Hierarchical Part Matching for Fine-Grained Visual Categorization

Lingxi Xie
Department of Computer Science and Technology
Tsinghua University
198808xc@gmail.com

Qi Tian
Department of Computer Science
University of Texas at San Antonio
qitian@cs.utsa.edu

Shuicheng Yan
Department of Electrical and Computer Engineering
National University of Singapore
eleyans@nus.edu.sg

Bo Zhang
Department of Computer Science and Technology
Tsinghua University
dcszb@mail.tsinghua.edu.cn

Abstract

Image classification algorithms based on local features, e.g., the Bag-of-Features (BoF) framework, have been proved powerful on various image collections. However, due to the well-known semantic gap between low-level features and high-level image concepts, it still remains a challenging problem to learn the real semantics of the objects. One of the major difficulties comes from the 'learning bias' of traditional image datasets. Objects in such datasets often differ from each other evidently, so that successful classification algorithms often pay more attention on object detection rather than description.

In this respect, we study a special topic of image classification, i.e., Fine-Grained Visual Categorization (FGVC). It provides us good opportunities for a deep learning of high-level concepts. However, traditional BoF model gives poor performances on FGVC tasks, due to the lack of using fine-grained properties. In this paper, we propose a novel model named Hierarchical Part Matching (HPM) by developing three new modules: (1) foreground inference and segmentation; (2) Hierarchical Structure Learning; and (3) Geometric Phrase Pooling. Using ground truth annotations, our approach achieves better image representation, and overwhelmingly outperforms the state-of-the-art algorithms on a challenging FGVC dataset.

1. Introduction

Classification of objects in large-scale image datasets has been a hot topic for many years. It is a basic way towards image understanding and implies a wide range of applications. Today, one of the most popular methods of image classification is to represent images with long vectors, and use a standard classifier for training and testing.

Traditional Bag-of-Features (BoF) framework [5] is widely used for image representation. It is a statistics-based model which summarizes local features in a sparse vector. Despite the great success of the model, it still suffers from the well-known semantic gap between low-level features and high-level concepts [14], as well as the poor object alignment on the images. Recent years, researchers proposed new approaches to deal with the above problems. Successful examples include extracting different kinds of descriptors [2], building mid-level representation [3], spatial weighting [10] and so on. Systems with these new modules produce state-of-the-art classification performances [21], but the connection between image representation and image semantics is still weak.

Fortunately, as evidences accumulate in Neuroscience, researchers realize that human beings recognize objects using a combinational representation of local features [16]. It suggests a structural model for learning high-level concepts. However, traditional image collections usually contain large number of highly irrelevant concepts, which limit the Computer Vision algorithms from learning structural models with few training examples. Therefore, a promis-

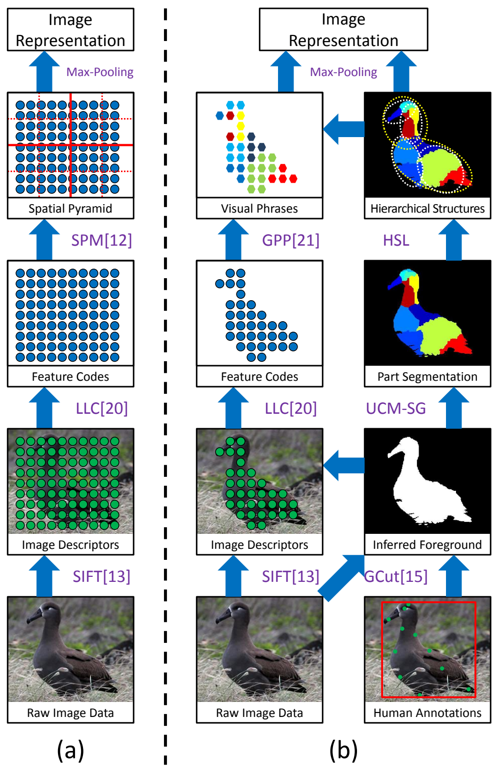

Figure 1. Comparison of the proposed HPM model (b) and the Bag-of-Features (BoF) framework (a) (best viewed in color PDF). HSL (Hierarchical Structural Learning, Section 5) and UCM-SG (UCM-based Segmentation, Section 4) are proposed modules.

ing direction is to consider Fine-Grained Visual Categorization (FGVC), in which we classify image categories sharing similar semantics. We could learn a better structural model using common features in all the categories.

However, the BoF framework gives poor performances on FGVC tasks. The main reason comes from the lack of object alignment, for which we need more ground truth information to conduct an accurate segmentation algorithm. Fortunately, FGVC datasets usually provide extra annotations on the object locations. We could use them to build a much more powerful classification framework.

In this paper, we follow the rationale above and propose the Hierarchical Part Matching (HPM) model for FGVC. Based on the ground truth annotations, we claim a three-fold contribution. First, we infer foreground and segment it into semantic regions. In this way we can depress noises and provide better object alignment. Second, we propose the Hierarchical Structure Learning (HSL) algorithm to find high-level concepts for object recognition. Third, we use the Geometric Phrase Pooling (GPP) algorithm to capture geometric features and combine them with texture ones. Integrating all the modules above gives a powerful model, which outperforms the state-of-the-art algorithms significantly. Figure 1 shows the comparison of our framework (HPM) with the popular BoF model.

The rest of this paper is organized as follows. Section 2 gives a survey of related works. In Section 3, we implement a baseline system for FGVC. We introduce the Hierarchical Part Matching (HPM) model by proposing three modules: foreground inference and segmentation in Section 4, Hierarchical Structure Learning (HSL) in Section 5, and Geometric Phrase Pooling (GPP) in Section 6. After experimental results are shown in Section 7, we draw the conclusions and summarize our future works in Section 8.

2. Related Works

2.1. The Bag-of-Features Framework

The Bag-of-Features (BoF) framework [5] is one of the most widely used models for image representation. The flowchart of this model is illustrated in Figure 1(a).

Starting from raw image data, we first extract SIFT [13] descriptors as local features. After that, a visual vocabulary, or codebook, is trained using K-Means clustering. Locality-sensitive Linear Coding (LLC) [20] is then used for a sparse representation of the descriptors. We use max-pooling for a statistical summarization, and Spatial Pyramid Matching (SPM) [12] for a naive spatial context modeling. Finally, the representation vectors are fed into a linear SVM for training and testing.

2.2. Fine-Grained Visual Categorization

Traditional image classification tasks are aimed at classifying objects with large inter-class differences in semantics. As an example, the Caltech101 Dataset [8] contains 101 concepts including animals, plants and man-made tools. To achieve a good performance on such a task, more attentions are paid on object detection and alignment rather than learning a high-level structural model, which tends to discriminate different objects with some robust features.

Oppositely, Fine-Grained Visual Categorization (FGVC) is an emerging topic in Computer Vision. A fine-grained image collection typically contains hundreds of categories sharing similar semantics. For example, the Caltech-UCSD Birds-200-2011 Dataset [19] contains 200 bird species, and there are 120 different kinds of dogs in the Stanford Dogs Dataset [11]. It is even difficult for a well-trained human to recognize all these categories. Also, FGVC datasets usually provide extra annotations for object alignment. It becomes more important to extract discriminative features and learn descriptive structures, in order to find the small inter-class variations. Researchers proposed to use Visual Attributes [6], random templates [22], or hierarchical matching [4] for FGVC, but the reported performances are still poor compared with traditional classification tasks.

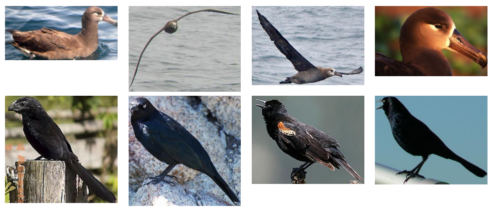

Figure 2. Samples from the Caltech-UCSD Birds-200-2011 Dataset [19]. Upper: images from the same category (001. Black-footed Albatross) showing large intra-class variation. Lower: images of different species showing small inter-class variation.

2.3. Foreground Inference and Segmentation

It is important to separate objects from background clutters, especially in FGVC, in which all the categories share similar background features. Interactive foreground inference is a well-studied topic in Computer Vision. A widely used tool is the Grab-Cut [15] algorithm, which iteratively infers the region of interest from a supervised mask matrix. A standard implementation of Grab-Cut is contained in the newest release of the OpenCV Library.

After the object is located, we further segment it into parts for fine-grained alignment. In this respect, we need to detect the boundaries, which are defined as pixels dividing regions with different semantics. The Ultrametric Contour Map (UCM) [1] is an unsupervised segmentation algorithm, which hierarchically divides an image into smaller and smaller regions.

2.4. The Geometric Phrase Pooling Algorithm

The basic units in the BoF framework are visual words, which are lack of semantics and produce poor pooling results. Visual phrases, or groups of visual words, are proved very useful for image classification [23] and retrieval [25].

In [21], Xie *et al.* proposed a novel algorithm named Geometric Phrase Pooling (GPP). By defining neighboring groups of visual words as Geometric Visual Phrases (GVP), it is possible to perform an efficient pooling algorithm to capture both geometric and feature similarities. Experimental results reveal that GPP is a supreme spatial context modeling approach towards better image representation.

3. Dataset and Baseline System

3.1. The Dataset

In our experiments, we use the Caltech-UCSD Birds-200-2011 (CUB-200-2011) Dataset [19], which contains 200 bird species and 11788 images in total. Also, a human-labeled bounding box and at most 15 part locations are provided for each image. It is a very challenging dataset. For the birds shown in Figure 2, it is even difficult for human beings to recognize them accurately. The main reason for choosing this dataset is the ground truth part locations, which are very important to our approach as they provide key information for object alignment.

People might argue that annotating all the part locations could be manually expensive, and debate on inferring the bounding boxes and part locations automatically. However, due to the poor performances of state-of-the-art part-based detectors [9], we simply use the ground truth annotations to bypass the difficulties in detection and focus on learning the fine-grained concepts.

3.2. The Baseline System

We use the Bag-of-Features (BoF) framework as our baseline system. Here, we build a mathematical notation system for this model.

In the Birds Dataset, a human-annotated **bounding box** is provided for each image. We use images within bounding boxes, and resize them into the same size, *i.e.*, width and height do not exceed 300. Denote a raw image as $\mathbf{I}$:

$$\mathbf{I} = (a_{ij})_{W \times H} \tag{1}$$

where a_{ij} is the **pixel** at position (i, j), and it is a 3-dimensional vector for RGB-images. Also, for each image, there are at most L **part locations** ($L = 15$ for the Birds Dataset). Denote the part locations as $\{p_l\}$, where $l = 1, 2, \ldots, L$.

We extract SIFT descriptors [13] on the image and obtain a set of local descriptors $\mathcal{D}$:

$$\mathcal{D} = \{(\mathbf{d}_1, \mathcal{R}_1), (\mathbf{d}_2, \mathcal{R}_2), \ldots, (\mathbf{d}_M, \mathcal{R}_M)\} \tag{2}$$

where $\mathbf{d}_m$ and $\mathcal{R}_m$ denote the **description vector** and **occupied region** of the m-th descriptor, respectively. M is the total number of descriptors, which could be hundreds or even thousands under dense sampling. The description vector $\mathbf{d}_m$ is a D-dimensional vector, where $D = 3 \times 128 = 384$ using OpponentSIFT (OppSIFT) [17] on RGB-images.

After descriptors have been extracted, they are quantized to be compact. For this purpose, we train a **codebook** $\mathbf{C}$ using descriptors from the whole dataset. $\mathbf{C}$ is a $B \times D$ matrix consisting of B vectors with dimension D, each of which is called a **codeword**. The number of codewords, or the **codebook size** B, is 2048 in our experiments.

Next, descriptors are represented using the codebook. This process is called **coding**, for we encode each descriptor as a sparse vector. We use the Locality-constrained Linear Coding (LLC) [20] algorithm. Given a codebook with B codewords, the quantization vector or **feature vector** for a descriptor $\mathbf{d}_m$ would be a B-dimensional vector $\mathbf{w}_m$, which is named the corresponding **visual word** of **descriptor** $\mathbf{d}_m$. $\mathcal{W}$ is the set of visual words:

$$\mathcal{W} = \{(\mathbf{w}_1, \mathcal{R}_1), (\mathbf{w}_2, \mathcal{R}_2), \ldots, (\mathbf{w}_M, \mathcal{R}_M)\} \tag{3}$$

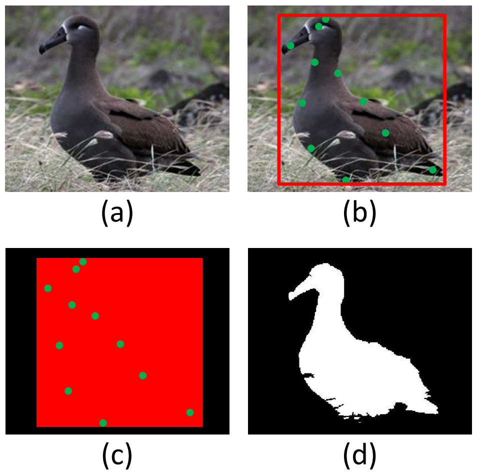

Figure 3. Foreground inference (best viewed in color PDF). (a) Original image. (b) Bounding box (red) and small areas around part locations (green). (c) Initial mask required by Grab-Cut, in which black, red and green regions are definite BG, possible FG and definite FG, respectively. (d) Inferred foreground (white).

Now, we aggregate the visual words for image representation. The **max-pooling** strategy calculates the maximal responses on each codeword:

$$\mathbf{f} = \max_{1 \leqslant m \leqslant M} \mathbf{w}_m \tag{4}$$

where the notation $\max_m$ denotes the element-wise maximization. A 3-layer Spatial Pyramid Matching (SPM) [12] follows by dividing the image into hierarchical subregions for individual max-pooling and concatenating pooled vectors as a **supervector**. Finally, the supervectors are fed into a linear SVM for classification.

4. Foreground Inference and Segmentation

4.1. Foreground Inference

A notable property of fine-grained datasets is the high similarity in backgrounds. Take the Birds Dataset as an example. Water surfaces, trees and grasses appear in almost all the categories. They introduce noises into the classification model. Therefore, it is reasonable to infer foreground regions before extracting robust descriptors.

Figure 3 illustrates the inference process. After collecting ground truth annotations, *i.e.*, the bounding box and part locations, we use them to construct the initial mask matrix for the Grab-Cut algorithm [15]. Set the pixels outside the bounding box as **definite background**, inside as **possible foreground** and the pixels around part locations as **definite foreground**. With no more than 10 iterations, the Grub-Cut algorithm gives the inference result.

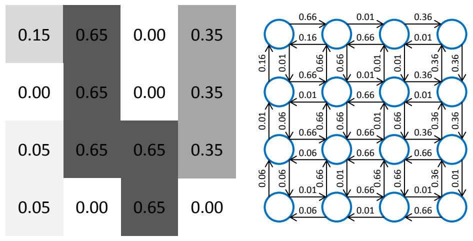

Figure 4. Construction of graph $\mathcal{G}$. Left: a small patch on UCM, where each grid is a pixel. Numbers on the grids are boundary intensities. Right: constructed subgraph with edge weights shown on the arcs. Step penalty λ is 0.01.

4.2. UCM-Based Graph Construction

Since different birds vary a lot in their poses, naive Spatial Pyramid produces poor matching accuracies. To achieve better object segmentation, we need to define a distance metric for each pair of pixels. For this, we first calculate the Ultrametric Contour Map (UCM) [1], which generates closed contours with hierarchically decreasing boundary intensities, and cuts the image into smaller and smaller regions. Denote the intensity map as $\mathbf{U}$:

$$\mathbf{U} = (u_{ij})_{W \times H} \tag{5}$$

where u_{ij} is the **boundary intensity** at position (i, j).

Based on UCM, we construct a **directed graph** $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_{ij}\}$ consists of all the pixels, and $\mathcal{E}$ is compromised of edges connecting adjacent pixels:

$$\mathcal{E} = \{(v_{ij} \to v_{i'j'}) \mid |i - i'| + |j - j'| = 1\} \tag{6}$$

The weight of an edge is related to the boundary intensity at the tail node (pixel):

$$w(v_{ij} \to v_{i'j'}) = u_{i'j'} + \lambda \tag{7}$$

Here, λ is called the **step penalty**, which takes the geometric distance into consideration. Figure 4 shows a sample graph $\mathcal{G}$ constructed on a small patch.

After the graph is complete, we take the part locations as source nodes, and calculate their **shortest paths** to other nodes (pixels). The Dijkstra algorithm gives a solution within $O(LN\log(N))$ time, where L is the number of parts and $N = W \times H$ is the image size. Denote the distances as $\{d(p_l, v_{ij})\}$, where p_l is the l-th part location, and v_{ij} is an arbitrary node in graph $\mathcal{G}$. For later convenience, we define $\{d(0, v_{ij})\}$ as the **background distances**, which takes 0 for background nodes and $+\infty$ for foreground ones.

4.3. Segmentation Strategies

The segmentation process is to assign each node (pixel) to one of the part locations. Denote an assignment as $\mathbf{S}$:

$$\mathbf{S} = (s_{ij})_{W \times H} \tag{8}$$

where s_{ij} is the index of assigned part location of node v_{ij}, $0 \leqslant s_{ij} \leqslant L$, and $s_{ij} = 0$ implies a background node. We minimize the cost function:

$$f(\mathbf{S}) = \sum_{i,j} d\big(p_{s_{ij}}, v_{ij}\big) \tag{9}$$

with an independent minimization on each node (pixel):

$$s^{\star}_{ij} = \arg \min_{0 \leqslant l \leqslant L} d(p_l, a_{ij}) \tag{10}$$

This is called the **Naive Segmentation** (NS) strategy.

To achieve better segmentation results, we need to slightly modify the cost function:

- **The reject option** assigns some foreground nodes to background by setting $d(0, v_{ij}) \equiv \tau$ for foreground nodes, where τ is a fixed **background penalty**.
- **The discontinuous penalty** penalizes the neighboring nodes which are assigned to different parts. A fixed value η is added for each pair of such nodes.

Taking both modifications gives the renewed cost function:

$$f(\mathbf{S}) = \sum_{i,j} d\big(p_{s_{ij}}, v_{ij}\big) + \sum_{\substack{|i-i'|+|j-j'|=1 \\ s_{ij} \neq s_{i'j'}}} \eta \tag{11}$$

Directly minimizing (11) is intractable. We exploit a two-step algorithm for an approximate optimization.

1. Directly use (10) on the modified distances for an initial minimization.
2. Use an iterative algorithm to refine the segmentation on its discontinuity. Search for each connective branch on graph $\mathcal{G}$, and try to assimilate it into one of its neighboring regions. If the modification gives a smaller cost function value, we keep it going on. The iteration stops if no change is performed in a complete pass.

The improved algorithm is named the **Refined Segmentation** (RS) strategy. Figure 5 illustrates the difference between the above two segmentation methods.

5. Hierarchical Structure Learning

Denote the set of all pixels as $\mathcal{I}$. Segmentation algorithm divides $\mathcal{I}$ into at most L foreground parts and one background region:

$$\mathcal{I} = \bigcup_{l=0}^{L} \mathcal{I}_l \tag{12}$$

where $\mathcal{I}_l$ represents the l-th body part for $l > 0$, and $\mathcal{I}_0$ is the background. All the segmented regions are exclusive, *i.e.*, $\mathcal{I}_{l_1} \cap \mathcal{I}_{l_2} = \varnothing$ for $l_1 \neq l_2$.

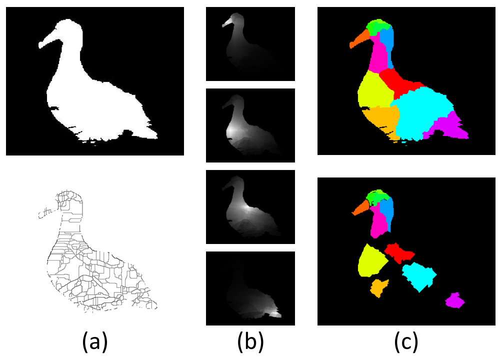

Figure 5. Segmentation illustration. (a) inferred foreground and UCM (darker pixel, larger intensity). (b) heatmap of distance from 'beak', 'breast', 'back' and 'tail', respectively. (c) Naive (above) and Refined (below) Segmentation. Refined strategy produces better classification results (see Table 2) by assigning ambiguous boundary pixels to background and removing small noisy patches.

$\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_L$ are low-level concepts for object understanding, *i.e.*, 'nape', 'left eye', 'right leg', etc. Semantically, there exist high-level concepts. For example, the concept 'eyes' consists of 'left eye' and 'right eye', and 'head' is compromised of 'forehead', 'crown', 'beak', and 'eyes'. Learning a hierarchy to organize low-level parts could be useful for a better structural model.

Close geometric locations and similar appearance features are necessary conditions for combining parts. Therefore, we need to quantize the geometric and feature distances for pairwise parts. Recall that for each image, we have a set of descriptors $\mathcal{D}$ as defined in (2), at most L part locations $p_1, p_2 \ldots, p_L$ and segmented regions $\mathcal{I}_1, \mathcal{I}_2 \ldots, \mathcal{I}_L$ obtained from the previous section. For an image in which $\mathcal{I}_{l_1}$ and $\mathcal{I}_{l_2}$ are both non-empty, the **geometric distance** between them is formulated as:

$$\mathrm{dist}_{\mathrm{g}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \Big[\big(p^{\mathrm{X}}_{l_1} - p^{\mathrm{X}}_{l_2}\big)^2 + \big(p^{\mathrm{Y}}_{l_1} - p^{\mathrm{Y}}_{l_2}\big)^2\Big]^{1/2} \tag{13}$$

and the **feature distance** is calculated as:

$$\mathrm{dist}_{\mathrm{f}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \Bigg\| \operatorname*{avg}_{\mathcal{R}_m \cap \mathcal{I}_{l_1} \neq \varnothing} \mathbf{d}_m - \operatorname*{avg}_{\mathcal{R}_m \cap \mathcal{I}_{l_2} \neq \varnothing} \mathbf{d}_m \Bigg\|_2 \tag{14}$$

Integrating (13) and (14) yields the **total distance**:

$$\mathrm{dist}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) = \frac{\mathrm{dist}_{\mathrm{g}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2})}{\max \mathrm{dist}_{\mathrm{g}}(\mathcal{I})} + \frac{\mathrm{dist}_{\mathrm{f}}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2})}{\max \mathrm{dist}_{\mathrm{f}}(\mathcal{I})} \tag{15}$$

where both distances are normalized. Finally, we average the total distance over all the images and obtain the **part distance** for l_1 and l_2.

$$\mathrm{dist}(l_1, l_2) = \operatorname*{avg}_{\mathcal{I}_{l_1} \cap \mathcal{I}_{l_2} \neq \varnothing} \mathrm{dist}(\mathcal{I}_{l_1}, \mathcal{I}_{l_2}) \tag{16}$$

Table 1. As the learning parameter μ increases, the learned structures are more and more complex. Bolded are the semantically nameable parts learned by our algorithm.

No.	μ	Learned Hierarchical Structures
#0	0.0	No high-level parts are learned.
#1	0.1	**eyes** (left/right eye), **legs** (left/right leg), **wings** (left/right wing).
#2	0.3	**eyes**, **legs**, **wings**, **neck** (nape/throat).
#3	0.5	**eyes**, **legs**, **wings**, **neck**, **head** (beak/crown/forehead/eyes), **body** (back/belly/breast/tail).
#4	1.0	**eyes**, **legs**, **wings**, **neck**, **head**, **body**, (**wings/legs**), (**body/wings/legs**), **ALL**.

A **combining operation** $\mathcal{C}$ is a set of existed parts:

$$\mathcal{C} = \{c_1, c_2, \ldots, c_T\} \tag{17}$$

where $2 \leqslant T \leqslant L$. The cost for the operation is:

$$\operatorname{cost}(\mathcal{C}) = \frac{\sum_{1 \leqslant i < j \leqslant T} \operatorname{dist}(c_i, c_j)}{\frac{1}{2} T (T-1)} \tag{18}$$

Now, the **Hierarchical Structure Learning** (HSL) algorithm is very easy to implement.

1. **Initialization.** Start from the original part set $\mathcal{P} = \{1, 2, \ldots, L\}$ and a pre-defined learning parameter μ.
2. **Learning.** Enumerate all the subsets $\mathcal{P}' \subseteq \mathcal{P}$, and calculate the cost function $c = \operatorname{cost}(\mathcal{P}')$. If $c \leqslant \mu$, then take $\mathcal{P}'$ to generate a learned part.
3. **Construction.** Organize all the original and learned parts as a hierarchical structure.

Denote $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{\tilde{L}}\}$ as all the original and learned parts, where $\tilde{L} \geqslant L$ is the number of parts.

Obviously, the **learning parameter** μ controls the acceptance degree of our algorithm: the learned structure becomes more and more complex as μ increases. We list some values of μ and the learned structures in Table 1. It is obvious that the learned parts are semantically nameable (bolded in the table). Also, the learned structure is hierarchical when $\mu = 0.5$ and $\mu = 1.0$. Both the above observations reveal the effectiveness of our algorithm.

6. Geometric Pooling Strategy

We return to the original image $\mathbf{I}$. After descriptor extraction and feature coding, we obtain the set of descriptors $\mathcal{D}$ defined in (2) and the corresponding set of visual words $\mathcal{W}$ defined in (3), respectively. Also, we have at most $\tilde{L}$ regions $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{\tilde{L}}\}$ for each image.

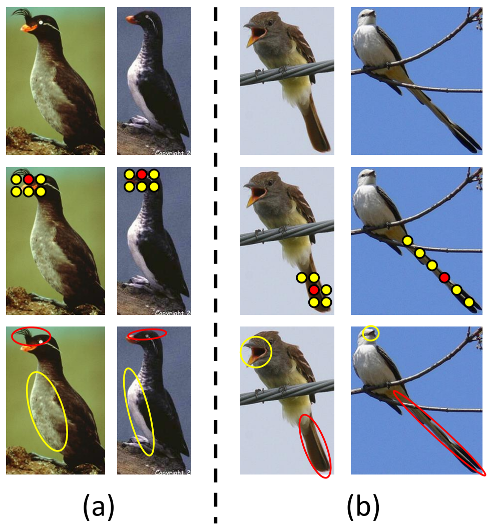

Figure 6. Examples illustrating Geometric Phrase Pooling (best viewed in color PDF). Upper: images differing from each other mainly in 'crown' shape (left pair) and 'tail' length (right pair). Middle: examples of Geometric Visual Phrases (GVP), in which red circles are central words and yellows are side ones. The GVP in the last case is irregular, for the definition limits the side words to the same region as the central word (the long 'tail' in this case). Bottom: the regions (of same color) with largest discriminativity increases. The differences in 'crown' and 'tail' are detected.

The **Naive Pooling** (NP) strategy summarizes each region by finding all the related features:

$$\mathcal{W}_l = \{(\mathbf{w}_m, \mathcal{R}_m) \mid \mathcal{R}_m \cap \mathcal{I}_l \neq \varnothing\} \tag{19}$$

and performing max-pooling:

$$\mathbf{f}_l^{(\mathrm{W})} = \max_{(\mathbf{w}_m, \mathcal{R}_m) \in \mathcal{W}_l} \mathbf{w}_m \tag{20}$$

Despite its simplicity, the Naive Pooling strategy fails to capture the geometric information, which could be very useful for fine-grained recognition. Figure 6 shows such examples with dominant geometric features, such as 'crown' shape and 'tail' length.

The Geometric Phrase Pooling (GPP) algorithm [21] is efficient at spatial context modeling. It defines Geometric Visual Phrases (GVP) as neighboring word groups, and performs an efficient pooling algorithm to enhance the correlation between local word pairs. In FGVC, GPP is performed within each segmented region $\mathcal{I}_l$ and the corresponding set $\mathcal{W}_l$. For each visual word $(\mathbf{w}_m, \mathcal{R}_m)$ in $\mathcal{W}_l$, we search for its K nearest neighbors in $\mathcal{W}_l$ and form a word group:

$$\mathcal{P}_{l,m} = \{(\mathbf{w}_{l,m,0}, \mathbf{l}_{l,m,0}), \ldots, (\mathbf{w}_{l,m,K}, \mathbf{l}_{l,m,K})\} \tag{21}$$

$\mathcal{P}_{l,m}$ is the m-th **Geometric Visual Phrase** (GVP) in $\mathcal{W}_l$, in which $\mathbf{w}_{l,m,0} = \mathbf{w}_m$ is the **central word**, and others are

side words. K is the **order** of $\mathcal{P}_{l,m}$, which is 20 as in [21]. Figure 6 illustrates some examples of visual phrases.

The **Geometric Phrase Pooling** (GPP) algorithm calculates the following representation vector on $\mathcal{P}_{l,m}$:

$$\mathbf{p}_{l,m} = \mathbf{w}_{l,m,0} + \max_{1 \leqslant k \leqslant K} \mathbf{w}_{l,m,k} \tag{22}$$

and summarizes all the phrases using max-pooling:

$$\mathbf{f}_l^{(\mathrm{P})} = \max_{(\mathbf{w}_m, \mathcal{R}_m) \in \mathcal{W}_l} \mathbf{p}_{l,m} \tag{23}$$

In [21], Xie *et al.* claimed that GPP actually finds those local features both similar and adjacent, and enhances the responses in their common stimulated dimensions. Here, we conduct an experiment to show the effectiveness of GPP. We perform GPP on the image pairs shown in Figure 6, and calculate the distance ϕ_l between the corresponding regions using visual words or phrases. ϕ_l is considered as the model's **discriminativity metric**:

$$\phi_l^{(\mathrm{W})} = \left\| \mathbf{f}_l^{(\mathrm{W})}(\mathbf{I}_1) - \mathbf{f}_l^{(\mathrm{W})}(\mathbf{I}_2) \right\|_2^2 \tag{24}$$

$$\phi_l^{(\mathrm{P})} = \left\| \mathbf{f}_l^{(\mathrm{P})}(\mathbf{I}_1) - \mathbf{f}_l^{(\mathrm{P})}(\mathbf{I}_2) \right\|_2^2 \tag{25}$$

We calculate the discriminativity increase $\phi_l^{(\mathrm{P})} - \phi_l^{(\mathrm{W})}$, and circle out regions with largest values in Figure 6. The results reveal that GPP actually discovers useful geometric features and combines them with the texture features.

7. Experiments

This section gives **classification results** on the Caltech-UCSD Birds-200-2011 Dataset [19]. To make comparison, we keep the same settings as the state-of-the-art algorithms:

- **Local descriptors.** We use the VLFeat [18] library to extract OppSIFT descriptors [17]. Step size and scale of the sliding window are 5 and 6, respectively.
- **Codebook learning.** We train a 2048-entry codebook with K-Means clustering. The number of descriptors collected for training is around 2 million.
- **Coding and pooling.** We use LLC [20] for coding, and max-pooling for statistics.
- **Classification.** A linear SVM, LibLinear [7], is used for training and testing. The penalty parameter C for slack variables is 10.
- **Accuracy evaluation.** We use random data split to test our algorithm. The random selection is repeated 10 times and we report the average accuracies.

Table 2. **Classification accuracies** (%) before and after foreground inference and segmentation. **FG Inference**: a 3-layer SPM on the foreground. **Naive** and **Refined Segmentation**: using naive or refined segmented regions as spatial bins for pooling.

# training	5	10	20	40
Baseline	13.64	20.25	28.36	37.77
FG Inference	19.25	27.66	37.08	46.59
Naive Seg.	27.01	38.71	50.90	60.92
Refined Seg.	**28.55**	**40.46**	**52.52**	**62.16**

Table 3. The HSL algorithm produces higher **classification accuracies** (%). See Table 1 for details of the structures. **Structure #**0, *i.e.*, No Structure, is just the Refined Segmentation in Table 2.

# training	5	10	20	40
Structure #0	28.55	40.46	52.52	62.16
Structure #1	29.29	41.62	53.36	62.74
Structure #2	29.75	42.03	53.55	62.59
Structure #3	**30.33**	**42.66**	**53.94**	**63.21**
Structure #4	27.38	38.64	50.22	59.71

Table 4. **Classification accuracies** (%) using Geometric Phrase Pooling. **No Phrase** is the same as Structure #3 in Table 3. In brackets are numbers of coding bases for central and side words.

# training	5	10	20	40
No Phrase	30.33	42.66	53.94	63.21
GPP(5,5)	31.69	43.80	55.26	64.65
GPP(5,10)	32.23	45.10	56.11	65.26
GPP(5,20)	34.13	47.29	58.60	67.14
GPP(5,40)	**36.09**	**48.87**	**60.56**	**69.07**

7.1. Model and Parameters

First, we test foreground inference and segmentation. Results are listed in Table 2. After filtering noises and conducting better spatial alignment, we obtain much better performances beyond the baseline system. The Refined Segmentation (RS) strategy gives the best performance, and is preserved for the next step.

Second, we test the Hierarchical Structure Learning (HSL) algorithm. We use the learned structures in Table 1 and list the corresponding results in Table 3. Using high-level concepts as extra bins, we improve image representation and achieve better accuracies. Exceptions come from the last case (Structure #4) where the structure is too complex. Therefore, we preserve Structure #3 for the next step.

Finally, we test the Geometric Phrase Pooling algorithm. Results are listed in Table 4. We see that GPP indeed provides a better solution for spatial context modeling. We

Table 5. **Classification accuracies** (%) on the Birds Dataset using random data split. LLC [20] is the baseline system.

# training	5	10	20	40
Wah [19]	10.05	-	-	-
Wang [20]	13.64	20.25	28.36	37.77
Xie [21]	15.34	22.91	31.01	40.43
Ours (Mean)	**36.09**	**48.87**	**60.56**	**69.07**
Ours (StdDev)	±0.31	±0.60	±0.50	±0.48

Table 6. **Classification accuracies** (%) on the Birds Dataset using the fixed training/testing data split provided by [19].

Wah [19]	Zhang [24]	Wang [20]	Xie [21]	Ours
17.31	24.21	33.91	36.33	**66.35**

choose 5 and 40 as the numbers of coding bases for central and side words, respectively.

It is reasonable to conduct a comparison between our Hierarchical Part Matching (HPM) model and the Spatial Pyramid Matching (SPM) [12]. SPM divides an image into fixed patches and uses a fixed hierarchical structure to organize them. However, given ground truth annotations, it is possible to perform better object alignment, and learn semantical hierarchical structures on the segmented regions. HPM is not only a supporter of SPM which proves that well-aligned parts and well-organized structures are useful for discrimination, but also a contradictor showing that using global features is not always better (See Structure #3 and #4 in Table 1 and 3. #4 uses global features and produces much lower accuracies than #3).

The length of image representation vectors in HPM is 21 (total number of parts in Structure #3) times the codebook size, which is the same as a 3-layer SPM.

7.2. Comparison to Other Works

Finally, we compare our system (HPM) with existing works **using ground truth part annotations**. We inherit the best parameters from the previous part, *i.e.*, Refined Segmentation, Structure #3 in Table 1, and numbers of coding bases (5,40) for GPP. Table 5 and 6 show the results on random and fixed data split, respectively. HPM overwhelmingly outperforms the existing algorithms.

8. Conclusions and Future Works

In this paper, we present a novel framework named Hierarchical Part Matching (HPM) for Fine-Grained Visual Categorization (FGVC). HPM contains three novel modules to enhance the BoF model. First, using the Grab-Cut algorithm and the Ultrametric Contour Map, we develop an effective algorithm for foreground inference and segmentation. It generates more accurate object alignment, which lays the foundation of FGVC. Second, we propose the Hierarchical Structure Learning (HSL) algorithm for finding high-level concepts beyond basic parts. The learned parts are semantically nameable. Third, we use the Geometric Phrase Pooling (GPP) algorithm for spatial context modeling. Integrating all the modules makes HPM a powerful model for FGVC tasks, which shows notable improvements over the existing works on the Birds Dataset.

Despite the leap in classification accuracies, our HPM model is still imperfect. Foreground inference and segmentation algorithms suffer from clutters as well as illuminance changes, and produce poor results on about 20% images. This introduces noises into our model. Also, biological taxonomy provides a scientific classification system for all the bird species (available on Wikipedia). It implies a hierarchical classifier, on which we could apply various techniques such as Transfer Learning for fine-grained understanding. We will investigate these problems in our future works and look forward to a better model for FGVC tasks.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From Contours to Regions: An Empirical Evaluation. *CVPR*, 2009.

[2] A. Bosch, A. Zisserman, and X. Muoz. Image Classification using Random Forests and Ferns. *ICCV*, 2007.

[3] Y. Boureau, F. Bach, Y. LeCun, J. Ponce, et al. Learning Mid-Level Features for Recognition. *CVPR*, 2010.

[4] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. Hierarchical Matching with Side Information for Image Classification. *CVPR*, 2012.

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[6] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering Localized Attributes for Fine-Grained Recognition. *CVPR*, 2012.

[7] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 2008.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *CVIU*, 2007.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *PAMI*, 2010.

[10] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-Norm Feature Pooling for Image Classification. *CVPR*, 2011.

[11] A. Khosla, N. Jayadevaprakash, B. Yao, and F. Li. Novel Dataset for Fine-Grained Image Categorization. *First Workshop on FGVC, CVPR*, 2011.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *CVPR*, 2006.

[13] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004.

[14] Y. Lu, L. Zhang, J. Liu, and Q. Tian. Constructing Lexica of High-level Concepts with Small Semantic Gap. *IEEE Transactions on Multimedia*, 2010.

[15] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, 2004.

[16] S. Ullman. Object Recognition and Segmentation by a Fragment-Based Hierarchy. *Trends in Cognitive Science*, 2007.

[17] K. Van De Sande, T. Gevers, and C. Snoek. Evaluating Color Descriptors for Object and Scene Recognition. *PAMI*, 2010.

[18] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. *ACM Multimedia*, 2010.

[19] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report*, 2011.

[20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *CVPR*, 2010.

[21] L. Xie, Q. Tian, and B. Zhang. Spatial Pooling of Heterogeneous Features for Image Applications. *ACM Multimedia*, 2012.

[22] B. Yao, G. Bradski, and L. Fei-Fei. A Codebook-Free and Annotation-Free Approach for Fine-Grained Image Categorization. *CVPR*, 2012.

[23] J. Yuan, M. Yang, and Y. Wu. Mining Discriminative Co-occurrence Patterns for Visual Recognition. *CVPR*, 2011.

[24] N. Zhang, R. Farrell, and T. Darrell. Pose Pooling Kernels for Sub-Category Recognition. *CVPR*, 2012.

[25] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive Visual Words and Visual Phrases for Image Applications. *ACM Multimedia*, 2009.