Bird Species Categorization Using Pose Normalized Deep Convolutional Nets

Steve Branson[1*]
sbranson@caltech.edu
Grant Van Horn[2*]
gvanhorn@ucsd.edu
Serge Belongie[3]
tech.cornell.edu
Pietro Perona[1]
perona@caltech.edu

[1] California Institute of Technology
Pasadena, CA, USA
[2] University of California, San Diego
La Jolla, CA, USA
[3] Cornell Tech
New York, NY, USA
[*] Authors had equal contribution

Abstract

We propose an architecture for fine-grained visual categorization that approaches expert human performance in the classification of bird species. Our architecture first computes an estimate of the object's pose; this is used to compute local image features which are, in turn, used for classification. The features are computed by applying deep convolutional nets to image patches that are located and normalized by the pose. We perform an empirical study of a number of pose normalization schemes, including an investigation of higher order geometric warping functions. We propose a novel graph-based clustering algorithm for learning a compact pose normalization space. We perform a detailed investigation of state-of-the-art deep convolutional feature implementations [17, 22, 26, 28] and fine-tuning feature learning for fine-grained classification. We observe that a model that integrates lower-level feature layers with pose-normalized extraction routines and higher-level feature layers with unaligned image features works best. Our experiments advance state-of-the-art performance on bird species recognition, with a large improvement of correct classification rates over previous methods (75% vs. 55-65%).

1 Introduction

Fine-grained categorization, also known as subcategory recognition, is a rapidly growing subfield in object recognition. Applications include distinguishing different types of flowers [36, 37], plants [2, 29], insects [30, 35], birds [5, 10, 15, 19, 31, 43, 50, 51], dogs [27, 33, 38, 39], vehicles [42], shoes [4], or architectural styles [34]. Each of these domains individually is of particular importance to its constituent enthusiasts; moreover, it has been shown that the mistakes of state-of-the-art recognition algorithms on the ImageNet Challenge usually pertain to distinguishing related subcategories [41]. Developing algorithms that perform well within specific fine-grained domains can provide valuable insight into what types of models, representations, learning algorithms, and annotation types might be necessary to solve visual recognition at performance levels that are good enough for practical use.

Within fine-grained categorization, bird species recognition has emerged as one of the most widely studied areas (if not the most) in the last few years, in part due to the release of CUB-200 [45] and CUB-200-2011 [44] as standard datasets. Performance improvements on

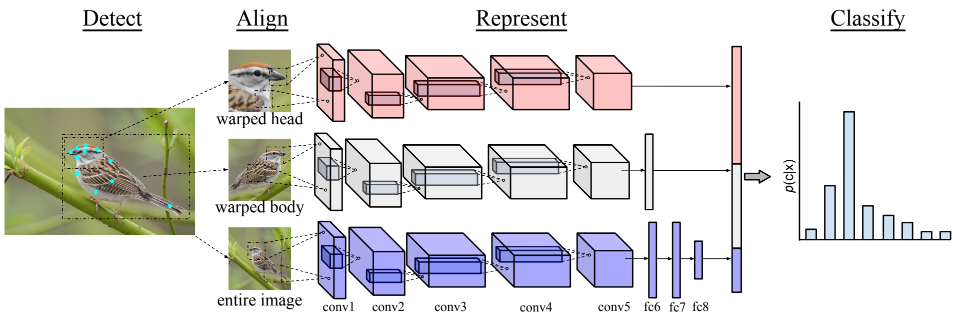


Figure 1: Pipeline Overview: Given a test image, we use groups of detected keypoints to compute multiple warped image regions that are aligned with prototypical models. Each region is fed through a deep convolutional network, and features are extracted from multiple layers. Features are concatenated and fed to a classifier.

the CUB datasets over the last few years have been remarkable, with early methods achieving $10-20\%$ 200-way classification accuracy [10, 44, 45, 47], and recent methods achieving $55-65\%$ accuracy [5, 12, 15, 17, 21, 51]. Here we report further accuracy gains up to 75.7%. Our approach extends earlier work on pose-normalized recognition [5, 12, 19, 51]–a two-staged recognition in which part detection precedes feature extraction for fine-grained classification. This paper makes 3 main contributions:

1. An empirical study of pose normalization schemes for fine-grained classification, including an investigation of higher order geometric warping functions and a novel graph-based clustering algorithm for learning a compact pose normalization space.
2. A detailed experimental investigation of state-of-the-art deep convolutional features [17, 22, 26, 28] and feature learning for fine-grained classification.
3. A completely automated system for detecting birds in images, locating their parts, and classifying the species of each bird. The system delivers unprecedented performance.

2 Related Work

Work on fine-grained categorization over the past 5 years has been extensive. Areas explored include feature representations that better preserve fine-grained information [35, 46, 47, 48], segmentation-based approaches [1, 13, 14, 15, 21, 37] that facilitate extraction of purer features, and part/pose normalized feature spaces [5, 6, 19, 33, 38, 39, 43, 50, 51]. Among this large body of work, it is a goal of our paper to empirically investigate which methods and techniques are most important toward achieving good performance. Consequently, we describe a simple pose normalization method that can be used to express many of the above techniques and logical extensions. We find that a similarity based pose warping function as used by Berg and Belhumeur [5] yields the best performance and can be improved by using more parts to estimate the warping, while being made more compact and efficient by learning pose regions. We investigate the interplay between pose-normalized images and the types of features that work best with them.

The impressive performance of deep convolutional networks [32] (CNNs) on large scale visual recognition challenges, ignited by [28], has motivated researchers to adapt CNNs that were pre-trained on ImageNet to other domains and datasets, including Caltech-101 [49], Caltech-256 [49], VOC detection [22], and VOC classification [49]. Donahue et al. [17] extracted CNN features from part regions detected using a DPM, obtaining state-of-the-art results in bird species classification. Our work is inspired by these results, and we improve on them by combining ideas inspired from fine-grained recognition and CNN research. In particular, we find that different layers of the CNN are appropriate for different levels of alignment. Secondly, we explore different methods for fine-tuning CNN weights on the CUB-200-2011 training set, inspired by techniques and results from Girshick et al. [22].

3 Pose Normalization Schemes

In this section, we define a class of pose normalization schemes based on aligning detected keypoints to the corresponding keypoints in a prototype image. In Section 3.2, we introduce an algorithm for learning a set of prototypes that minimizes the pixel-wise alignment error of keypoint annotations in a training set and works for arbitrary warping functions.

3.1 Pose Normalization By Prototypical Regions

Let $\{(X_i, Y_i)\}_{i=1}^{n}$ be a training set of n images and ground truth part annotations, where each annotation $Y_i = \{y_{ij}\}_{j=1}^{K}$ labels the pixel location and visibility of K 2D keypoints in the image X_i. Due to its simplicity and ease of collection, this style of 2D keypoint annotations is widely used (*e.g.*, for birds [44], dogs [33], faces [24], and humans [9]).

Let $\Psi(X,Y) = [\psi_p(X,Y)]_{p=1}^{P}$ be a feature vector that is obtained by concatenating P pose normalized feature spaces, where each $\psi_p(X,Y)$ may correspond to a different part or region of an object and can be estimated using some subset of keypoints in Y. We consider a simple definition of $\psi_p(X,Y)$ based on prototypical examples. Let the p-th prototype $R_p = \{i_p, b_p, S_p\}$ consist of a reference image i_p, a rectangle b_p defining a region of interest in X_{i_p} for feature extraction, and a set of keypoint indices S_p. Given a test image X_t with detected keypoints Y_t, we solve for the transformation $W(y_{tj}, w)$ in some class of warping functions $\mathcal{W}$ that best aligns the corresponding keypoints in Y_t to Y_{i_p}:

$$w_{tp}^{*} = \arg\min_{w \in \mathcal{W}} \sum_{j \in S_p} E(y_{tj}, R_p, w), \quad \text{where } E(y_{tj}, R_p, w) = \|\hat{y}_{i_p j} - W(y_{tj}, w)\|^2 \qquad (1)$$

where $\|\cdot\|$ indicates Euclidean distance, and $\hat{y}_{i_p j}$ is a version of $y_{i_p j}$ after normalizing by the bounding box b_p (by subtracting the upper-left coordinate and dividing by the width/height). The induced pose normalized feature space $\psi_p(X,Y) = \phi(X(w_{tp}^{*}))$ is obtained by applying this warp to the image X_t and then extracting some base feature $\phi(X)$, where $X(w)$ is a warped version of image X.

In Table 1, we define how Eq 1 can be computed for many different warping families, including simple translations, similarity transformations (2D rotation, scale, and translation), and affine transformations. The extension to other families such as homographies and thin-plate-splines [3, 8] is straightforward. The above transformations have simple closed form solutions and have well understood properties in approximating projective geometry of 3D objects. In each case, the applicable warping function is only well-defined if the number of points available $|S|$ is sufficiently large. Let $S \subseteq S_p$ be the subset of points in S_p that are visible as determined by detected keypoints Y_t. If $|S|$ falls below the applicable minimum threshold, we set the induced feature vector $\psi_p(X,Y)$ to zero.

3.2 Learning Pose Prototypes

In this section, we introduce an algorithm for learning pose prototypes from training images with keypoint annotations. The approach has a similar objective to a poselet learning algorithm [9]. The main difference is that our approach generalizes to arbitrary warping schemes while explicitly optimizing pixel-wise alignment error in the induced feature space.

Name	$W(y,w)$	Solve w^*_{tp}	# Pts
Translation	$y = y_t + T$	$T = \mu_i - \mu_t$	$\lvert S\rvert \geq 1$
2D Similarity	$y = sRy_t + t$	$R = V\text{diag}(1, \det(VU^\top))U^\top, \ s = \frac{\text{tr}(\bar{M}_i^\top R\bar{M}_t)}{\text{tr}(\bar{M}_t^\top \bar{M}_t)}, \ T = \mu_i - sR\mu_t$	$\lvert S\rvert \geq 2$
2D Affine	$y = Ay_t^h$	$A = M_i M_t^{h\top}(M_t^h M_t^{h\top})^{-1}$	$\lvert S\rvert \geq 3$

Table 1: Computation of warping function $W(y,w)$ from detected points $Y_t[S]$ to a prototype $Y_i[S]$ for different warping families. In the above notation, let M_t and M_i be $2 \times |S|$ matrices obtained by stacking points in Y_t and Y_i, and μ_t and μ_i be their means. Let $\bar{M}_t$ and $\bar{M}_i$ denote mean subtracted versions of these matrices, and the superscript h denote points in homogeneous coordinates. Let $C = U\Sigma V^\top$ be the SVD of $C = \bar{M}_t \bar{M}_i^\top$.

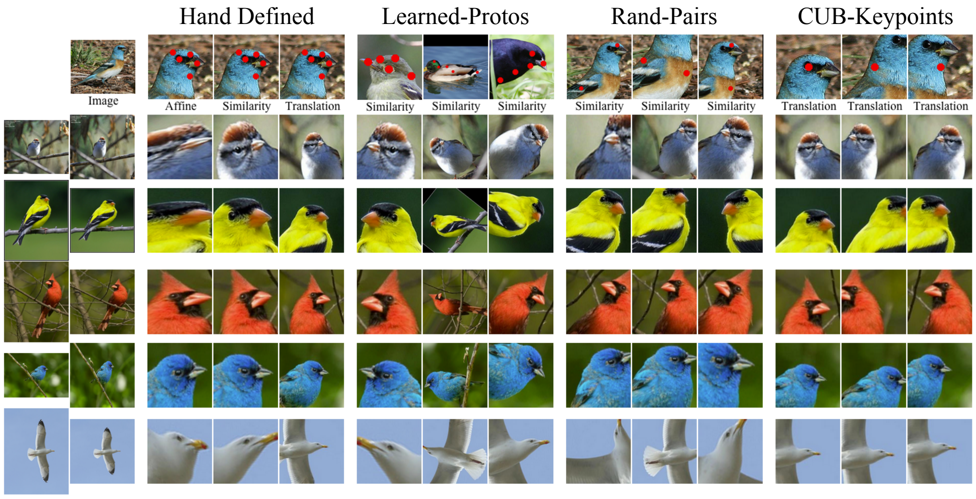


Figure 2: Example Warped Regions: The top row visualizes different prototypes, each of which defines a region of interest and multiple keypoints that are used to estimate a warping. The bottom rows show the resulting warped regions $X(w^*_{tp})$ when 5 images are aligned with each prototype. The 4 groupings of warped regions represent 4 baseline experiments analyzed in Table 3, which includes 1) Hand-Defined head or body regions, 2) the 1st 3 prototypes learned using our method from Section 3.2, 3) Rand-Pairs, which simulates [[illegible]], 4) CUB-Keypoints, which simulates [[illegible]]. In general, we see that a similarity transform captures the scale/orientation of an object better than a translation, while an affine transformation sometimes overly distorts the image. Using more points to estimate the warping allows for non-visible keypoints and ambiguous image flipping problems to be handled consistently.

Recall that $E(y_{tj}, R_p, w^*_{tp})$ is the squared pixel alignment error in the induced pose-normalized feature space when matching image t to prototype R_p. We attempt to learn a set of prototypes $\{R_p\}_{p=1}^P$ that minimizes the alignment error under the constraint that each keypoint y_{tj} in the training set must be aligned with low error to at least one prototype. The intuitive justification is that all portions of an object – each of which might contain important discriminative information – should be consistently aligned in at least one component of the feature space $\Psi(X,Y) = [\psi_p(X,Y)]_{p=1}^P$. Our goal is to choose a set of prototypes $\mathbf{R}^*$ that optimizes the objective

$$\mathbf{R}^* = \arg\min_{\mathbf{R}} \lambda P + \frac{1}{nK}\sum_{t=1}^{n}\sum_{j=1}^{K}\min_p E(y_{tj}, R_p, w^*_{tp}) \tag{2}$$

where the first term penalizes the number of prototypes selected and the second term minimizes pixel-wise alignment error, with λ being a tweakable tradeoff parameter. The optimization problem can be complex due to the possibility of complex warping functions w^*_{tp} (Eq 1) and prototype definitions. To make the problem tractible, we consider candidate prototypes anchored by a keypoint in the training set (nK candidates in total). Given an anchor y_{ik} representing the k-th keypoint in image i, we define the candidate prototype $R_{ik} = \{i, S_{ik}, b_{ik}\}$ in terms of a set S_{ik} of the M-nearest neighbors in Y_i to y_{ik}, and b_{ik} as an expanded bounding box around those keypoints.

We can solve Eq. 2 as a non-metric facility location problem [18]. Given predefined costs c_{lm} of connecting city l to facility m and costs o_m of opening facilities, the goal is to open a subset of facilities with minimum cost such that each city must be connected to one facility. Eq 2 reduces to a facility location problem, where each anchor point y_{ik} is a candidate facility with open cost λ, and each keypoint y_{tj} is a city that can be connected to a facilty with cost $c_{tj,ik} = E(y_{tj}, R_{ik}, w^*_{t,ik})$. A nice property of facility location problems is that, unlike some clustering algorithms like k-means, a fast greedy algorithm [25] has good approximation guarantees (1.61 [25] when the city-facility costs are metric, $1 + \ln P$ [23] for non-metric costs). This algorithm requires precomputing pairwise costs $c_{tj,ik}$ and sorting them. Examples of learned prototypes can be seen in Fig 2.

4 Deep Convolutional Features

Our pose-warped image regions $\{X(w^*_{tp})\}_p$ are each fed into a feature extractor $\phi(X)$, where $\phi(X)$ is the output of one or more layers of a deep convolutional neural network (CNN) [28]. We use the network structure from Krizhevsky et al. [28].

4.1 Multi-Layer CNN Features For Different Alignment Models

The progression through the 8-layer CNN network can be thought of as a progression from low to mid to high-level features. The later layers aggregate more complex structural information across larger scales–sequences of convolutional layers interleaved with max-pooling are capable of capturing deformable parts, and fully connected layers can capture complex co-occurence statistics. On the other hand, later layers preserve less and less spatial information, as max-pooling between each convolutional layer successively reduces the resolution of the convolutional output, and fully connected layers drop semantics of spatial location. We thus hypothesize (and verify empirically in Section 5), that different layers of this pipeline are more appropriate for different alignment models, and combining multiple levels of alignment can yield superior performance.

Our final feature space concatenates features from multiple regions and layers, and one-vs-all linear SVMs are used to learn weights on each feature. The use of an SVM (instead of the multiclass logistic loss used by CNNs) is primarily for technical convenience when combining multiple regions. To handle layers with different scales of magnitude, each CNN layer output is normalized independently during feature extraction. In the next section, we explore a few different approaches for training the internal weights of the CNN.

4.2 Training the Convolutional Neural Net

We consider 4 training/initialization methods:

Pre-Trained ImageNet Model: This corresponds to the methodology explored in [17], where the CNN is pre-trained on the 1.2 million image ImageNet dataset and used directly as a feature extractor.

Fine-Tuning the ImageNet Model: This corresponds to the methodology explored in [22]. Here, the final 1000-class ImageNet output layer is chopped off and replaced by a 200-class CUB-200-2011 output layer. The weights of the new layer are initialized randomly, and stochastic gradient descent (SGD) and back propagation are used to train the entire network jointly with a small learning rate. Because the last layer is new and its weights are random, its weights are likely much further from convergence than the pre-trained ImageNet layers. Consequently, its learning is increased by a factor of 10.

Two Step Fine-Tuning Method: We explore a 2nd possible fine-tuning method that aims to avoid using unbalanced learning rates for different layers. Here, we use the same network structure as for the previous method. We use a two step process. In the first step, we fix the weights of the old ImageNet layers and learn the weights of the new 200-class output layer–this is equivalent to training a multiclass logistic regression model using the pre-trained ImageNet model as a feature extractor. It is a fast, convex optimization problem. This fixes the problem of initializing the new layer. SGD and back propagation are then used to jointly train all weights of the entire network, where each layer is given the same learning rate. To our knowledge, this initialization scheme has not yet been explored in earlier work.

Training From Scratch: The earlier three approaches can be seen as an application of transfer learning, where information from the ImageNet dataset has been used to train a better classifier on a different set of classes/images. To help differentiate between gains from more training data and the network structure of the CNN, we investigate training the CNN without ImageNet initialization. Weights are initialized randomly before training with SGD.

5 Experiments

We evaluate performance on the CUB-200-2011 dataset [44], a challenging dataset of 200 bird species and 11,788 images. The dataset includes annotations of 15 semantic keypoint locations. We use the standard train/test split and report results in terms of classification accuracy. Although we believe our methods will generalize to other fine-grained datasets, we forgo experiments on other datasets in favor of performing more extensive empirical studies and analysis of the most important factors to achieving good performance on CUB-200-2011. Specifically, we analyze the effect of different types of features, alignment models, and CNN learning methods. We believe that the results of these experiments will be informative and useful to researchers who work on object recognition in general.

Implementation Details: We used the DPM implementation from [11], which outputs predicted 2D locations and visibility of 13 semantic part keypoints. To learn pose prototype regions, we chose $\lambda = 8^2$, which means that a new prototype should be added if it reduces the average keypoint alignment error by 8 pixels. For our best classifier, we concatenated features extracted from each prototype region with features extracted from the entire image.

We used the Caffe code base from Jia [26] to extract, train, and fine-tune the CNN with the default structure and parameter settings. When extracting feature outputs from different CNN layers, we use the names *conv3*, *conv4*, *conv5*, *fc6*, and *fc7*, where *conv* denotes a convolutional layer, *fc* denotes a fully connected layer, and the number indicates the layer number in the full CNN. We appended these names with the suffix *-ft* to denote features extracted on a CNN that was fine-tuned on CUB-200-2011. To fine-tune the CNN, we set the base learning rate to 0.001.

5.1 Summary of Results and Comparison to Related Work

Table 2 summarizes our main results and comparison to related work. Our fully automatic approach achieves a classification accuracy of 75.7%, a 30% reduction in error from the highest performing (to our knowledge) existing method [17]. We note that our method does not assume ground truth object bounding boxes are provided at test time (unlike many/most

Method	Oracle Parts	Oracle BBox	Part Scheme	Features	Learning	% Acc
POOF [5]		✓	Sim-2-131	POOF	SVM	56.8
Alignments [21]		✓	Trans-X-4	Fisher	SVM	62.7
Symbiotic [15]		✓	Trans-1-1	Fisher	SVM	61.0
DPD [51]		✓	Trans-1-8	KDES	SVM	51.0
Decaf [17]		✓	Trans-1-8	CNN	Logistic Regr.	65.0
CUB [44]			Trans-1-15	BoW	SVM	10.3
Visipedia [12]			Trans-1-13	Fisher	SVM	56.5
Ours			Sim-5-6	CNN	SVM+CNN-FT	**75.7**
CUB Loc. [44]	✓	✓	Trans-1-15	BoW	SVM	17.3
POOF Loc. [5]	✓	✓	Sim-2-131	POOF	SVM	73.3
Ours Loc.	✓	✓	Sim-5-6	CNN	SVM+CNN-FT	**85.4**

Table 2: Comparison to Related Work on CUB-200-2011: Our method significantly outperforms all earlier methods to our knowledge, both in terms of fully automatic classification accuracy (top grouping), and classification accuracy if part locations are provided at test time (bottom grouping). We categorize each method according to 4 axes which we believe significantly affect performance: 1) *Level of automation*, where column 2-3 indicate whether or not parts or object bounding boxes are assumed to be given at test time, 2) *Part localization scheme* (column 4), using the naming scheme Transformation-X-Y, where Transformation indicates the image warping function used (see Table 1), X indicates the number of keypoints/base-parts used to warp each region, and Y indicates the number of pose regions used, 3) *Type of features* (column 5), and 4) *Learning algorithm* (column 6), where CNN-FT is short for CNN fine-tuning.

methods). If we assume ground truth part locations are provided at test time, accuracy is boosted to 85.4%. These results were obtained using prototype learning using a similarity warping function computed using 5 keypoints per region, CNN fine-tuning, and concatenating features from all layers of the CNN for each region.

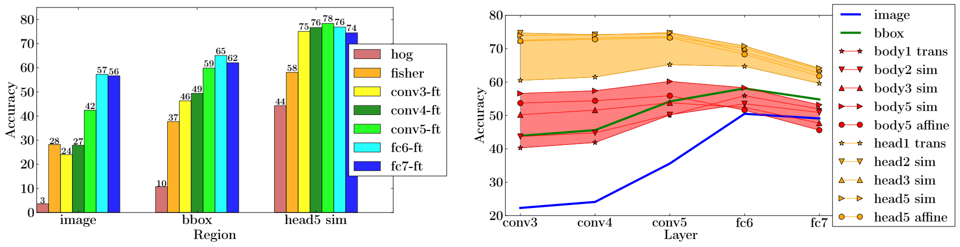
We attempt to categorize each related method according to part localization scheme, features used, and learning method. See the caption of Table 2 for details. The major factors that we believe explain performance trends and improvements are summarized below:

1. Choice of features caused the most significant jumps in performance. The earliest methods that used bag-of-words features achieved performance in the $10-30\%$ range [44, 50]. Recently methods that employed more modern features like POOF [5], Fisher-encoded SIFT and color descriptors [40], and Kernel Descriptors (KDES) [7] significantly boosted performance into the $50-62\%$ range [5, 12, 15, 21, 51]. CNN features [28] have helped yield a second major jump in performance to $65-76\%$.
2. Incorporating a stronger localization/alignment model is also important. Among alignment models, a similarity transformation model fairly significantly outperformed a simpler translation-based model. Using more keypoints to estimate warpings and learning pose regions yielded minor improvements in performance.
3. When using CNN features, fine-tuning the weights of the network and extracting features from mid-level layers yielded substantial improvements in performance beyond what had been explored in [17].

We support these conclusions by performing lesion studies in the next 3 sections.

5.2 Comparing Feature Representations

We performed experiments to quantify the effect of different image features and their performance properties with different alignment models. We compare CNN features from different layers to HOG [16] and Fisher-encoded [40] color and SIFT features while controlling for other aspects of our algorithms. HOG is widely used as a good feature for localized models, whereas Fisher-encoded SIFT is widely used on CUB-200-2011 with state-of-the-art results [12, 15, 21]. For HOG, we use the implementation/parameter settings of [20] and



(a) Feature Performance Comparison (b) Effect of CNN Layers For Different Regions

Figure 3: Effect of features and region type on CUB-200-2011: (a) CNN features significantly outperform HOG and Fisher features for all levels of alignment (image, bounding box, head). **(b)** Comparing classification performance for different CNN layers and regions if we assume ground truth part locations are known at test time (no fine-tuning used), we see that 1) features extracted from the head (yellow tube) significantly outperform other regions, 2) The later fully connected layers (fc6 & fc7) significantly outperform earlier layers when a crude alignment model is used (image-level alignment), whereas convolutional layers (conv5) begin to dominate performance as we move to a stronger alignment model (from image $\rightarrow$ bbox $\rightarrow$ body $\rightarrow$ head), 3) Using a similarity warping model significantly outperforms a translation model (width of the red and yellow tubes), and slightly outperforms an affine model, 4) Using more points (from 1 to 5) to estimate the warping improves performance for the body, whereas 2 points is sufficient for the head.

induce a $16 \times 16 \times 31$ descriptor for each region type. For Fisher features, we use the implementation and parameter settings from [12]. We summarize the results below:

CNN features significantly improve performance: In Fig 3, we see that CNN features significantly outperform other features for all levels of alignment, 57.3% vs. 28.2% for image-level features, and 78.4% vs. 58.1% for a similarity-aligned head. HOG performs well only for aligned regions (the head), while Fisher features perform fairly well across different levels of alignment.

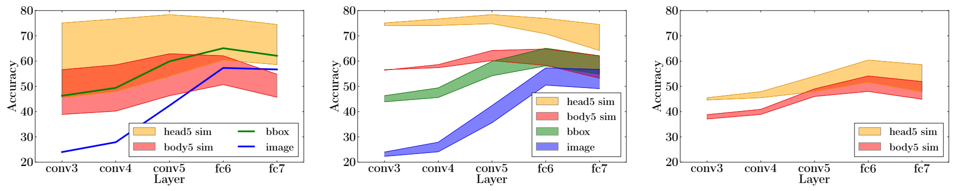
Different layers of the CNN are appropriate for different alignment models: In Fig. 3(b), we see that the later fully connected layers of the CNN (fc6 & fc7) significantly outperform earlier layers when a crude alignment model is used (57.3% vs 42.4% for image-level alignment), whereas convolutional layers (conv5) begin to dominate performance as we move to a stronger alignment model (from image $\rightarrow$ bbox $\rightarrow$ body $\rightarrow$ head).

5.3 Comparing Part Localization Schemes

We next perform experiments to quantify the effect of our pose normalization scheme, including the effect of the type of warping function used, a comparison of different methods of combining multiple pose regions, and the effect of imperfect part detection.

A similarity alignment model works best: In Fig. 3(b), we compare the effect of different choices of warping functions (translation, similarity, and affine) and the number of keypoints used to estimate them. We see that a similarity warping model significantly outperforms a translation model and slightly outperforms an affine model (on the head region, 74.8% for similarity vs. 65.2% for translation vs. 73.3% for affine). Secondly, we see that using more points (from 1 to 5) to estimate the warping improves performance for the body, whereas 2 points is sufficient for the head.

Combining multiple regions improves performance: In Table 3, we compare different strategies for combining multiple pose regions. We note that combining multiple regions improves performance over the best single region: 85.4% vs. 78.4% for the head. We compare to several different baseline methods for inducing a multi-region feature space while keeping our feature implementation fixed. The Proto-Learn method employs our pose



(a) Improvement From GT Parts (b) Effect of Fine-Tuning, GT Parts (c) Effect of Fine-Tuning, Pred Parts

Figure 4: Effect of fine-tuning and ground truth parts on CUB-200-2011: (a) If ground truth parts were available at test time or part detection could be improved, performance would be improved significantly (width of red/yellow tubes, with fine-tuning). **(b)** Fine-tuning significantly improves performance for all alignment levels (width of each tube). Improvements occur for all CNN layers; however, the effect is largest for fully connected layers. **(c)** The same effect holds for automated part prediction.

learning scheme from Section 3.2 using a similarity warping model and slightly outperforms other methods while being compact. Rand-Pairs simulates the alignment method used by POOF [5], where random pairs of keypoints induce similarity-aligned regions. CUB-Keypoints simulates the method used by [12] (among others), where each detected keypoint directly induces a surrounding pose region. Head-Body represents a baseline of expert-defined regions, and concatenates hand-defined similarity-aligned head and body regions with image and bounding box features.

Head Body (2)	Proto-Learn (6)	Rand-Pairs (6)	Rand-Pairs (30)	CUB-Keypoints (13)
83.7	85.4	83.2	84.1	79.6

Table 3: Comparing Different stategies for combining multiple regions when part locations are given at test time. The number in parentheses indicates the number of regions used for each method.

Imperfect part detection causes a significant but manageable drop in performance: In Fig. 4(a), we visualize the drop in performance caused by using detected vs. ground truth parts for different regions, which results in a drop in performance from 85.4% to 75.7% when combining regions. This is a sizeable drop in performance that we hope to reduce in future work by improving our part detection method; however, this gap is also surprisingly small, in large part due to the excellent performance of CNN features on image-level features.

5.4 Comparing CNN Learning Methods

In this section, we compare different strategies for learning the internal weights of the CNN.

Fine-tuning CNN weights consistently improves performance: In Fig. 4(b)-4(c), we compare performance when using the pre-trained ImageNet model as a feature extractor vs. fine-tuning the ImageNet model on the CUB-200-2011 dataset (see Section 4.2 for details). We see that fine-tuning improves performance by $2-10\%$, and improvements occur for all region types (image, bounding box, head, body), all CNN layers, and both on predicted and ground truth parts.

ImageNet pre-training is essential: The default CNN implementation was pre-trained on ImageNet and performance improvements come in part from this additional training data. We tried training the same network structure from scratch on the CUB-200-2011 dataset over 5 trials with random initialization. Performance was significantly worse, with 10.9% and 54.7% accuracy on image-level and similarity-aligned head regions, respectively (compared to 57.0% and 78.6% performance with pre-training) The problem relates to overfitting–the CNN model has 60 million learnable parameters [28] and the CUB-200-2011 dataset has < 6000 training images. Learning converged to near zero training error for both fine-tuning and training from scratch.

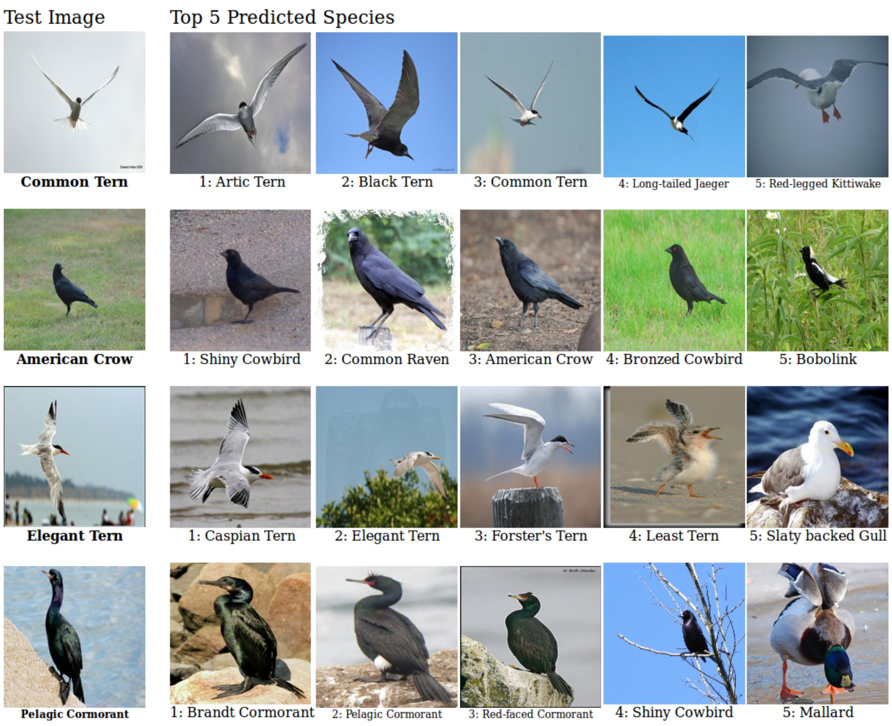


Figure 5: Most Misclassified Species and Failure Cases: Each row shows a random misclassified test example from the top 4 most misclassified species (Common Tern, American Crow, Elegant Tern, Pelagic Cormorant), and the 5 highest scoring predicted classes according to our fully automated computer vision system. Each predicted species is visualized using the training image that is the nearest neighbor in feature space. Common sources of failures include 1) highly related species (*e.g.*, terns), 2) black birds that are very similar in terms of color and texture, and 3) birds that are very shape deformable (*e.g.*, cormorants and flying birds).

The two step fine-tuning method yields more reliable improvements: Over 5 random trials, our proposed 2-step fine-tuning method improved average accuracy on both the image and head regions by about 2% compared to the method used in [22] (57.0% and 78.6% compared to 55.1% and 76.9%).

6 Conclusion

In this paper, we reduced the error rate on CUB-200-2011 by 30% compared to previous state-of-the-art methods, and analyzed which design decisions were most important to achieving good performance. Our method is based on part detection and extracting CNN features from multiple pose-normalized regions. Performance improvements resulted in large part from 1) using CNN features that were fine-tuned on CUB-200-2011 for each region, 2) using different CNN layers for different types of alignment levels, 3) using a similarity-based warping function that is estimated using larger numbers of detected keypoints. We also introduced a novel method for learning a set of pose regions that explicitly minimizes pixel alignment error and works for complex pose warping functions. In future work, we hope to apply our methods to other fine-grained datasets and explore customized CNN network structures and their training.

7 Acknowledgments

This work is supported by a Google Focused Research Award.

References

[1] Anelia Angelova and Shenghuo Zhu. Efficient object detection and segmentation for fine-grained recognition. In *CVPR*, 2013.

[2] Peter Belhumeur, Daozheng Chen, Steven Feiner, David Jacobs, W. Kress, Haibin Ling, Ida Lopez, Ravi Ramamoorthi, Sameer Sheorey, Sean White, and Ling Zhang. Searching the world's herbaria. In *ECCV*, 2008.

[3] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *NIPS*, volume 2, page 3, 2000.

[4] Tamara L Berg, Alexander C Berg, and Jonathan Shih. Automatic attribute discovery and characterization from noisy web data. In *ECCV*. 2010.

[5] Thomas Berg and Peter N Belhumeur. POOF: Part-based one-vs-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013.

[6] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*, 2014.

[7] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *NIPS*, 2010.

[8] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6): 567–585, 1989.

[9] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *ICCV*, 2009.

[10] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *ECCV*. 2010.

[11] Steve Branson, Oscar Beijbom, and Serge Belongie. Efficient large-scale structured learning. In *CVPR*, 2013.

[12] Steve Branson, Grant Van Horn, Catherine Wah, Pietro Perona, and Serge Belongie. The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization. *IJCV*, 2014.

[13] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Bicos: A bi-level co-segmentation method. In *ICCV*, 2011.

[14] Yuning Chai, Esa Rahtu, Victor Lempitsky, Luc Van Gool, and Andrew Zisserman. Tricos. In *ECCV*, 2012.

[15] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *ICCV*, 2013.

[16] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[17] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[18] Donald Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 1978.

[19] R. Farrell, O. Oza, N. Zhang, V.I. Morariu, T. Darrell, and L.S. Davis. Birdlets. In *ICCV*, 2011.

[20] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[21] E. Gavves, B. Fernando, C.G.M. Snoek, A.W.M. Smeulders, and T. Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013.

[22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[23] Dorit S Hochbaum. Heuristics for the fixed cost median problem. *Mathematical programming*, 1982.

[24] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[25] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *JACM*, 2003.

[26] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[27] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for FGVC: Stanford dogs. *CVPR Workshop on FGVC*, 2011.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[29] Neeraj Kumar, Peter Belhumeur, Arijit Biswas, David Jacobs, W Kress, Ida Lopez, and João Soares. Leafsnap: A computer vision system for automatic plant species identification. *ECCV*, 2012.

[30] Natalia Larios, Bilge Soran, Linda G Shapiro, G Martínez-Muñoz, Junyuan Lin, and Thomas G Dietterich. Haar random forest features and svm spatial matching kernel for stonefly species identification. In *ICPR*, 2010.

[31] S. Lazebnik, C. Schmid, and J. Ponce. A maximum entropy framework for part-based texture and object recognition. In *ICCV*, 2005.

[32] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *Handbook of brain theory*, 1995.

[33] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. *ECCV*, 2012.

[34] Subhransu Maji. Discovering a lexicon of parts and attributes. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 21–30. Springer, 2012.

[35] G. Martínez-Muñoz et al. Dictionary-free categorization of very similar objects. In *CVPR*, 2009.

[36] M-E Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[37] M.E. Nilsback and A. Zisserman. Automated flower classification. In *ICVGIP*, 2008.

[38] Omkar Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012.

[39] Omkar M Parkhi, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. The truth about cats and dogs. In *ICCV*, 2011.

[40] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010.

[41] Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C Berg, Li Fei-Fei, and UNC Chapel Hill. Detecting avocados to zucchinis: what have we done, and where are we going?

[42] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James J Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3D scene understanding. *International Journal of Robotics Research*, 30(13):1543–1552, 2011.

[43] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *ICCV*, 2011.

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

[45] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD Birds 200. 2010.

[46] Shulin Yang, Liefeng Bo, Jue Wang, and Linda G Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012.

[47] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining randomization and discrimination for FGVC. In *CVPR*, 2011.

[48] Bangpeng Yao, Gray Bradski, and Li Fei-Fei. A codebook and annotation-free approach for FGVC. In *CVPR*, 2012.

[49] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013.

[50] Ning Zhang, Ryan Farrell, and Trever Darrell. Pose pooling kernels for sub-category recognition. In *CVPR*, 2012.

[51] Ning Zhang, Ryan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013.