Revisiting Pose-Normalization for Fine-Grained Few-Shot Recognition

Luming Tang Davis Wertheimer Bharath Hariharan
Cornell University
{lt453,dww78,bh497}@cornell.edu

Abstract

Few-shot, fine-grained classification requires a model to learn subtle, fine-grained distinctions between different classes (e.g., birds) based on a few images alone. This requires a remarkable degree of invariance to pose, articulation and background. A solution is to use pose-normalized representations: first localize semantic parts in each image, and then describe images by characterizing the appearance of each part. While such representations are out of favor for fully supervised classification, we show that they are extremely effective for few-shot fine-grained classification. With a minimal increase in model capacity, pose normalization improves accuracy between 10 and 20 percentage points for shallow and deep architectures, generalizes better to new domains, and is effective for multiple few-shot algorithms and network backbones. Code is available at `https://github.com/Tsingularity/PoseNorm_Fewshot`.

1. Introduction

The ability to generalize with minimal fine-tuning is a crucial property for learned neural models, not just to unseen data but also to unseen *types* of data. Consider the task shown in Figure 1. We are given just a single image (or a very small number) from a few bird species, and from this information alone we must learn to recognize them. Humans are known to be very good at this *few-shot learning* task [19], but machines struggle: in spite of dramatic progress in visual recognition and two years of focused research, performance on several few-shot benchmarks remains far below that of fully supervised approaches.

This is a problem in practice, especially for fine-grained classification problems (such as that in Figure 1). In this setting, distinct classes can number in the hundreds, while the expertise and effort required to correctly label these classes can make annotation expensive. Together, this makes the collection of large labeled training sets for fine-grained classification difficult, sometimes prohibitively so. The ability of neural networks to handle fine-grained, few-shot learning can thus be crucial for real-world applications.

What is the reason behind the large gap between machine and human performance on this task? An intuitive hypothesis is that humans use a much more stable feature representation, which is invariant to large spatial deformations. For example, in the bird classification task, we might characterize a bird image using the attributes of its various parts: the shape of the beak, the color of the wing, the presence or absence of a crown. Such a characterization is invariant not just to changes in the image background, but also to variation in camera pose and articulation, allowing us to effectively perceive similarities and differences across a wide range of bird species, and individual images of them.

Such a featurization is *"pose normalized"*, and was explored as a promising direction for fine-grained classification before the re-discovery of convolutional networks [32]. Researchers found, however, that end-to-end training with black-box architectures, and without pose normalization, led to great improvement in the standard benchmarks (albeit with consistent modifications, such as bilinear pooling [16]). Indeed, in recent years, winners on the annual fine-grained classification challenges [1] have mostly focused on these black-box architectures. The intuitive idea of pose normalization has fallen by the wayside.

In contrast, we argue that the dominance of black-box architectures over pose-normalized representations is an artifact of the fully-supervised classification problem. In these settings, all classes to be distinguished are known *a priori*, and we have significant amounts of training data for each class. This reduces the need for pose and background invariance, since the training data will likely include a broad range of variation within each class. At the same time, leveraging category-specific biases in pose and background will likely be beneficial, since the representation need not generalize to new classes. These factors act in favor of black-box architectures with no built-in inductive biases. However, if we want the learnt model to *adapt to new classes from limited data*, as in few-shot learning, the intuitive invariance of pose normalization becomes more useful.

In this paper, we revisit pose normalization for the task of few-shot, fine-grained classification, and demonstrate its

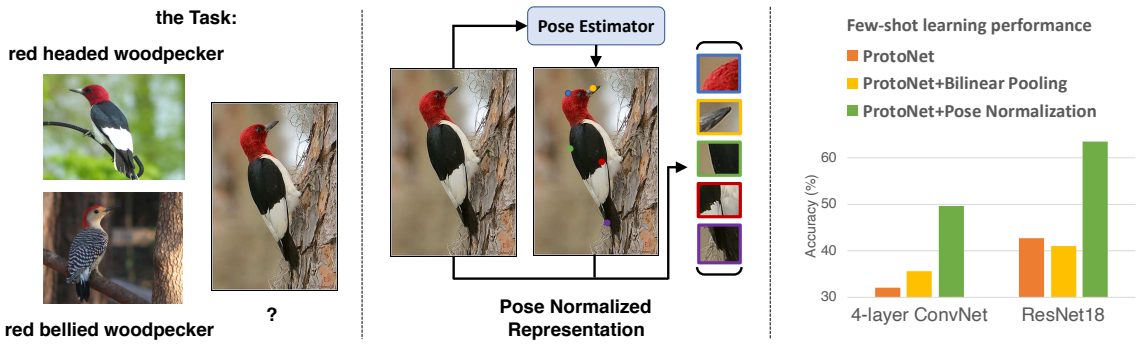

Figure 1. **Left:** The fine-grained few-shot recognition task. Objects share the same part structure and differences between categories are subtle. **Middle:** Based on a simple pose estimator, a pose-normalized representation can capture semantic part information. **Right:** On both shallow and deep backbones, pose normalization increases few-shot learning performance significantly. A shallow architecture with our representation (4-layer ConvNet+Pose Normalization) even outperforms a much deeper blackbox network without it (ResNet18).

usefulness in this setting. Pose normalization is implemented through an extremely simple modification to convolutional architectures, adding very few new parameters (in contrast to prior methods that increase network size by a factor of two or higher [34, 10]). Our method is orthogonal to the choice of few-shot learning technique and backbone neural architecture. We evaluate our approach on three different few-shot learning techniques, two differently-sized backbone architectures, and three fine-grained classification datasets of bird species and aircraft. We find that:

1. Pose normalization provides significant gains across the board, in some cases providing a more than 20 point improvement in accuracy, while requiring *no* part annotations for novel classes.

2. In all settings, pose normalization outperforms black-box modifications to the neural architecture, such as bilinear pooling.

3. The advantages of pose normalization are apparent even when as little as only 5% of the base class training data is annotated with pose.

4. Pose normalization is effective for both shallow and deep network architectures. Shallow networks with pose normalization *outperform* deeper blackbox ones.

The large performance gains we observe, along with the simplicity of the architecture itself, points to the power of pose normalization in fine-grained, few-shot classification.

2. Related Work

Fine-grained recognition is a classic problem in computer vision, and a recurring challenge [1]. While we focus on bird species classification [26], the presented ideas apply to other fine-grained tasks, such as identifying models of aircraft [17], cars [15], or any other problem where objects have a consistent set of parts. In the context of fine-grained recognition, Farrell et al. [6] proposed the idea of pose normalization: predicting the parts of the object and recording the appearance of each part as a descriptor. Many versions of the idea have since been explored, including varying the kind of parts [32, 10, 33], the part detector [31], and the combination of these ideas with neural networks [34]. The last of these is the most similar to our work. However, all of these approaches are concerned with fully supervised recognition, whereas here we look at few-shot recognition.

Pose normalization has also served as inspiration for black-box models where the parts are unsupervised. Lin et al. [16] introduce bilinear pooling as a generalization of such normalization, and we compare to this idea in our work. Spatial Transformer Networks [14] instantiate unsupervised pose normalization explicitly and train it end-to-end. Other instantiations of this intuition have also been proposed [4, 11, 21]. However, these unsupervised approaches add significant complexity and computation, making it difficult to discern the benefits of pose-normalization alone. In contrast, we focus on a lightweight, straightforward, semantic approach to show that pose normalization, not added network power, is responsible for improved performance.

Few-shot learning methods can be loosely organized into the following three groups: 1) Transfer learning baselines train standard classification networks on base classes, and then learn a new linear classifier for the novel classes on the frozen representation. Recent work has shown this to be competitive [3, 27, 18]. 2) Meta-learning techniques train a "learner": a function that maps small labeled training sets and test images to test predictions. Examples include ProtoNet [20], MatchingNet [25], RelationNet [22] and MAML [7]. These learners might sometimes include learnt data augmentation [28], which some methods train

using pose annotations [5]. 3) Weight generation techniques generate classification weights for new categories [8, 9].

Most few-shot learning methods use blackbox network architectures, which function well given enough labeled data, but may suffer in the highly constrained few-shot learning scenario. Wertheimer and Hariharan [29] revisit the bilinear pooling of Lin et al. [16] and find it to work well. They also introduce a simple, effective localization-normalized representation, but which is limited to coarse object bounding boxes instead of fine-grained parts. Zhu et al. [35] introduce a semantic-guided multi-attention module to help zero-shot learning, but is fully unsupervised. We compare to an unsupervised baseline in our experiments.

Pose normalization increases invariance to common modes of variation. An alternative to increasing invariance is to use learnt data augmentation [12, 28, 5]. However, this typically requires large additional networks and significant computation. Instead, we focus on a lightweight approach. Note also that one of our baselines [8] already outperforms a recent augmentation method [28].

In the following sections, we first overview few-shot recognition. We then show that pose-normalization of features can act as a plug-and-play network layer in a range of few-shot learning algorithms.

3. Few-Shot Recognition

The goal of few-shot learning is to build a *learner* that can produce an effective classifier given only a small labeled set of examples. In the classic few-shot setting, the learner is first provided a large labeled set (the *representation set*, D_{repre}) consisting of many labeled images from base classes Y_{base}. The learner must set its parameters, and any hyper-parameters, using this data. It then encounters a disjoint set of novel classes Y_{novel} from which it gets a small set of reference images D_{refer}. The learner must then learn a classifier for the novel classes from this set.

In most techniques, we can divide the learner into three modules: a feature-map extractor f_θ, a feature aggregator g_ϕ, and a learning algorithm h_w.

The feature map extractor f_θ is usually implemented as a deep convolutional neural network, with learnable parameters θ. For each input image x, the network yields the corresponding feature map tensor $\boldsymbol{F}{=}f_\theta(x){\in}\mathbb{R}^{C\times H\times W}$, where C, H, W denote respectively the channel, height, and width dimensions of the feature map.

The feature aggregator g_ϕ is a transformation parameterized by ϕ, converting feature maps into global feature vectors: $\boldsymbol{v}{=}g_\phi(\boldsymbol{F}){\in}\mathbb{R}^d$, where d is the latent dimensionality. Typically g_ϕ is a global average pooling module.

The learning algorithm h_w takes a dataset S of training feature vectors and corresponding labels, and a test feature vector $\boldsymbol{v}$, and outputs a probability distribution over labels $\hat{p}$ for the latter: $\hat{p}(x){=}h_w(\boldsymbol{v}, S)$. For our purposes we consider three representative methods:

Transfer learning follows the standard network pretraining and fine-tuning procedure. h_w is implemented by a simple linear classifier with a learned weight matrix and softmax activation. Functions f_θ, g_ϕ are trained concurrently with h_w, minimizing the standard cross-entropy loss over data in D_{repre}. To adapt the model to novel classes, feature extractor parameters θ, ϕ are frozen, and h_w trains a new linear classifier on the novel classes in D_{refer}.

Prototypical network [20] is a representative meta-learning method that produces a prototype representation for each class by averaging the feature vectors within that class. h_w is then a non-parametric classifier assigning class probabilities based on the distance between a datapoint's feature vector and each class prototype. Every training episode samples N classes from the base categories Y_{base}, and a small support set and query set of images from within each one. Support images form class prototypes, while N-way classification on the query set produces the loss, and corresponding update gradients to parameters θ, ϕ.

In *Dynamic few-shot learning* [8], h_w is once again a linear (or cosine) classifier, but instead of being directly fine-tuned on D_{refer}, the classifier is generated by a learnt *weight generator* G. The training process consists of two stages. The first is standard classification training on D_{repre}. During the second stage, the feature extractor parameters θ, ϕ are frozen. To train the generator G, the algorithm randomly picks several "fake" novel classes from Y_{base}, and treats them as if they were truly novel, performing classification with the classifier weights generated by G and minimizing the classification loss on simulated "test" examples from these classes.

4. Pose-Normalized Feature Vectors

Two intuitions motivate our proposed method. First, for fine-grained recognition, the difference in appearances between two classes tends to be extremely small. In the few-shot setting, it is even harder for an algorithm to capture these subtle differences, as only a few examples are available for reference. Using pose normalization to focus the feature representation on the most informative parts of each image should then benefit the learning process. Second, because fine-grained recognition involves similar kinds of objects, they are likely to share the same semantic structures. Thus it is highly probable that a pose estimator trained on base classes will generalize, even to unseen novel classes.

We assume M distinct parts. Part annotations are available for (some) base-class training samples in D_{repre}, but *not for novel classes*. We format part annotations for each image x as an $M{\times}H{\times}W$ location tensor $\boldsymbol{m}^*$, where $H{\times}W$ is the spatial resolution of the feature map.

We now present our method for extracting pose-

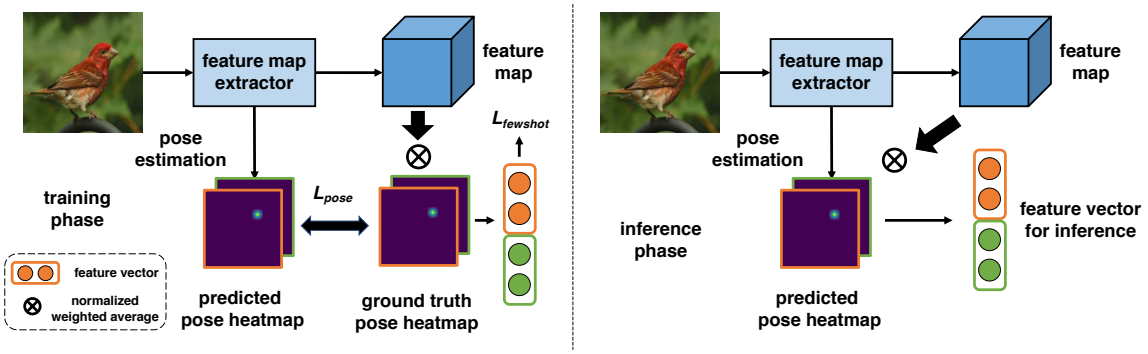

Figure 2. The pose normalization framework for training and inference. The pose estimator takes an intermediate output of the network backbone as input and generates pose heatmap predictions. The feature vector is calculated by applying each heatmap as an attention over the feature map. The final representation is the concatenation of these vectors. In this example, the number of parts M=2.

normalized feature vectors. For this, the network must first estimate pose. We use an extremely small, two-layer convolutional network q_ϕ. This operates on a feature map tensor $\boldsymbol{F}'\in\mathbb{R}^{C'\times H'\times W'}$ extracted from an intermediate layer of the feature map extractor f_θ. q_ϕ uses sigmoid activation in the final layer and produces a heatmap location prediction for all annotated parts $\boldsymbol{m}=q_\phi(\boldsymbol{F}')\in\mathbb{R}^{M\times H\times W}$. We deliberately use a small q_ϕ and *reuse* computation in f_θ to minimize the effect the additional parameters might have on the final performance of the classifier. Improved performance should indicate that *pose information* is useful for fine-grained few-shot learning, not a larger network.

Given the heatmap $\boldsymbol{m}$ and feature map $\boldsymbol{F}$, we must construct a feature vector $\boldsymbol{v}$. Each channel in $\boldsymbol{m}$ is applied as a spatial attention mask to the feature map, producing an attention-normalized feature vector. Concatenating these M part description vectors produces the final representation vector for the image. Formally, denoting $\boldsymbol{F}(h,w)\in\mathbb{R}^C$ as the feature vector for location (h,w) in feature map $\boldsymbol{F}$, and $m_i(h,w)\in\mathbb{R}$ as the heatmap pixel value at position (h,w) for the i-th part category, $\boldsymbol{v}\in\mathbb{R}^{CM}$ is calculated as:

$$\boldsymbol{v}_i = \frac{\sum_{h,w}^{H,W} \boldsymbol{F}(h,w)\cdot m_i(h,w)}{\epsilon + \sum_{h,w}^{H,W} m_i(h,w)} \tag{1}$$

$$\boldsymbol{v} = [\boldsymbol{v}_0,\ldots,\boldsymbol{v}_i,\ldots,\boldsymbol{v}_M] \tag{2}$$

where ϵ=10^{-5}. The loss during training is the sum of the pixel-wise log loss between the ground truth part location heatmap $\boldsymbol{m}^*$ and the predicted heatmap $\boldsymbol{m}$, and the original few-shot classification loss:

$$L_{pose} = -\frac{1}{MHW}\sum_{i,h,w}^{M,H,W} [m_i^*(h,w)\log m_i(h,w) + (1-m_i^*(h,w))\log(1-m_i(h,w))] \tag{3}$$

$$L_{total} = L_{fewshot} + \alpha\cdot L_{pose} \tag{4}$$

where α is a balancing hyper-parameter. To facilitate learning in the classification branch, feature vectors for few-shot classification are initially produced from the ground truth part annotation heatmap $\boldsymbol{m}^*$ instead of the predicted heatmap $\boldsymbol{m}$. Afterwards, the pose estimation network's parameters ϕ are frozen. In subsequent adaptation/fine-tuning and evaluation/inference stages on novel classes, feature vectors are calculated from the predicted heatmap $\boldsymbol{m}$. An overview of our approach is provided in figure 2.

Note that while we assume a fixed set of consistent part labels during training, we do not require parts to consistently appear across all objects, nor must any particular object contain all the specified parts. Thus, our pose estimator should generalize broadly: *any* fine-grained classification of objects that depends on the appearance of various parts (e.g., cars, furniture, insects) is amenable to this approach.

5. Experiments

5.1. Datasets and implementation details

We experiment with the **CUB dataset** [26] which consists of 11,788 images from 200 classes. It also includes 15 part annotations for each image, thus M=15. Following the evaluation setup in [29, 3], we randomly split the dataset into 100 base, 50 validation and 50 novel classes. Base category images form the representation set D^{CUB}_{repre}. For each validation and novel class, we randomly sample 20% of its images to form the reference set D^{CUB}_{refer}. The remaining novel images form the query set D^{CUB}_{query}, which is used for evaluating algorithms. *Note that our models have access to part annotations only in base classes*. No part annotation information is available for any image in the validation or novel classes, including both their reference and query sets.

NABird evaluation: There are only 50 novel classes in CUB's evaluation set, which can potentially make evaluation noisy. The accuracy differences between few-shot learning algorithms also decrease significantly in the pres-

ence of domain shift [3]. Thus, in order to verify the robustness and generalization capacity of our proposed method, we also evaluate our CUB models on another, much larger bird dataset: NABird [23] (NA), which, after removing overlap with CUB, contains 418 classes and 35,733 images. As before, we randomly sample 20% of images from each category to form the reference set D^{NA}_{refer}. The remaining images form the query set D^{NA}_{query}.

Network backbone: For the feature map extractor f_θ, previous work [20, 29, 8] adopts a standard architecture: a 4-layer, 64-channel convolution network with batch normalization and ReLU. In this setting, the input image size is 84×84 and the output feature map is $64\times10\times10$. Deeper backbones can significantly reduce the differences in performance between these methods [3], so in addition to the 4-layer network, we also train and evaluate a ResNet18 [13] backbone, with a few technical modifications that increase performance across all models. We change the stride of the last block's first convolution and downsampling layers from 2 to 1. The output size of the last block thus remains at 14×14 instead of 7×7. We also add a 1×1 convolution with batch normalization to the last layer of the original ResNet18, which reduces the number of channels from 512 to 32. The input size of our modified ResNet18 is still 224×224, but the output size becomes $32\times14\times14$.

Pose estimation module: The layers of the pose estimation network q_ϕ are composed as Conv-BN-ReLU-Conv, where Conv denotes 3×3 convolution. In the 4-layer ConvNet, q_ϕ takes as input the feature map after the second convolution. The number of input/output channels for the two convolution layers in q_ϕ are $64/30$ and $30/M$ where M is the number of part categories. In the ResNet18, q_ϕ takes the third block's feature map as input, and the corresponding convolution channel sizes are $256/64$ and $64/M$. It can be seen that the number of learnable parameters introduced by q_ϕ is small compared to the original backbone network.

5.2. Baseline methods

For the few-shot learning algorithm, we denote transfer learning, prototypical networks, and dynamic few-shot learning as transfer, proto, and dynamic, respectively. We compare our proposed pose normalization approach (PN) with the following feature aggregation methods, across all learning algorithms and network backbones:

Average pooling is the most straightforward method, commonly adopted in previous work. All subsequent models use average pooling when a feature aggregator is not otherwise specified.

We also present a baseline that trains this average-pooled feature extractor and classifier jointly with a localizer, with the latter discarded at test time. This **Multi-Task** model, denoted MT, examines whether pose estimation functions purely as a regularizer in few-shot training.

Bilinear pooling (BP) [16] is an effective module for expanding the latent feature space and increasing expressive power in fine-grained visual classifiers. Recent work [29] found that BP can be adapted to prototypical networks, improving performance without increasing parameter count.

Few-shot localization (FSL) [29] uses bounding box annotations in the representation and reference sets. The model learns to localize an object before classifying it, thus improving few-shot classification accuracy. Since this model's localizer is learnt in a prototypical way, it doesn't introduce any additional convolutional layers.

Bounding box normalization (bbN) is a more direct comparison to bounding box based methods that does not require box annotations for novel classes. We use the PN model but set $M{=}2$, and train the localizer to separate images into foreground/background regions based on the ground truth bounding boxes for base class training data.

Unsupervised pose normalization (uPN) is based on unsupervised localization [29], a competitive localization method where feature maps are partitioned into soft regions based on feature distance from a set of learned parameter vectors. Following the same core idea, we introduce $M{=}15$ learned, category-agnostic pose vectors, and spatially partition the feature map based on relative feature distance to each vector at each location. We mean-pool over the resulting 15 soft regions, as if they were 15 predicted part locations, to produce a feature vector for the classifier. The pose vectors are learned parameters, trained end-to-end and jointly with the classifier architecture, requiring no part annotations or separate localization module.

In addition, we include an **oracle** version of our model: **Pose normalization with ground truth pose** (PN_gt).

5.3. Few-shot recognition results

We first train all models on D^{CUB}_{repre}, using the validation set to select the best hyper-parameters and stopping point for each model. We then evaluate them on D^{CUB}_{query} using the limited set of labeled novel class images in D^{CUB}_{refer}. For the evaluation metric, we use the all-way evaluation [29, 12, 28] rather than the commonly adopted 5-way task. The algorithm is required to distinguish all novel classes simultaneously, a more challenging setup. For the number of reference images, we consider both the standard 1-shot/5-shot and the all-shot setting proposed by [29], i.e. utilizing all the labeled images for each novel category in D^{CUB}_{refer}.

For CUB, all-shot results are shown in table 1. For 1 and 5 shot settings, we plot the mean of 600 randomly generated test episodes in figures 3 and 4. The 95% confidence intervals are all less than 0.6 percentage points. Using the above models trained on CUB, we then do the same evaluation on NA, using D^{NA}_{refer} and D^{NA}_{query}. The number of novel classes in NA is large (418), and the number of images per class is unbalanced. We therefore only report the all-way

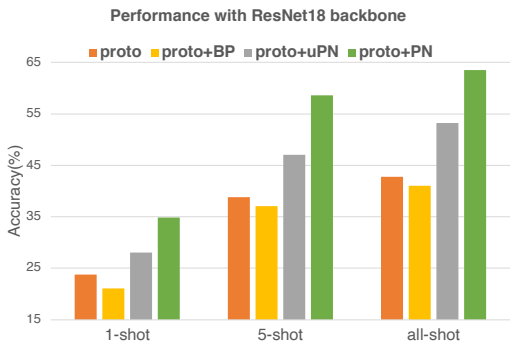

Figure 3. Accuracy comparison on CUB. All models use a ResNet18 prototypical network. Pose normalization dominates other methods under all settings.

all-shot results in table 2, with both mean accuracy over all test samples and mean accuracy per class. We average over 8 trials for the proto, proto+uPN, and proto+PN models in each of the above mentioned settings. 95% confidence intervals are all within 0.9 percentage points.

From these experimental results, we conclude that:

1. **Pose normalization provides significant and consistent performance gains over the (average-pooled) baseline.** Accuracy improves for both shallow and deep network backbones, for all three few-shot learning approaches, and for both evaluation datasets. Under the all-way, all-shot setting on CUB, the accuracy gain is consistently greater than **15 points** for the 4-layer ConvNet, across all three learning algorithms, and reaches **20** points on ResNet18. Shallow networks with pose normalization can even outperform their deeper counterparts.

2. In all settings, **pose normalization outperforms other aggregation functions**, including black-box modifications (bilinear pooling), techniques based on bounding box localization (FSL and bbN) and unsupervised pose normalization. It also outperforms multi-task training, indicating that normalization, rather than the additional auxiliary loss, is key.

3. **Pose information is more effective than coarse object location.** In table 1, PN and bbN contribute similar quantities of new learnable parameters, but the fine-grained pose information in PN causes it to outperform bbN, which only focuses on a coarse bounding box. By comparing PN with PN_gt, we see that a better pose estimator could potentially contribute an even larger boost to performance.

5.4. Impact of the number of pose annotations

While part locations are often cheaper to obtain than fine-grained expert class labels (see the careful labelling pipeline of [23]), it could still be the case that high-quantity part annotations are difficult to collect. We therefore consider an ablation of our model, where a limited number of training images have part annotations. For the remaining images, L_{pose} is not computed, and the predicted pose

Model	4-layer ConvNet	ResNet18
transfer	33.42	46.47
transfer+PN	**49.96**	**57.53**
transfer+PN_gt	56.40	58.54
proto	32.09	42.73
proto+MT	35.56	50.93
proto+BP	35.56	41.04
proto+FSL	39.60	47.43
proto+bbN	37.75	44.02
proto+uPN	46.24	53.18
proto+PN	**49.56**	**63.44**
proto+PN_gt	59.55	62.63
dynamic	35.77	43.27
dynamic+PN	**54.17**	**60.19**
dynamic+PN_gt	62.67	60.09

Table 1. Few-shot classification results for different models on the CUB dataset. Models are organized by few-shot learning algorithm, then by feature representation method. Pose normalization gives a significant performance boost for all three few-shot learning algorithms, with both shallow and deep network backbones.

	4-layer ConvNet		ResNet18	
Model	**mean**	**per-class**	**mean**	**per-class**
transfer	12.63	11.24	20.22	17.54
transfer+PN	**24.60**	**21.76**	**28.36**	**25.57**
proto	8.73	8.37	13.33	12.55
proto+MT	10.59	10.10	16.41	15.42
proto+BP	10.47	9.83	15.09	14.04
proto+FSL	12.34	11.61	15.62	14.81
proto+bbN	10.57	10.00	13.05	12.32
proto+uPN	18.91	17.51	22.12	20.77
proto+PN	**21.02**	**19.47**	**32.66**	**30.59**
dynamic	12.13	11.26	14.82	13.44
dynamic+PN	**26.17**	**24.07**	**30.10**	**27.86**

Table 2. Performance of CUB models on NA. The performance boost introduced by pose normalization is still significant in this new domain. Performance is consistent with CUB observations.

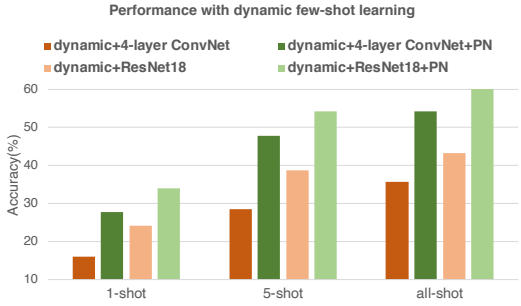

Figure 4. Performance comparison for dynamic few-shot learning models on CUB. The accuracy boost from pose normalization is significant and consistent.

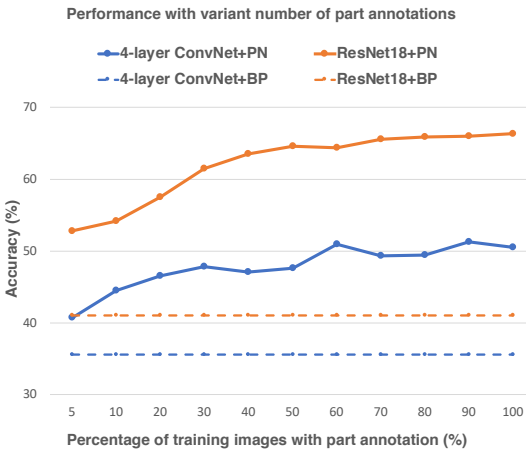

Figure 5. Few-shot test accuracy for pose normalization when part annotations are sparse. The performance drop is surprisingly small. Pose normalization outperforms bilinear pooling even when only 5% of annotations are available during training.

heatmap produces feature vectors for classifier training instead of the ground truth.

We evaluate prototypical networks on CUB with both shallow and deep backbones, and vary the percentage of images with part annotation. Results are given in figure 5. Pose normalization is highly robust to annotation sparsity during training (less than 5 points fluctuation when above 30% availability), and consistently outperforms BP even with as few as 5% pose annotations available.

5.5. Evaluation on FGVC-Aircraft

We evaluate the generality of these conclusions on fine-grained aircraft classification [17] (FGVC-Aircraft), which contains 10,000 images spanning 100 aircraft models. Following the same ratio as CUB, we split the classes into 50 base, 25 validation and 25 novel. The reference/query split is as described in Section 5.1. Since this dataset doesn't contain any part annotation, we use an independent dataset OID-Aircraft [24] (OID) to jointly train our pose normalization module. OID contains 6,357 images, ignoring those shared with FGVC, and 5 part annotations per image (thus $M{=}5$). OID contains no classification labels.

Each training iteration samples an image batch from OID and FGVC. OID images are used to calculate L_{pose}, while FGVC images use predicted pose heatmaps to get feature vectors. Results are shown in Table 3. Although the pose estimator is trained on disjoint images, it remains effective at boosting aircraft recognition performance. We conclude that pose normalization generalizes across fine-grained few-shot classification tasks. Extending this approach to non-fine-grained tasks or class-specific parts is not straightforward, but could be a valuable direction for future research.

	4-layer ConvNet			ResNet18		
Model	**1-shot**	**5-shot**	**all-shot**	**1-shot**	**5-shot**	**all-shot**
proto	24.40	43.24	52.06	46.27	63.15	67.76
proto+PN	**26.04**	**50.35**	**60.83**	**58.72**	**77.75**	**81.96**

Table 3. Few-shot results under all three evaluation settings on the FGVC-Aircraft dataset. Results averaged over five trials.

6. Analysis

6.1. Model interpretation

Accuracy notwithstanding, we would like for pose-normalized representations to be human-interpretable, unlike prior black-box representations. To investigate what the model actually learns, we conduct two experiments to analyze the learnt pose normalized representation. Both use the proto+PN model with a ResNet18 backbone.

Part importance: Every type of bird is likely to have a set of particularly distinguishable part attributes. To verify that our model learns this, we conduct the following test. For each class, we iterate over the parts and calculate the test accuracy when the corresponding part feature vector is removed from the representation. The magnitude of the resulting drop in accuracy can be construed as the *importance* of each part for this class as learned by the model. We visualize this learned importance for three species in figure 6 and compare it with species descriptions from a field guide. Our network scores largely conform to expert judgments.

Nearest neighbors: Different birds might share the same part attribute; for example, the California Gull and the Ring-billed Gull have the same beak shape. Therefore in pose normalization, the beak vectors for these two birds should be close, as part vectors are designed to encode regional information in a class-agnostic way. To verify this, we find the top-5 images in the reference set with the closest part vectors to a given vector from a query image/part pair. Four random examples are given in figure 7. Generally, our assumption holds - the vector describing the given part in each query image does generalize to other species.

6.2. Pose estimation

Following prior work on evaluating pose estimation [30, 2] we calculate the normalized PCK (normalized using the diagonal of the bounding box) at different thresholds for both shallow and deep network backbones. Results are given in figure 8. We see that both estimates can give accurate results. While the deeper network backbone does produce a better estimate, this boost is also quite limited. We believe that a more sophisticated pose estimator could lead to better results on few-shot recognition.

6.3. Unsupervised pose normalization

We note that unsupervised pose normalization also performs well from a classification perspective (see table 1).

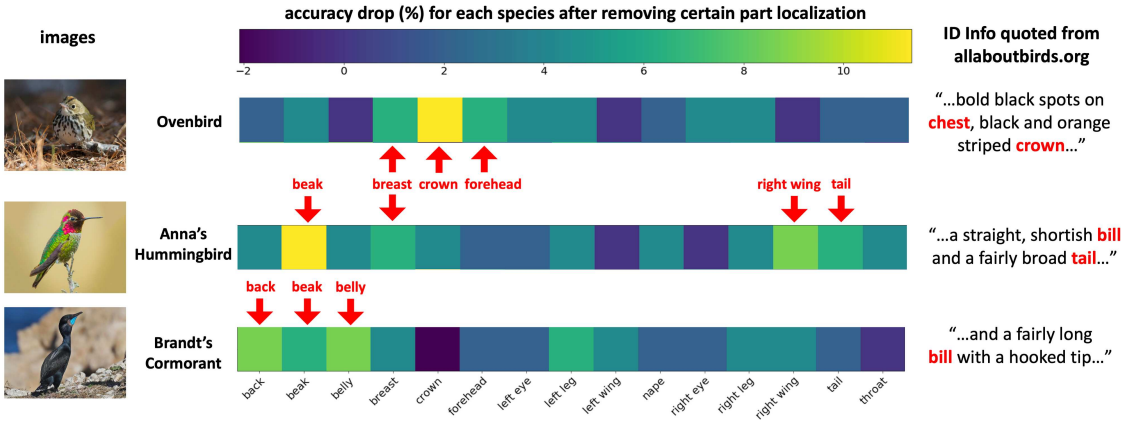

Figure 6. Visualizing accuracy drop for selected bird species when removing individual part vectors (part importance). On the right are quoted descriptions from bird experts on how to recognize those species. The estimated part importance matches well to expert judgments.

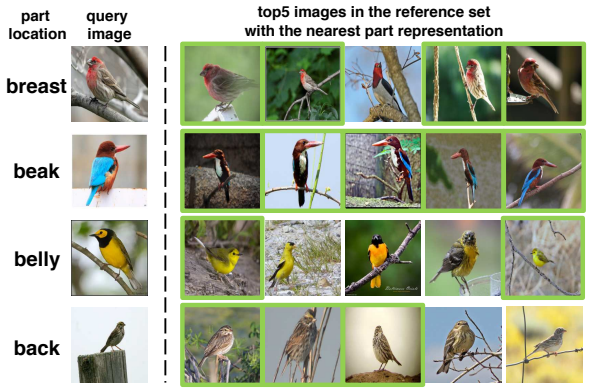

Figure 7. Images with the closest part vector to the query image, for a given part location. Images are labeled with a green box if it belongs to the same category as the query image. We see that part representations capture semantically meaningful attributes of the part location across classes.

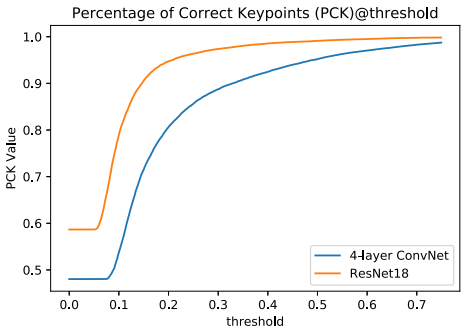

Figure 8. Pose estimation results for different network backbones.

As shown in Figure 9, the deeper backbone with unsupervised pose normalization does produce localized keypoints, which might help classification. However, observe that the semantic meaning of these keypoints is not consistent

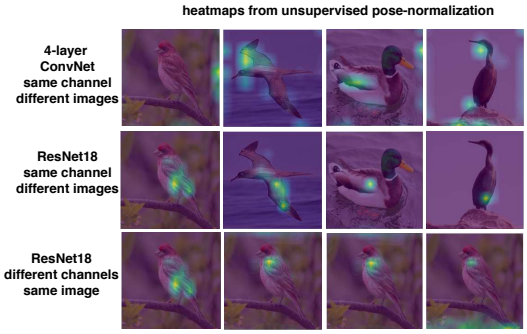

Figure 9. Visualization of unsupervised heatmaps. Semantic content is highly inconsistent, and difficult to interpret meaningfully.

across images (figure 9, top two rows). The prediction is also unstable, with different channels sometimes providing similar heatmaps (figure 9, bottom row). This inconsistency could help to explain why machine-discovered parts underperform hand-designed ones in fine-grained, few-shot classification.

7. Conclusion

We show that a simple, lightweight pose normalization module can lead to consistently large performance gains on fine-grained few-shot recognition without any part annotations at test time. Our results hold for shallow and deep network backbones, multiple few-shot learning algorithms, and multiple domains. In addition to significant accuracy improvements, we also show that pose-normalized representations are highly human-interpretable. We therefore highly recommend pose normalization as a general area for the fine-grained few-shot learning community to revisit.

Acknowledgements

This work was partly supported by a DARPA LwLL grant.

References

[1] https://sites.google.com/view/fgvc6/home.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[4] 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*, pages 764–773, 2017.

[5] Mandar Dixit, Roland Kwitt, Marc Niethammer, and Nuno Vasconcelos. Aga: Attribute-guided augmentation. In *CVPR*, 2017.

[6] Ryan Farrell, Om Oza, Ning Zhang, Vlad I Morariu, Trevor Darrell, and Larry S Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *2011 International Conference on Computer Vision*, pages 161–168. IEEE, 2011.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[8] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[9] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–30, 2019.

[10] Pei Guo and Ryan Farrell. Aligned to the object, not to the image: A unified pose-aligned representation for fine-grained recognition. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1876–1885. IEEE, 2019.

[11] Junwei Han, Xiwen Yao, Gong Cheng, Xiaoxu Feng, and Dong Xu. P-cnn: Part-based convolutional neural networks for fine-grained visual categorization. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[12] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[15] 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.

[16] 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.

[17] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[18] Akihiro Nakamura and Tatsuya Harada. Revisiting fine-tuning for few-shot learning. *arXiv preprint arXiv:1910.00216*, 2019.

[19] Lauren A Schmidt. *Meaning and compositionality as statistical induction of categories and constraints*. PhD thesis, Massachusetts Institute of Technology, 2009.

[20] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[21] Yifan Sun, Liang Zheng, Yali Li, Yi Yang, Qi Tian, and Shengjin Wang. Learning part-based convolutional features for person re-identification. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[22] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[23] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

[24] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, B. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. Weiss, B. Taskar, K. Simonyan, N. Saphra, and S. Mohamed. Understanding objects in detail with fine-grained attributes. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[25] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[26] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[27] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[28] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[29] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2878–2890, 2012.

[31] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[32] Ning Zhang, Ryan Farrell, and Trever Darrell. Pose pooling kernels for sub-category recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3665–3672. IEEE, 2012.

[33] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1644, 2014.

[34] Ning Zhang, Evan Shelhamer, Yang Gao, and Trevor Darrell. Fine-grained pose prediction, normalization, and recognition. *arXiv preprint arXiv:1511.07063*, 2015.

[35] Yizhe Zhu, Jianwen Xie, Zhiqiang Tang, Xi Peng, and Ahmed Elgammal. Learning where to look: Semantic-guided multi-attention localization for zero-shot learning. *arXiv preprint arXiv:1903.00502*, 2019.