$S_{\text{con}} = \frac{1}{P} \sum_{j=1}^{P} \mathbb{1} \left[ \max \left( \sum_{x_i \in X_j} \left( o_{p_j}(x_i) \odot u(x_i) \right) \right) \geq \mu \right]$$

where $X_j$ are samples of the same class allocated to $p_j$, $\odot$ denotes element-wise division, and $\mathbb{1}$ is the indicator function. To compare with results reported in [30], we change the receptive field size in our pixel space mapping to equal this, as well as set $\mu = 0.8$. A notable weakness of the evaluation approach is that it uses a fixed $72 \times 72$ pixel region independent of the architecture. While the approach is not perfect, it allows for reproducible and comparative interpretability evaluation between ProtoPartNN variants.

Results are shown in Tables 1 and 2 for CUB-200-2011, which provides human-annotated object part annotations. We outperform PROTOFNN and many of its variants, as well as the original pixel space mapping (Table 2).

**Stability**  The stability metric [30] measures how robust object part association is when noise is added to an image. Simply, some noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is added to each sample $x_i$ and the object part associations are compared as

$$S_{\text{sta}} = \frac{1}{P} \sum_{j=1}^{P} \frac{\sum_{x_i \in X_j} \mathbb{1} \left[ o_{p_j}(x_i) = o_{p_j}(x_i + \epsilon) \right]}{|X_j|}.$$

Following [30], we set $\sigma=0.2$. Results in Tables 1 and 2 support the robustness of PIXNET compared to other ProtoPartNNs and the original pixel space mapping. There is a marginal decrease in stability as the receptive field lessens.

### Table 2. Evaluation of pixel space mapping (PSM) methods with functionally-grounded interpretability metrics. Methods are compared on PIXNET with “Goldilocks” zone and RESNET backbones on CUB-200-2011 (no BBox cropping). Our PSM outperforms both the original and PRP PSMs across all backbones.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>MRF</th>
<th>Acc.</th>
<th>PSM</th>
<th>$S_{\text{con}}$</th>
<th>$S_{\text{sta}}$</th>
<th>AUSC</th>
<th>%2R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>↓</td>
</tr>
<tr>
<td>VGG11 @maxpool4</td>
<td>8.31</td>
<td>72.9</td>
<td>Ours</td>
<td>65.3</td>
<td>48.3</td>
<td>0.99</td>
<td>11.2</td>
</tr>
<tr>
<td>VGG13 @maxpool4</td>
<td>9.69</td>
<td>75.3</td>
<td>Ours</td>
<td>66.9</td>
<td>45.0</td>
<td>0.97</td>
<td>13.0</td>
</tr>
<tr>
<td>VGG16 @maxpool4</td>
<td>15.7</td>
<td>76.4</td>
<td>Ours</td>
<td>62.0</td>
<td>46.4</td>
<td>1.02</td>
<td>6.98</td>
</tr>
<tr>
<td>VGG19 @maxpool4</td>
<td>22.8</td>
<td>77.1</td>
<td>Ours</td>
<td>60.1</td>
<td>42.5</td>
<td>0.94</td>
<td>21.4</td>
</tr>
<tr>
<td>VGG13 @maxpool5</td>
<td>33.5</td>
<td>78.1</td>
<td>Ours</td>
<td>67.0</td>
<td>42.5</td>
<td>0.90</td>
<td>29.5</td>
</tr>
<tr>
<td>VGG16 @maxpool5</td>
<td>52.5</td>
<td>79.8</td>
<td>Ours</td>
<td>69.5</td>
<td>51.6</td>
<td>0.90</td>
<td>32.0</td>
</tr>
<tr>
<td>WRN50 @layer3</td>
<td>69.9</td>
<td>80.1</td>
<td>Ours</td>
<td>56.4</td>
<td>64.7</td>
<td>0.93</td>
<td>13.0</td>
</tr>
<tr>
<td>VGG19 @maxpool5</td>
<td>70.4</td>
<td>80.1</td>
<td>Ours</td>
<td>47.6</td>
<td>64.2</td>
<td>0.92</td>
<td>43.4</td>
</tr>
<tr>
<td>ResNet18 @layer2</td>
<td>15.4</td>
<td>57.2</td>
<td>PRP</td>
<td>–</td>
<td>–</td>
<td>0.95</td>
<td>25.4</td>
</tr>
<tr>
<td>ResNet50 @layer3</td>
<td>69.8</td>
<td>76.6</td>
<td>PRP</td>
<td>–</td>
<td>–</td>
<td>0.34</td>
<td>100.0</td>
</tr>
</tbody>
</table>

The receptive field constraint is a design choice and is inherently application-specific, subject to data characteristics and interpretability requirements. Future work should investigate multi-scale receptive fields and automated receptive field design techniques. Nevertheless, we trade off network depth for significant gains in interpretability with very little penalty in accuracy. Prior studies have shown that ProtoPartNNs have a semantic similarity gap with humans, prototypes can be redundant or indistinct, and limited utility in improving human performance [27,28,37,63]. Moreover, the consistency and stability evaluation metrics are imperfect. Although we improve upon interpretability over other networks, human studies are needed to understand other facets of interpretability, such as trustworthiness, acceptance, and utility [59]. In the future, architectural improvements should be made, e.g., the enriched embedding space of TeSNET, prototype diversity constraints [58,68,71], and human-in-the-loop training [48].

### 6. Limitations and Future Work

The receptive field constraint is a design choice and is inherently application-specific, subject to data characteristics and interpretability requirements. Future work should investigate multi-scale receptive fields and automated receptive field design techniques. Nevertheless, we trade off network depth for significant gains in interpretability with very little penalty in accuracy. Prior studies have shown that ProtoPartNNs have a semantic similarity gap with humans, prototypes can be redundant or indistinct, and limited utility in improving human performance [27,28,37,63]. Moreover, the consistency and stability evaluation metrics are imperfect. Although we improve upon interpretability over other networks, human studies are needed to understand other facets of interpretability, such as trustworthiness, acceptability, and utility [59]. In the future, architectural improvements should be made, e.g., the enriched embedding space of TeSNET, prototype diversity constraints [58,68,71], and human-in-the-loop training [48].
References


[10] Selina Carter and Jonathan Hersh. Explainable ai helps bridge the ai skills gap: Evidence from a large bank. Economics Faculty Articles and Research, 276, 2022. 2


Adrian Hoffmann, Claudio Fanconi, Rahul Rade, and Jonas Kohler. This looks like that... does it? Shortcomings of latent space prototype interpretability in deep networks. In ICML Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI, 2022.


Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In 4th International IEEE Workshop on 3D Representation and Recognition (3DRR-13), Sydney, Australia, 2013.


Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In Ankur Teredesai, Vinip Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, SIGKDD Intern-
national Conference on Knowledge Discovery & Data Mining, KDD, pages 903–913. ACM, 2019. 1, 8


[57] Mikolaj Sacha, Bartosz Jura, Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zielinski. Interpretability benchmark for evaluating spatial misalignment of prototypical parts explanations. arXiv, 2023. 4


[59] Gesina Schwalbe and Bettina Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Mining and Knowledge Discovery, Jan 2023. 2, 4, 8


[73] Mengqi Xue, Qihan Huang, Haofei Zhang, Lechao Cheng, Jie Song, Minghui Wu, and Mingli Song. ProtoPFormer:
Concentrating on prototypical parts in vision transformers for interpretable image recognition. *arXiv*, 2022. 3, 7